Credit Default Swap Rate Construction Methods by Machine Learning Techniques

Ahmet Baglan
Mehmet Eyyupoglu

Kongens Lyngby 2019
DTU-Compute-M.Sc and B.Sc-2019

Technical University of Denmark
Department of Applied Mathematics and Computer Science
Richard Petersens Plads, building 324,
2800 Kongens Lyngby, Denmark
Phone +45 4525 3351
compute@compute.dtu.dk
www.compute.dtu.dk

Summary

Credit Default Swap(CDS) is a financial instrument that provides a form of insurance to eliminate possible costs arising from default of a counterparty for the bond owners. However, since the CDS market is not developed enough, the majority of the entities do not have liquid CDS quotes. This is the reason why it is important to either find a proxy entity or directly estimate spread values to construct historical CDS spread time series.

In this project, we have researched the possibility to use modern Machine Learning methods from two different perspectives. Firstly, we have run classification models to find the best proxy of a given entity for constructing the CDS spread time series using the proxy entity's spread. Secondly, we have run regression models to directly estimate CDS spread.

After comparison of 37 different models, we have concluded that it is possible to improve currently used method, i.e Cross-Section method, by using Machine Learning based algorithms. As the main contribution to the field, we have proposed different models for solving the problem. Among the models a Deep Learning model using a simple Long short-term memory architecture was found to be outperforming all of the current methods.

Preface

This thesis was prepared at the department of Applied Mathematics and Computer Science at the Technical University of Denmark (DTU) in fulfillment with the requirements of acquiring a M.Sc. Digital Media Engineering for Ahmet Baglan(s171787) and B.Sc. General Engineering for Mehmet Eyyupoglu(s174448). The project is written in collaboration with the risk models department of Nordea Bank Danmark A/S.

The thesis presents work conducted during the spring semester of 2019. It was written under the supervision of Professor Ole Winther with the co-supervision of Dr. Alexander Subbotin and front office quantitative analyst Shengyao Zhu.

Lyngby, 29-June-2019

Ahmet Baglan
Mehmet Eyyupoglu

Acknowledgements

We would like to sincerely thank to our supervisors Ole Winther, Alexander Subbotin and Shengyao Zhu.

We would also like to thank our families who always cheered us whenever we really needed it. This accomplishment would not have been possible without them.

Contents

Chapter 1

Introduction

Credit risk is one of the main risk types for financial institutions that have relations with many credit counterparties. For a healthy financial sector and a healthy economy, this risk should be managed meticulously. This is the reason why financial institutions are required to manage *counterparty*[1] *default*[2] risk. Yet, the risk should be measured before handling. However, the probability of default for a given party cannot be easily calculated using the historical data. This is because defaulting is a comparatively rare event. Thus, financial institutions use liquid credit *derivatives*[3] to obtain the market implied probability defaults.

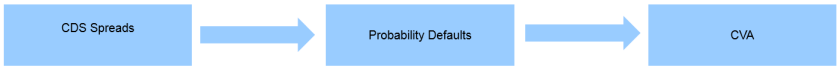
Credit derivatives form a type of insurance for credit risk. A third party covers the credit risk in return for a premium fee. As explained at [33] and ([36] p.402),the main aim for the credit derivatives is repackaging credit risk for more efficient credit transfer. Protection fee is a variable dependent on the riskiness

[1] A counterparty (sometimes contraparty) is a legal entity, unincorporated entity, or collection of entities to which an exposure to financial risk might exist. *Source: Wikipedia/Counterparty*

[2] In finance, default is failure to meet the legal obligations (or conditions) of a loan, for example when a home buyer fails to make a mortgage payment, or when a corporation or government fails to pay a bond which has reached maturity. *Source: Wikipedia/Counterparty*

[3] A derivative is a financial security with a value that is reliant upon or derived from, an underlying asset or group of assets—a benchmark. The derivative itself is a contract between two or more parties, and the derivative derives its price from fluctuations in the underlying asset. *Source: Investopedia/Derivative*

Figure 1.1: High level explanation of why there is a need for CDS Spreads



of the underlying asset. If the risk of loss increases, so does the fee. The credit derivative market works similarly to other markets. Thus, credit derivatives could be both traded for hedging and speculative intentions ([36] p.402).

Credit Default Swaps(CDS) are by far the most common and liquid credit derivatives. A CDS spread of a given party is essentially an indicator that shows the market expectation for the default of the party. The higher the protection fee(CDS Spread), the higher the probability default is according to the market. However, as stated in European Banking Authority survey (EBA 2015), for many institutions such a derivative is not available in the market. Also, in many cases it is very difficult to obtain the data that fully covers all the counterparties that a financial institution have credit relations to. Thus, in such cases proxy CDS spreads are incorporated in the exposure calculations, specifically in *Credit Value Adjustment*[4](CVA) calculations. This high level relationship can be seen at Figure 1.1.

In this project, we have used Machine Learning techniques to construct historical proxy CDS spreads for a given unobservable counterparty. The main scenario is the following. In January 2018, an investor wants to learn what would be the CDS spread time series of the counterparty that has just issued a new bond. The investor has a risk calculation model that requires CDS spread of the counterparty for the whole year of 2017. However, there is no data for such a CDS spread for 2017 because the company is relatively small and there is no CDS product covering its bonds in the market. Thus, the investor needs to come up with an approximation based on the observable companies. In this scenario, it should be possible to use the models investigated in this work to construct such a time series.

Note that even though the work is fully constructed and tested on liquid CDS spreads, there should be no limitations for using these methods for other credit derivatives.

[4]A calculation of the market value for counterparty risk. It is further discussed at the theory section.

1.1 Problem motivation

The problem studied in this project is the lack of data in the Credit Default Swap spreads. This is either due to the lack of a liquid CDS spread for a given counterparty or because of data coverage issues.

Currently used models for solving the problem only use sector, region and rating of a counterparty for CDS spread proxying. However, stock market is much deeper and liquid compared to the CDS market, and daily stock price returns inherently has the information about the riskiness of a counterparty. This is because stock returns generally show higher variance for riskier counterparties. Thus, it could be possible to infer CDS spreads using easily accessible data. Moreover, current models are relatively simple models and have not yet implemented new Machine Learning methods. Thus there is room for improvement.

1.2 Goal

The main goal of this project is to construct historical Credit Default Swap spread time series by using daily stock prices, daily credit ratings and other easily accessible data about a counterparty. The models developed are to be compared with the existing models to analyze performance.

1.3 Thesis scope

Throughout the project, different boundaries have been set to limit the scope of the project to keep it feasible within the allowed time frame.

The main and the most important boundary is that this work does not cover any forecasting. The aim is to construct historical time series. Thus, for all the predictions made, it is assumed that the models developed could be informed of all the input variables without any problem.

Secondly, the models in principle were developed by keeping a general perspective to be usable for other similar financial products. However, in this work, it was only applied with the most liquid Credit Default Swap products.

Thirdly, there are many different types of Machine Learning models. In this

work, a limited number of them was implemented and applied to solve the proxy problem.

Finally, this not a research about a improving performance of a specific model but this is a research on application of Machine Learning models to the CDS construction problem. Thus, even though, models are optimized, especially for complicated models, there could be room left for further optimization.

1.4 Report structure

The report is structured as the following:

- **Introduction**: Introduction to the problem motivation, the goal of the project and scope of the work.
- **Theory**: General theory about the financial product and different Machine Learning models used throughout the project, including different classification and regression models, their limitations and evaluation methods.
- **Related work**: Description of different models that are currently being used for the CDS proxy problem.
- **Data**: Analysis of the data used and proof of motivation.
- **Implementation**: Software and hardware specifications that we used in this project.
- **Model evaluation and running flow**: Evaluation metrics and experimental setup.
- **Experiments & results**: Description of the different models developed as well as obtained results.
- **Discussion**: Discussion about the results from the experiments. Comparison of models.
- **Conclusion & future work**: The conclusion of the project and possible research points that could not be analyzed further in the scope of this project.

Chapter 2

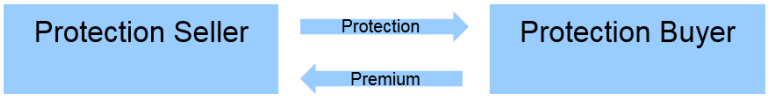
Theory

2.1 Financial aspect

2.1.1 Credit Default Swap

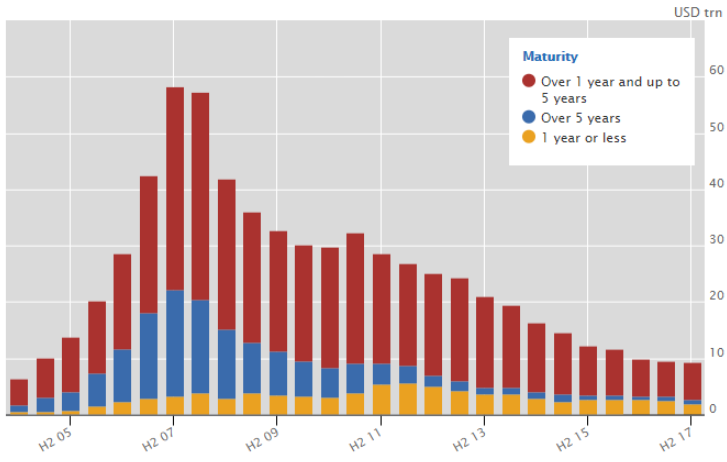
Credit Default Swap is a financial derivative that fundamentally gives purchaser the insurance against default for a series of bond payments. It was first defined in 1997, aiming increase in security for financial institutions [14]. This is a contract between purchaser and the provider where the provider upholds the payments if the underlying bond provider fails [32]. CDS serves as a tool to transfer the risk that a certain counterparty does not meet its obligations from the protection buyer to the protection seller [31]. The contract only covers the notional amount. Thus, it isolates only the counterparty default risk. In other words, it does not protect against others risks such as interest rate risk, FX risk etc. An investor might buy CDS for hedging counterparty risk. The buyer/seller relation can be seen at the Figure 2.1

Figure 2.1: The mechanism of a CDS deal



The term "spread" for a CDS is the annual fee that is paid by the protection buyer as expressed as the percentage of the notional. For example, if the CDS spread of Apple is 130 basis points, or 1.3% (1 basis point = 0.01%), then buying the protection for 10 millions dollars, costs 130,000 dollars annual fee. By its construction, CDS is a standardized product. The payment dates and other specifications such as payment conditions are generally the same for different reference entities. This makes it possible to buy or to sell CDS regardless of whether the investor holds any bonds that the CDS covers. Thus, an investor can trade on this product for speculating on the credit worthiness of the reference entity. Different CDS contracts may have different maturity periods. Commonly, CDS have one to ten years of maturity. The most liquid CDS product traded in the market is five year maturity CDS. The market for CDS contracts especially boosted in the 2008 crisis as hedging held tremendous importance in crisis conditions. Moreover in the crisis conditions, it is relatively easier to speculate on the credit worthiness of counterparties. The Figure 2.2 shows the change in the market size and distribution of the CDS products in terms of tenors.

Figure 2.2: CDS Market Size and Maturity Distribution over years. Figure from [11]



CDS introduces a number of advantages to the markets. First, with this product it is much easier to manage credit risk. The ability of hedging makes it easier to calculate capital requirement for financial institutions. Secondly, CDS contract lowers the barriers of entry, to buy or to sell bonds issued by various entities. This creates a deeper bond market.

Since the focus of this work is more on Machine Learning aspect, the further details of Credit Default Swaps and its effects on market was kept rather short. For further reference please see [31] and [36].

2.1.2 CVA(Credit Value Adjustment)

Financial institutions are required to calculate their counterparty risks. Credit Value Adjustment is one of the most common calculation methods of riskiness. In this section, this calculation is explained shortly. Understandign the concept of CVA is important because CDS spreads are used to extract default probabilities. Even though, we didn't directly calculated CVA using our predictions, we have included this explanation for the reader to understand the use of CDS spreads. For further reference you can apply to [30], [35] and [27].

A financial institution's counterparty exposure has 3 major steps [30]. In the first step, **Scenario Generation**, the random stochastic processes in the economy such as interest rates, FX rates are modelled and simulated a certain amount of time, i.e., 5000 scenarios. These are called risk factors. In the second step, **Instrument Valuation**, the value of the products that were sold/bought are calculated for today and for the future until the end of the maturity of the deals, i.e., 2050. It is possible to value them in the future because the risk factors that these trades are based on, were already modelled in the first step and expected values of them are projected in the future as well. In the third step, **Portfolio Aggregation**, all the projected valuations of the trades per each counterparty added. In other words the portfolios that belong to the counterparties that the financial institution has aggreements with, are aggregated. When we are done with all these steps we get an exposure profile for a product such as interest rate swap as shown in Figure 2.3. We assume that there is no wrong way risk or right way risk. In other words, there is an independence between exposure and counterparty's credit quality. Then the calculation becomes as stated in [39].

$$CVA = (1 - R) \int_0^T EE(t) dPD(0, t) \tag{2.1}$$

where EE is the expected exposure profile shown in Figure 2.3 and R is a deterministic percent ratio called recovery rate, (i.e. 0.4). PD is the probability defaults that are shown in 2.4 that are constructed by the CDS spreads

Figure 2.3: After simulating the risk factors, the valuations of the trades are summed up and we get different valuation paths for different scenarios of the risk factors. Finally we extract the expected values of the simulated valuations to reduce them to a one curve. *Source:* [27]

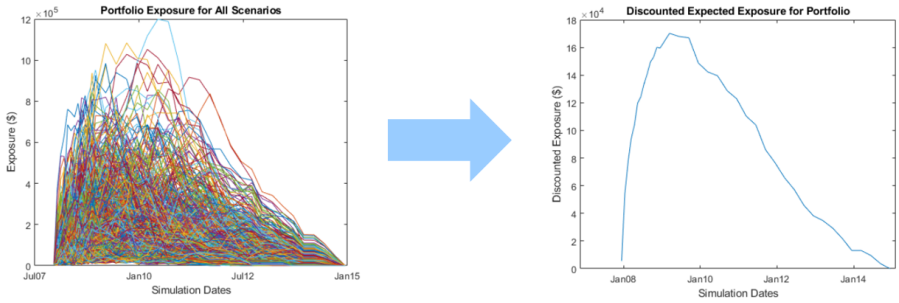
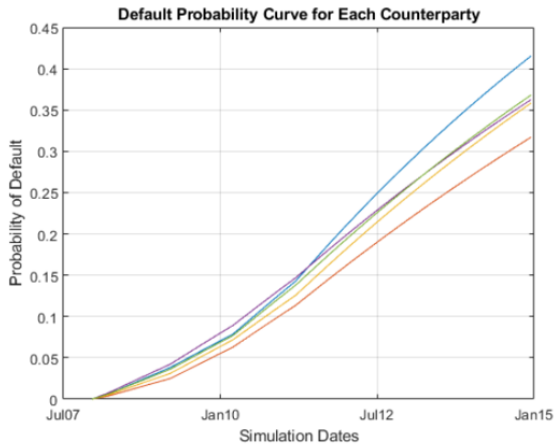


Figure 2.4: Example probability of defaults of some counterparties are shown *Source*: [27]. These are extracted by using CDS bootstrapping method [35], p. 22,23

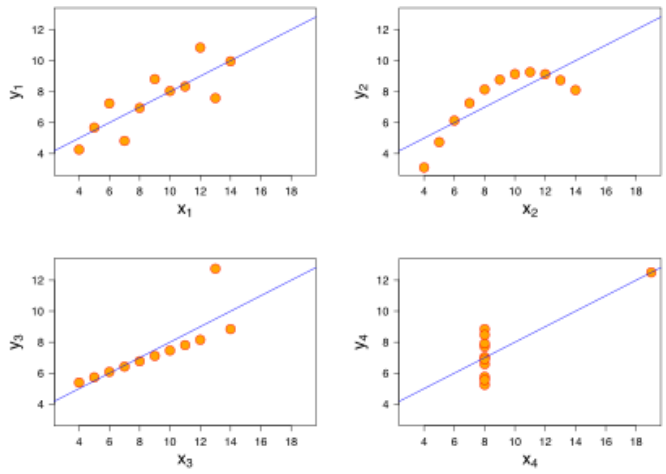


In summary, CVA is the portfolio value which incorporates the probability defaults of the counterparties. In other words, it does not treat them as they are immune to defaulting. For being able to calculate CVA, it is important to have market implied probability of default, thus it is important to have CDS spreads.

2.2 Data exploration

Data exploration and visualization is an indispensable process in Machine Learning applications. This is because human mind is specialized to recognize and perceive information through visual content. Moreover, only having statistical analysis might often be deceiving. This is best represented in Anscombe's quartet [5] where 4 different data sets with exact same mean and variance are visualized as shown at Figure 2.5 .

Figure 2.5: Ancombe's Quartets



For avoiding such a misperception, throughout the project, an emphasis was put on data exploration and visualization. For exploration purposes especially, Principal Component Analysis and Linear Discriminant Analysis was used frequently in this work. Thus, in the following sections, PCA and LDA is described.

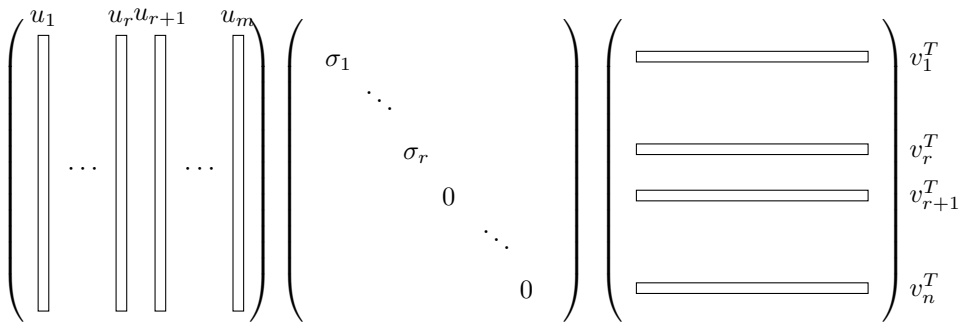
2.2.1 Principal Component Analysis

Principle component analysis(PCA) is one of the most common dimensionality reduction methods. It was invented in 1901 by Karl Pearson [12]. It transforms the data and reduces the number of dimensions while minimizing the information loss. The main intuition is to take correlated variables and orthogonally transform them to a new coordinate system that is represented by principal

components. In the new coordinate system the first principal component covers the most variance in the data while cumulatively all principal components covers all the variance.

In order find exclusively orthogonal principal components, eigen vectors and eigen values of the covariance matrix have to be computed. Thus the problem could be reduced to a matrix factorization problem by using Singular Value Decomposition (SVD).

SVD is a matrix factorization method where the matrices are formed the following way.



$$X = U\Sigma V^T \tag{2.2}$$

Here U and V aro orthogonal matrices as such $V^TV = I$ and $U^TU = I$.

Since the main objective is to find eigen vectors and eigen values of the covariance matrix,it is possible to write SVD as,

$$C = \frac{1}{n}XX^T \tag{2.3a}$$
$$XX^T = (V\Sigma^T U^T U\Sigma V^T) = V\Sigma^2 V^T \tag{2.3b}$$
$$(V\Sigma^2 V^T)v_i = V\Sigma^2 e_i = V\sigma_i^2 e_i = \sigma_i^2 v_i \tag{2.3c}$$

Here v_i is the eigen vector of C.

For calculating PCA, after normalizing the data by substracting the mean and dividing with the standard deviation, singular value decomposition is calculated. The principal component vectors are then given by the V matrix.

Since in the new space, all of the principle vectors are orthogonal to each other, PCA is especially useful for correlated data. For example it can be handy against multicollinearity assumptions of linear regression.

2.2.2 Discriminant Analysis

Discriminant analysis is a data analysis method which was invented by by R.A. Fisher [10] at 1936. It is a multivariate method for solving classification problem. Regression analysis and discriminant analysis has similarities in a way that they both fit a curve/plane/hyperplane to a dataset but differs in the sense that discriminant analysis maps the data onto a categorical variable instead of a continous one. These curves are called discriminant functions. Their purpose is to draw the boundaries among the classes so that the new data points can be assigned to these classes. In this sense logistic regression and discriminant analysis have the same purpose because Logit function is also used for fitting boundary surface. One difference is that LDA assumes the density function of the classes as normal. Therefore, we can fit a normal curve to the data so that we will get a probability density function and find the one which gives us the maximum at a given data point. Besides, LDA is a form of linear transformation for dimensionality reduction that maximizes class separability. This method can also be used for classification.

Discriminant analysis is also similar to Gaussian Mixture Models(GMM). In both of them we fit a probability density function. However, GMM does this without knowing the class labels. It starts by constructing a certain number of distributions on top of each other and optimizes the distribution moments such as μ, Σ and the weights of the clusters. The only input is how many different distributions will be fitted. Then it finds the optimal means, covariances and the ratios of the distributions. On the other hand, in LDA the situation is a bit different than GMM because the relevant weights of the reconstructed common distributions come from the *prior* probabilities. According to the Bayes theorem:

$$\Pr(G = k|X = x) = \frac{f_k(x)\pi_k}{\sum_{l=1}^{K} f_l(x)\pi_l} \tag{2.4}$$

Assume we model each class density the **same** co-variance as following,

$$f_k(x) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(x-\mu_k^T)\Sigma_k^{-1}(x-\mu_k)} \tag{2.5}$$

In order to decide which class does our observation belongs, we need to derive all the probabilities and compare them. $\frac{Pr(G=k)|Pr(X=x)}{Pr(G=l)|Pr(X=x)}$ is greater or smaller than 1. The points where it is exactly 1 is our decision boundary between these two classes. For simplicity we write it in log form as stated in [15, p. 108],

$$\log \frac{Pr(G=k)|Pr(X=x)}{Pr(G=l)|Pr(X=x)} = \log \frac{f_k(x)}{f_l(x)} + log \frac{\pi_k}{\pi_l} \tag{2.6}$$

After expanding the density functions we get,

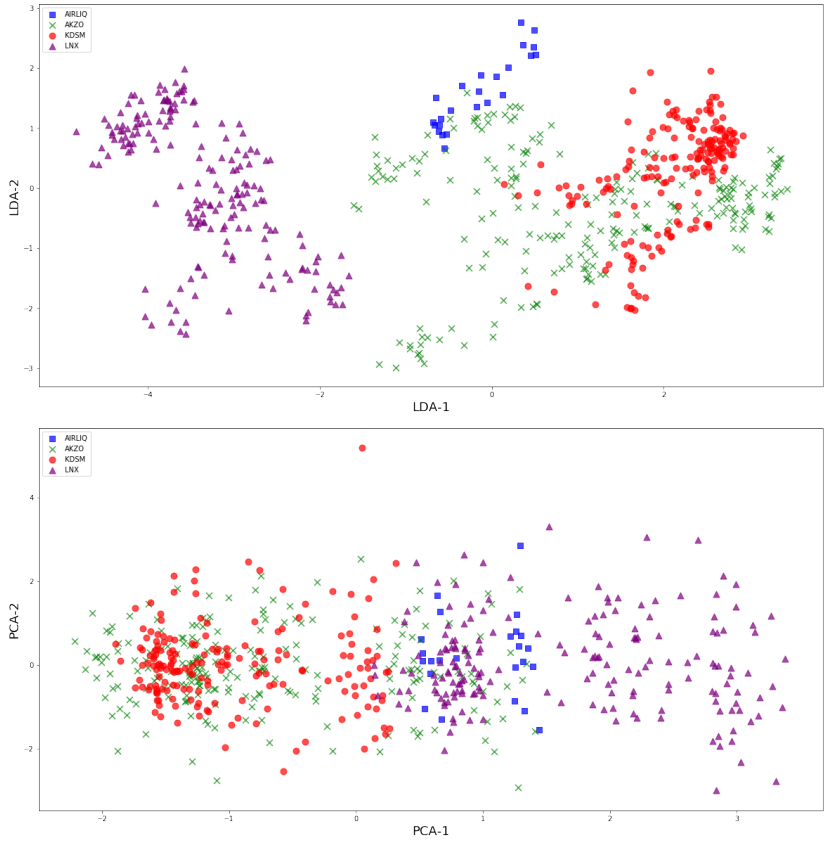
$$= \log \frac{\pi_k}{\pi_l} - \frac{1}{2}(\mu_k + \mu_l)^T \Sigma^{-1}(\mu_k - \mu_l) + x^T \Sigma^{-1}(\mu_k - \mu_l) \tag{2.7}$$

At this point we can generalize all the decision boundaries as the maximum of the following function,

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log \pi_k \tag{2.8}$$

At the Figure 2.6, a visualization describing the projection of some of our data set using LDA and PCA can be seen.

Figure 2.6: An example illustration of the dimensionally reduced data for the same bucket dataset (Basic Materials, Europe, A). Using FS2 feature selection.(*Source*: our own)



2.3 Model evaluation

In this section we introduce the theory for different evaluation methods that we have used for the comparisons of the different algorithms.

2.3.1 Evaluation metrics

There are different evaluation metrics related to classification and regression problems. For the classification models in this work, the accuracy was reported. It is the ratio of the correct predictions to the number of all data points.

$$\text{accuracy} = \frac{\text{number of correct prediction}}{\text{total number of test}}$$

For regression models, the truth and the predictions are continuous variables. For comparing real number predicted values and true values, it is needed to find an evaluation method that aggregates the diffence between prediction and truth. In this work, for the time series comparison evaluations, mean squared error was used. Mean squared error is the average of squares of differences between estimations and true values.

$$MSE = \frac{1}{N}\Sigma_{j=1}^{n}(y_j - \hat{y})$$

2.3.2 Cross Validation

Cross validation is a method commonly used for evaluating how good a model generalizes the data space. For k-fold cross validation, the data is randomly split into k equal sized parts. One of the parts is used for measure validation metric while last four is used for training. We repeat the same process for each of the k folds and average all the measurements. Note that k is a variable. In this work we have used 5 fold cross validation.

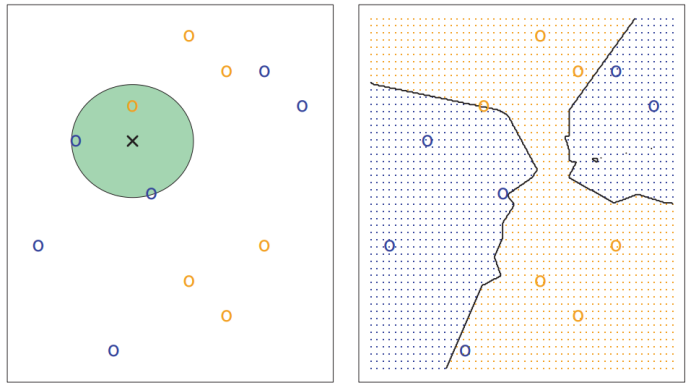
2.4 Classification/Approximation models

2.4.1 k-Nearest Neighbors

k-Nearest Neighbor method is classification method that find the closest k different observations to the test data points and assigns the most common label between the neighbor points.

This is a very simple model which does not require many parameters except k [4]. Also it is one of the lazy learning models where all the computation is done on the prediction time. Figure below shows the idea of KNN. On the left, a case classification is seen where k is equal to 3. Since the majority of the neighbors are blue, the test point(cross) will be assigned to blue class. On the right, the boundaries representing different classes can be seen.

Figure 2.7: An Illustrative Example of k-NN algorithm,[18, p. 40]



2.4.2 Decision Tree

A decision tree is a model that explains a data set with different partitions [23]. It is constructed by nodes which form a directed tree. In a decision tree, nodes split data space into sub spaces.

In the tree, leaves represent a single class. Test data is classified by navigating from root to one the leaves according to the property of the input data.

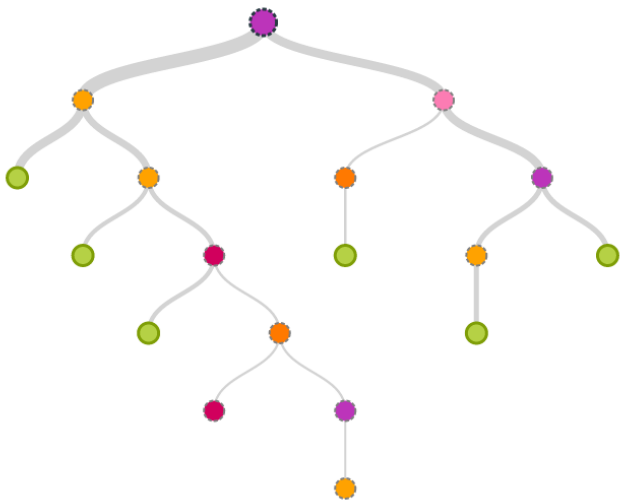
Figure 2.8, illustrates an example decision tree where each node represents a question that navigates input data to one of the leaves.

Main properties of decision trees are as follows,

- Decision trees are easily interpretable since they are visually close to human thinking
- Decision trees are robust to noisy data as they perform a type of feature selection
- Decision trees are generally robust to outliers

  Also please not that decision trees are the fundamentals of different Machine Learning methods such as Random Forest and different boosting methods.

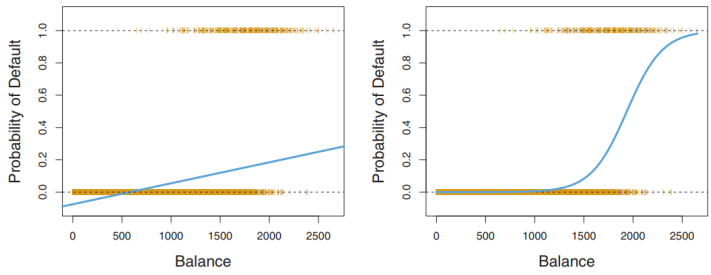
Figure 2.8: An Illustrative Example of decision tree algorithm



2.4.3 Logistic Regression

Rather than modelling the response variable directly, we can model the probability that the dependent variable belongs. Like discriminant analysis, logistic regression can also be used to determine the categorical probability of an event given some either continuous of categorical variables as input. In Figure 2.9 linear and logistic regression are compared. It is seen that linear model is not capable to explain the probabilities correctly. Values out of 0 and 1 are observed. Therefore, logistic regression is a better choice for modelling the probabilities.

Figure 2.9: Left: *Linear regression is fitted to the categorical data. It can be observed that we cannot impose the model to have positive probability as well as probability greater than 1.* Right: *Logistic function is fitted to the categorical data. This time all the probabilities are stuck between 0 and 1 due the nature of the logistic function.* [18]



In order to avoid the problem that arises from linear regression, *logistic function* is fitted.

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}} \tag{2.9}$$

Multiple Logistic Regression

Often the classification problems are not binary as it is stated in equation 2.9. Especially in this project we want to classify the feature vectors with multiple classes. Therefore, we can generalize the equation in 2.9 as:

$$p(X) = \frac{e^{\beta_0+\beta_1 X_1 \ldots \beta_p X_p}}{1 + e^{\beta_0+\beta_1 X_1 \ldots \beta_p X_p}} \tag{2.10}$$

Optimization

Usually, in order to fit the model in equation 2.9, *maximum likelihood* method is used to calibrate the parameters. The intuition behind maximum likelihood is to maximize sum of the probabilities that are obtained from the model. For example in equation 2.9 β_0 and β_1 are chosen in a way that they will yield the

maximum probability sum. We define the sum of probabilities of our all fitted data points as:

$$l(\theta) = \sum_{i=1}^{N} log(p_i)(x_i;\theta) \tag{2.11}$$

For simplicity we assume that there are only two classes. Then the class probabilities of an input vector will be $p_1(x;\theta) = p(x;\theta)$, and $p_2(x;\theta) = 1 - p(x;\theta)$. Without loss of generality θ in equation 2.11 can be written as $[\beta_1\beta_2]$. Then the log likelihood becomes,

$$\begin{aligned} l(\beta) = \sum_{i=1}^{N} y_i \log p(x_i;\beta) + (1-y_i)\log(1-p(x_i;\beta)) \\ = y_i\beta^T x_i - \log(1+e^{\beta^T x_i}) \end{aligned} \tag{2.12}$$

To maximize the function we set its derivatives to zero:

$$\frac{\partial l(\beta)}{\partial \beta} = \sum_{i=1}^{N} x_i(y_i - p(x_i;\beta)) = 0 \tag{2.13}$$

Now the problem turned into a root finding issue. In order to find the β values that satisfy the equation 2.13, we use Newton-Raphson algorithm [15, p. 120]. Newton-Raphson necessitates the second order partial derivative of the function around a point. This is also called as Hessian matrix.

$$\frac{\partial^2 l(\beta)}{\partial \beta \partial \beta^T} = \sum_{i=1}^{N} x_i x_i^T p(x_i;\beta)(1-p(x_i;\beta)) \tag{2.14}$$

Then we apply the algortihm $x_{n+1} = x_n - \alpha H[f(x_n)]^{-1}\nabla f(x_n)$ as:

$$\beta^{new} = \beta^{old} - \alpha(\frac{\partial^2 l(\beta)}{\partial \beta \partial \beta^T})^{-1}\frac{\partial l(\beta)}{\partial \beta} \tag{2.15}$$

This is a very similar approach with gradient descent explained in equation 2.24. However, Newton's method approaches to the minima much faster. This is because it contains the second order derivatives as well. Therefore, it has less "zigzag" path than gradient descent as .

2.5 Regression models

2.5.1 Linear Regression

Given a matrix of inputs $X = (x_1, x_2, x_3...x_n)$, following structure is written

$$\hat{y} = X\theta + \epsilon \tag{2.16}$$

where $\theta = (\theta_1, ..., \theta_n)$ are the unknown parameters. ϵ is a random variable with mean $E[\epsilon] = 0$.

Based on the observations we want an estimate of real θ such that $f(X, \hat{\theta})$ describes the observations in terms of the closeness. In this project sum of squared errors (SSE) $S(\theta) = \sum[y - f(X, \hat{\theta}]^2$ is chosen.

$$S(\theta) = (y - X\theta)^T(y - X\theta)^T \tag{2.17}$$

$$\nabla_\theta S(\theta) = -2X^T(y - X\theta) = 0 \tag{2.18}$$

$$\theta = (X^T X)^{-1} X^T y \tag{2.19}$$

Assumptions

1. **Multicollinearity** is observed if the feature matrix does not have a full rank. In other words, it has a perfect correlation between two or more columns. It can be detected if there is either, too high R^2, too high pairwise correlation between the features, low significance level in parameters. Independent variables does not have to have zero correlation. However, correlation increases the uncertainty in our prediction.

2. **Heteroscedasticity** The data points to be explained, ought to be distributed homogeneously throughout the axes. In other words, the variance of the data points should bot be changing a lot along an axis,

3. **Auto correlation of residuals** is the correlation of the residuals of the fitted model with their own historical values. When there is an auto correlation among the errors, then the errors are not white noise. Therefore,

there must be a signal which was not captured with the current model. This is a good indicator for redesigning the model by either adding or subtracting some features. If the variables are perfectly auto correlated then integrating the them can remove the auto correlation.

2.5.2 Neural Networks

Artificial neural networks are another Machine Learning method which is inspired by brain biology. Similar to the nerve cells, ANN has neural units(neurons) which are activated at a certain signal threshold. In other words the main idea of the artificial neural networks is to extract linear combinations of the inputs as features and mapping them with a non linear set of transformations to meet with the desired target.

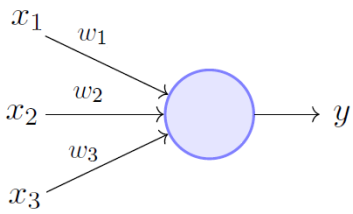
2.5.2.1 Neuron

The artificial neuron is the processing unit of a neural network. A neuron consists of inputs, weights, a bias and an activation function that are all used to calculate an output. The output value is calculated by multiplying the weights with the inputs and summing them.

$$y = \begin{cases} 1 & if \sum_k w_k x_k \\ 0 & otherwise \end{cases} \tag{2.20}$$

This process can be visualized as in figure 2.10.

Figure 2.10: A perceptron *Source:* [25]



2.5.2.2 Forward Propagation

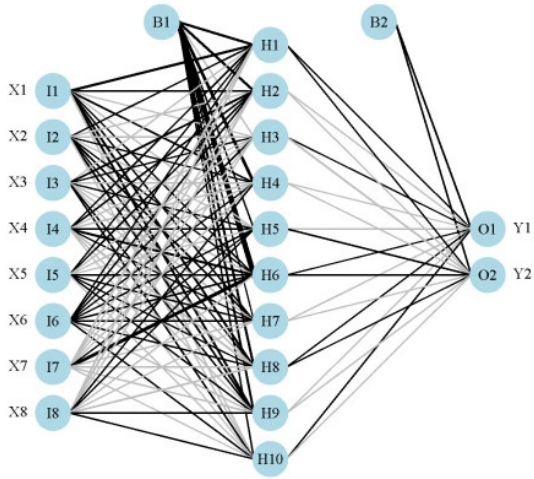
Assume we collect the first and second weight layers of the network shown in Figure 2.11 as,

$$W^{(1)} = [\mathbf{w}_1^1 \mathbf{w}_2^1 ... \mathbf{w}_H^1] \text{ and } W^{(2)} = [\mathbf{w}_1^2 \mathbf{w}_2^2 ... \mathbf{w}_H^2] \tag{2.21}$$

Then by using one of the activation functions in figure 2.12, we get the output of the network as,

$$Y = h^{(2)}\left(\sum_{j=1}^{H} W_{kj}^2 h^{(1)}\left(X^T w_j^{(1)}\right)\right) \tag{2.22}$$

Figure 2.11: Illustration of a simple fully connected neural network *Source:* [26]



2.5.2.3 Activation functions

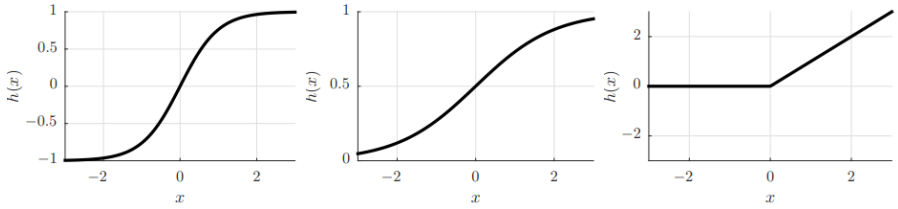
Logistic sigmoid is one of the most popular activation functions. It is an S shaped activation function. Therefore, it is great for classification problems since it can represent probabilities. It is in the range between 0 and 1 but at the same time smooth and increasing non linear form.

Tangent hyperbolic(Tanh) is an activation function that looks much like the sigmoid function.

Rectified Linear Unit (ReLU) is another popular activation function which avoids the vanishing gradient problem of sigmoid and tangent hyperbolic. In our project this is very crucial because recurrent neural networks have this problem as wel.

The activation function in equation 2.22 can be chosen as $h(x) = tanh(x)$, logistic sigmoid $h(x) = (1 + e^{-x})^{-1}$ or rectified linear unit $h(x) = 0$ if $x < 0$ and otherwise $h(x) = x$.

Figure 2.12: Hyperbolic tangent, logistic sigmoid, rectified linear unit, respectively are shown[37].



2.5.2.4 Loss function

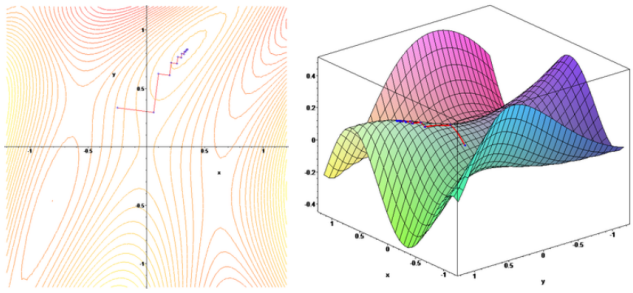
A loss function, also known as a cost function is a measure of how good the model is fitted to the target data. Even in unsupervised learning in deep learning, a cost function is necessary to successfully find the way to go in the parameter space. Therefore, it is a significant hyper parameter. One of the loss functions in this project used for regression in neural network is means squared error.

$$L(W^1, W^2) = \frac{1}{N} \sum_{i=1}^{N} (Y_i - \hat{Y}_i)^2 \tag{2.23}$$

2.5.2.5 Optimization

Gradient Descent The main goal of our neural network is to find the right weight combinations to meet with our target values. Learning takes place by adjusting the network's weights while finding a minimum of the loss function.

Figure 2.13: Gradient descent algorithm illustration for the function $F(x, y) = (\sin\frac{1}{2}x^2 - \frac{y^2}{4} + 3)\cos(2x + 1 - exp^{-y})$ *Source:* Wikipedia/Gradient Descent



Most of the time this is done by the gradient descent algorthm even though there are other tehcniques such as Newton, Gauss-Newton.

$$W^1 := W^1 - \alpha \frac{\partial L(W^1)}{\partial W^1} \tag{2.24}$$

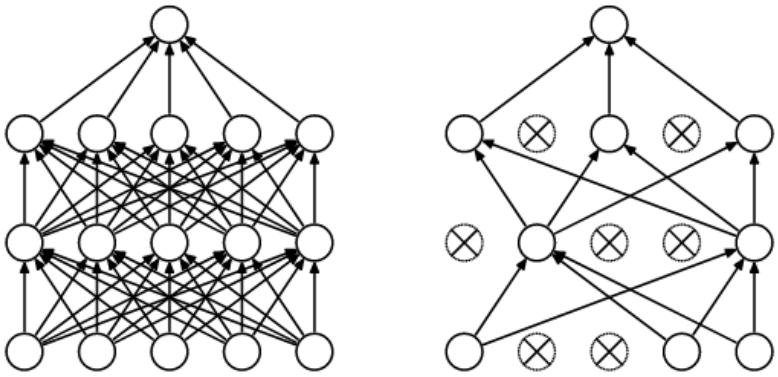
In the (2.24) we see that the new weight is adjusted by a learning rate denoted as α. It is adjusted in the steepest direction in the parameter space. Finally we expect it to converge slowly to the optimal point the weights are not stuck in a local minima.

2.5.2.6 Regularization

Very important problem in machine learning is that even though the algorithms seems to perform well, they might be actually perform significantly worse in an independent data set. This is not only a problem of deep neural networks but it also happens when using many other machine learning algorithms as well. In conventional statistical machine learning algorithms this problem can be overcome with ensembling methods. In neural network context, there many different regularization methods. Currently, Deep Learning regularization is a popular research topic. In this work we have mainly used dropout layers for regularization.

Dropout method was first described by [16]. This method randomly closes learning flow to some of the neurons in the network. It is in a sense choosing a subset of the network for training at for every training batch. This way every neuron is expected to learn signal without totally depending on another specific neuron.

Figure 2.14: A standard neural network (Left), Application of dropout to the network(Right). Crossed units have been dropped.(*Source:* [34])



(a) Standard Neural Net (b) After applying dropout.

2.5.3 Feed Forward Neural Networks (FFN)

FFN is the most straightforward use of Neural Networks. It built by input layer x, hidden layers h_t and output layer y. Defining equations for the architecture are as below.

$$h_t^j = f(W^{(hh)} h_{t-1} + W^{(hx)} x_t) \tag{2.25}$$

$$\hat{y}_t^k = g(W(y) h_t) \tag{2.26}$$

Where $f(z)$ is the activation function, for example a *sigmoid* function:

$$f(z) = \frac{1}{1 + \exp^{-z}} \tag{2.27}$$

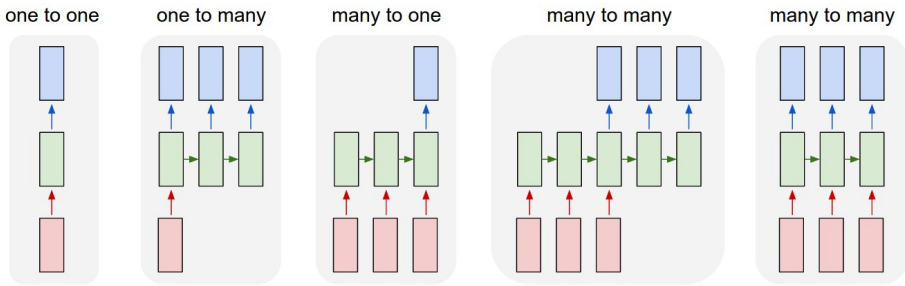
and $g(z)$ is the *softmax* function:

$$g(z_m) = \frac{\exp z_m}{\sum_k \exp z_k} \tag{2.28}$$

2.5.4 Recurrent Neural Networks

Recurrent neural networks (RNNs) are a sub architecture of Neural Networks which has the capability to model the long term dependencies in a sequential data. It is the neural network equivalance of autoregressive models like ARMA, ARIMA, GARCH etc. Research shows that neural networks outperforms these conventional econometric models [20].

Figure 2.15: A schematic overview of a simple recurrent neural network, (*Source:* [19])



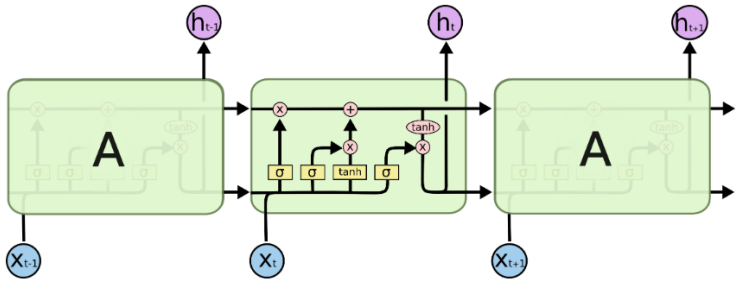
2.5.5 Long Short Term Memory Neural Networks

This section is about a sub architecture of Recurrent Neural Networks which is known as Long Short Term Memory Neural Networks(LSTM). It was mostly inspired by a project paper we have written for the Deep Learning course in Technical University of Denmark where we focused on language modelling and generating word sequences. In RNN when the backpropagation through time is high, the weights between these layers are multiplied by each other and eventually become zero. In other words, α^T goes to zero when T is too big $-1 < \alpha < 1$. Therefore the weights are updated but with a very small change. This means that the network is not learning. In order to solve this problem, the Long-Short Term Memory (LSTM) network architecture was introduced in 1997 by Hochreiter [17] and extends the idea of an RNN by using special memory cells as a way to include updates over large time lags. These memory cells store the long term dependencies within the sequence and come in two forms, a *new memory cell* $\tilde{c}_t$ and a *final memory cell* c_t. Additionally, there are 3 vectors that influence how much information of each of these cells flows from one state to another, the *input gate* i_t, *forget gate* f_t and the *output gate* o_t. The mathematical formulation of

the LSTM, taken from [28], are as follows:

$$i_t = \sigma(W^i x_t + U^i h_{t-1}) \quad \text{Input Gate} \tag{2.29}$$
$$f_t = \sigma(W^f x_t + U^f h_{t-1}) \quad \text{Forget Gate} \tag{2.30}$$
$$o_t = \sigma(W^o x_t + U^o h_{t-1}) \quad \text{Output Gate} \tag{2.31}$$
$$\tilde{c_t} = tanh(W^c x_t + U^c h_{t-1}) \quad \text{New Memory Cell} \tag{2.32}$$
$$c_t = i_t \circ \tilde{c_t} + f_t \circ \tilde{c}_{t-1} \quad \text{Final Memory Cell} \tag{2.33}$$
$$h_t = o_t \circ tanh(c_t) \quad \text{Output} \tag{2.34}$$

Figure 2.16: Visual illustration of a LSTM cell. *Source:* [29]



Where σ is the sigmoid function (see equation 2.27), [W^i, W^f, W^o, W^c, U^i, U^f, U^o, U^c] are weight matrices that are learned by the model, and $\circ$ is the element-wise multiplication of two matrices.

The gates i_t, f_t and o_t all use a sigmoid 2.27 function to get to a number of 0 and 1, which at any given time t determines how much of the input or previous states should be incorporated in the next state. The input gate does this the information from the *new memory cell* $\tilde{c}_t$ that is created from the input at t and the previous state h_{t-1} to the *final memory* c_t. The forget gate does this for the previous step $\tilde{c}_{t-1}$, and the output gate blocks or unblocks the flow of information the final memory $c(t)$ to the output $h(t)$.

2.5.6 Convolutional Neural Networks (CNN)

Convolutional Neural Networks are Deep Learning architectures which includes convolutional layers and pooling layers. Currently, they were especially found to be performing good on data which includes spacial patterns such as images.

Currently there are many CNN architectures successful applied on image, video and speech processing based tasks [21].

2.5.6.1 Convolutional Layer

The name convolution comes from the convolution operation in CNN. For a 2-d input X, forward propagation of a convolutional layer with K kernels is as follows.

$$H^{(k)} = f(W^{(k)} * X + b^{(k)}$$

Where k = 1,...,K; f is activation function and '*' is convolution operation. The operation is defined as,

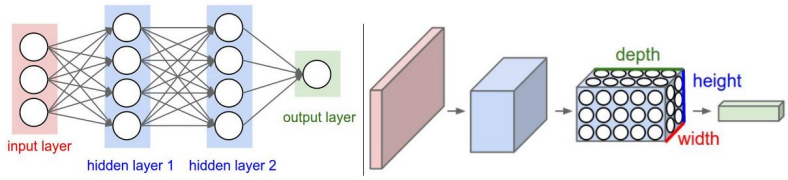
$$W^{(k)} * X = \Sigma_{m=1}^{|W_{:,j}^k|} \Sigma_{n=1}^{|W_{i,:}^k|}$$

The output is by doing concatanation on all K convolution to form a 3D tensor.

$$conv2d(X, W) = H^{(1} \oplus H^{(2)} \oplus ....H^{(K)}$$

Here, $W^(k)$ is the filter. Every filter in the vector learns different spacial patterns. Difference between Fully Connected layer and Convolutional layer can be seen in the next figure.

Figure 2.17: Left: A Fully Connected Neural Network. Right: A Convolutional Network which transforms 3d input(spacial information 2 dimension, value 3rd dimension) to 3d output. *Source:* [38]

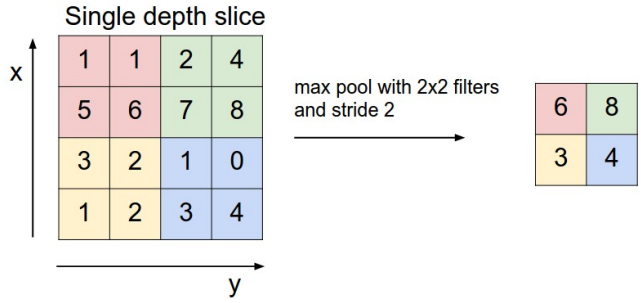


2.5.6.2 Pooling Layer

Since the output of Convolotional layer generally holds redundant information, generally pooling layers are added after these layers for a sub-sampling process. Currently max pooling is the most common sampling method [21].

Maxpooling layer reduces the the data load by down sampling . This is also a regularization type since it prevents overfitting (decreases number of weights). An example max pooling application is shown in the next figure.

Figure 2.18: Max pooling operation *Source:* [38]



Chapter 3

Related work

In this section we describe conventional models commonly used for CDS proxying as well as new attempts that aims to leverage Machine Learning.

3.1 Conventional models

3.1.1 Intersection Method

European Banking Authority proposed a methodology for CDS proxy problem. As explained at *[6]*[1] the methodology is to average CDS spreads in the same rating, region and sector at the observation day. This methodology is called Intersection or Bucket method. In this method, the proxy CDS of an unobservable entity is the average of the all liquid CDS spreads in the same bucket. The proxy spread is defined as :

$$S_i{}^{\text{proxy}} = \frac{1}{N} \sum_{j=1}^{N} S_i \tag{3.1}$$

[1]Direct link: You can reach to the document from this **link**

where $N > 1$ is the number of entities in a bucket and S_j is the spread of the entity j.

For understanding the bucket concept, we have grouped the entities in our data set similar to the ways done in [9, p.3–4]. On 2018-11-30, the buckets look as in the table at the appendix **??**.

3.1.2 Cross-section Method

In [9], it was suggested that the spreads can be explained by the multi-dimensional regression across rating, region and industry sector. Since the regression is run on the logarithm of the CDS spread; coefficients representing sector, region and rating fields could be shown as multiplicative components. The main equations for cross-section algorithm are shown below.

$$\begin{aligned} S_i^{proxy} &= M_{glob} \times M_{sector} \times M_{region} \times M_{rating} \\ log(S_i^{proxy}) &= \Sigma_n^{j=1} A_{ij} x_j \end{aligned} \tag{3.2}$$

As explained in [9], and as confirmed by our experiments, it was observed that Cross-section method outperforms intersection in terms of robustness and error performance.

3.2 Machine Learning models

CDS spread proxy problem is not among the most popular Machine Learning challenges since it is about a very specific problem in finance. There are still some attempts that should be mentioned.

In [7], different machine learning approaches are utilized in a comparative manner as it is done in this project. However, their approach is to focus on the different classifiers rather than including regression analysis as well. They trained and tested their classifiers in each bucket. Some of the regression approaches that are in use in the industry which was suggested in [9], were criticized by them even though their performances are quite good in terms of curve similarity measures. The main argument to this was not complying with all the regulator's criteria. Among the classifiers that they tried, there are Linear Discriminant

Analysis(LDA), Naive Bayes, k-NN, Logistic Regression, Decision trees, Support Vector Machines, Neural Networks, Bagged Decision Trees. It is concluded that the best classifiers are Neural Network, Support Vector Machine, Ensemble/Bagged Trees. The results were shown with the accuracy score only by excluding time series evaluation metrics.

Finally on the other finance fields, Machine Learning algorithms are widely exploited. For example, many applications of Machine Learning models are being heavily used in Portfolio Management [22], Algorithmic Trading [13] and Fraud Detection [8]. Moreover, in recent work, there are attempts to bring different fields for financial forecasting. For example, some new research was done on forecasting stock prices using financial news [24]. However, to our knowledge, currently, only finance research was done on Credit Derivatives and our work is one of the first attempts to use Machine Learning on this field.

Chapter 4

Data

One of the major drawbacks to use Machine Learning for solving problems related to finance is the lack of access to sufficiently large data sets. Companies owning the data sets generally are not willing to share financial information because of commercial motives and privacy issues. This is the reason why in this work, mostly easily accessible data about a counterparty was prioritized.

As explained in the introduction, Credit Default Swap is a product that gives some clues about the default probability of a counterparty. Thus, the data should always be of a nature that holds information about default risk of a counterparty.

Basel III [1] articulates that, if a given counterpary has not liquid CDS spread, financial institutions are required to use a proxy spread using a model that takes the rating, region and sector of the counterparty into account. As explained at the related works section, currently used common methods such as Intersection method and Cross-Section method was built on this requirement. There sector, region and rating properties are the natural, minimum information that should be collected about a counterparty.

As the first step of our project, daily data about the features mentioned above and CDS spreads on different tenors were collected using Nordea's internal data sources to create a full dataset. This dataset will be referred as *S1*. It consists

of daily tables including following columns.

- *TICKER:* Ticker/identifier of the counterparty
- *DATE:* The date the data represents.
- *COUNTRY:* The country in which counterparty operates i.e France, Canada.
- *SECTOR:* The sector in which counterparty operates, i.e Finance, Technology.
- *REGION:* The region in which counterparty operates, i.e Europe, N.America
- *RATING:* The credit rating of the counterparty given by the credit rating agencies i.e: AA, A, BBB, BB ...
- *SPREAD6M:* Spread for 6 months credit default swap
- *SPREAD1Y:* Spread for 1 year credit default swap
- *SPREAD2Y/3Y/5Y/7Y/10Y/15Y/20Y/30Y*: 8 columns showing spreads for respectively 2Y, 3Y, 5Y, 7Y, 10Y, 15Y, 2Y, 30Y credit default swaps.

At the Apendix **??**, you can see heads of two tables from 2018-03-21 and 2018-03-04. If the tables are analyzed, it will be noticed that there are missing counterparties in tables. For example the ticker ABHLTD is missing in the first table. This proves the need for data cleaning before the start of the actual work.

Before any cleaning, S1 dataset consists of above mentioned fields between 2008-03-18 and 2018-11-30 for 2772 working days for 3462 counterparties . Also for a better understanding of S1, please refer to **??** to see the distribution of the companies on the sector, region and rating fields as a snapshot from 2018/11/30, the last day where the data is available.

S1 consists of many counterparties, including governments, public companies as well as private companies. Besides the features mentioned above, if the counterparty equity is traded on the open market (if it is a public company), the changes in the equity prices could give some clues about changes in Credit Default Swap spreads. For example, we would expect a negative correlation between the credit default swap spreads and the equity returns when some negative news about the counterparty are spread between people. Because of this reason, stock prices for open companies were collected as well.This data source will be referred as *S2*. This source is open and can be obtained from [3] and [2]. S2 consists of the following fields.

- *TICKER:* Ticker/identifier of the public company (counterparty)
- *DATE:* The date the data represents.
- *CLOSING PRICE:* Stock price for the company at the closing time of the stock market.
- *VOLUME:* Total number of stock units traded in the related date.

At the Apendix **??**, Figure **??** shows the stock prices information for company AmerisourceBergen Corp. (with ticker 'ABC').

In this work, after cleaning process we have merged S1 and S2 data sets to create one golden data source.

4.1 Data cleaning

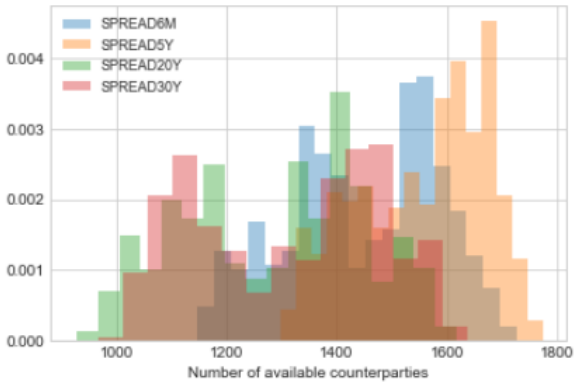
Cleaning the data is an important step for obtaining a healthy working model. From the two datasets used, *S2* was already clean and ready to use for 410 counterparties. However, cleaning was necessary for S1.

The dataset S1 consists of CDS spread of 11 different tenors. However, as shown at Figure 2.2 and stated by European Banking Authority [6], the most liquid CDS product is 5 year maturity CDS products. This is also the reason why the 5 years spread data is much more easy to collect compared to other CDS products covering different periods.

In S1, the number of counterparties of which CDS spread is available changes for different tenors. For example on a given day there might be 1500 counterparties for which we know 5 years CDS spread while 1000 companies for 10 years spread.

Before any cleaning, the distribution of the number of counterparties available for different working days is shown at Figure 4.1

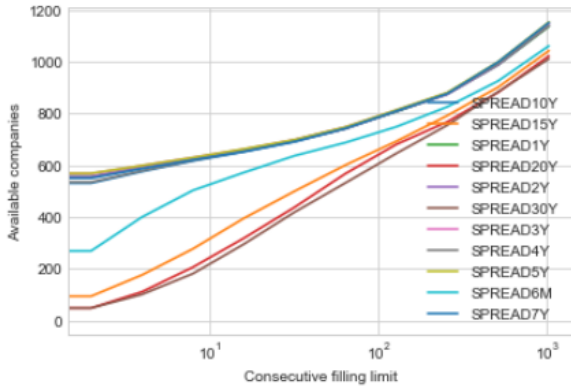
Figure 4.1: Daily spread maturity distribution in raw data



As expected 5 years maturity CDS spreads are most complete. This is expected by the fact that the market is dominated with 5 years CDS. Also, after discussion with field experts, we have learned that most of the tenors other than 5 years maturity are calculated through interpolation methods by using other available tenors. Therefore 5 years maturity products are by far the most interesting to use and to predict.

A simple strategy to clean the data is to interpolate if the number of consecutive days that the data is not available is small. For example, if only one line, one day, is missing, instead of dropping the columns all together it is wise to just interpolate the missing point. However, we should be careful to limit interpolation. For this we have put a limit stating maximum number of consecutive missing points to interpolate. If this number is higher than the limit than the data about the counterparty is dropped all together. Figure 4.2 shows the relation with this limit and available counter parties.

Figure 4.2: Relation between consecutive interpolation limit and number of companies



According to the graph at Figure 4.2, the consecutive missing point interpolation limit was chosen to be 15 working days. This limit provides us 680 counterparties for which CDS spread time series is fully available for the historical period between 2009 and 2018.

Other missing fields such as sector, region and rating are easier to fill. This is because sector and region are nearly constant features for a counterparty. Also rating fields is not fluctuating daily but after credit changes which is not at daily frequency. Therefore it is possible to fill the missing points on these fields by forward filling.

In summary the data was cleaned in the following process.

- All the missing sector, region, country, and rating fields are cleaned by forward filling and backward filling.
- All the tenors except 5 years tenor is dropped.
- All the tickers which has more than 15 consecutive missing lines for 5 years CDS spread were dropped.

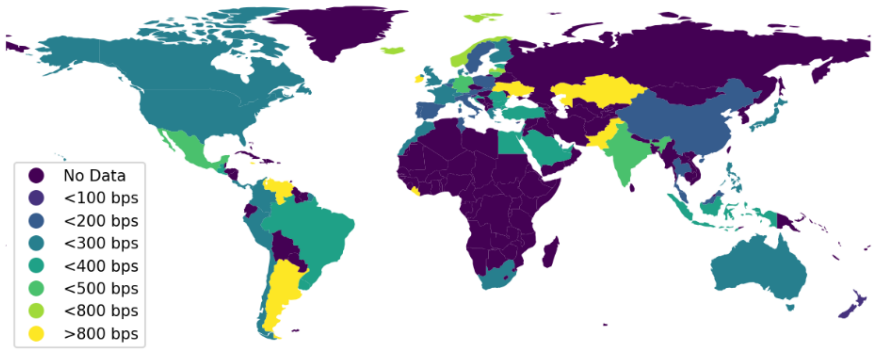
After the cleaning, there exist 680 counterparties of which daily 5 years spread rates are known. From these counterparties, daily stock prices are know for 410 companies.

4.2 Data investigation

By its nature, CDS is a product that heavily depends on the geopolitical location of the counterparty. Thus, in general, it is expected to have relatively lower CDS spread for counterparties operating in the developed countries compared to developing and underdeveloped countries. Of course, a counterparty located in a developed country might still be too risky and have high CDS spread however this is a less frequent case.

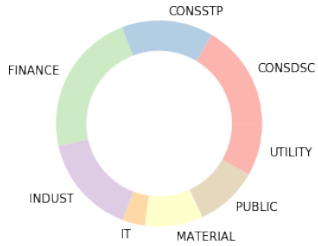
To better see this relation, the World map at the Figure 4.3 was generated. The map shows average CDS spread per country as observed in 2009.

Figure 4.3: Average CDS spread per country 2009

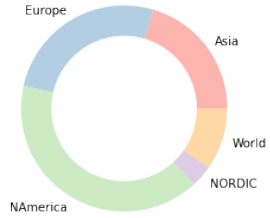


Apart from the geographical location distribution, another interesting distribution of our data set is on sectors.In S1 the sectors and region distribution looks as shown at Figure 4.4 for 680 counterparties.

Figure 4.4: Sector and Region Distribution on S1



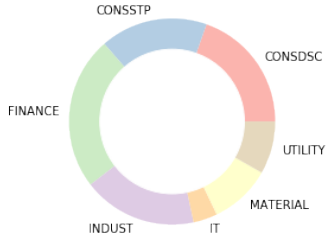
(a) Sector Distribution



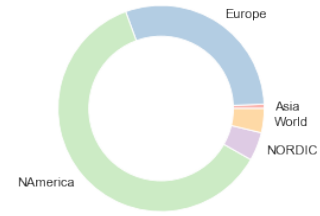
(b) Region Distribution

As the counterparties in S1 include, private companies and government which does not have equities, only 410 of those counterparties has fully available information from both S1 and S2. Thus it is relevant to see the distributions for those 410 companies(counterparties) to see if there is an imbalance in our data.

Figure 4.5: Sector and region distributions on S2



(a) Sector Distribution



(b) Region Distribution

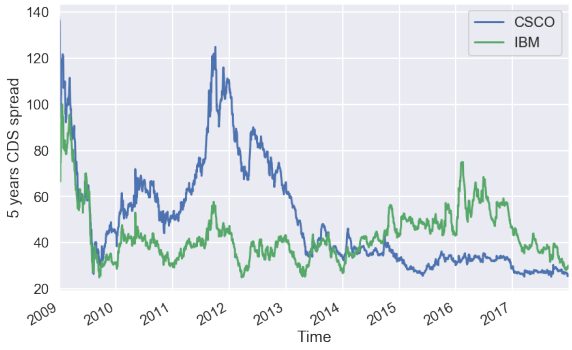
Figure 4.5 shows that there is an imbalance on the region distribution of the companies that both S1 and S2 information is available. This is because it is much easier to find historical stock prices for companies operating in Europe and North America. On the other hand sector distribution is balanced. Please note a we did not take any further cleaning for balancing region data. This is because as it is, we didn't find any clue that shows it created any bias on the models. After the discussion with finance experts, we concluded that the imbalance is not serious.

To understand the data further, it is important see the actual time series. Thus,

in the following, we will explore time series CDS spreads.

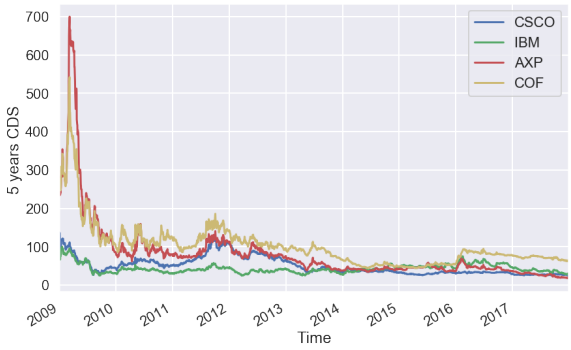
CDS spread for the counterparties that are in the same region, sector and rating bucket are generally highly correlated. At Figure 4.6, 5 years CDS spread time series is seen for 2 different companies from North America, operating in technology sector and had the 'A' rating at 01-01-2009. As seen both companies generally perform in the similar CDS spread bands.

Figure 4.6: CSCO and IBM, 5 year maturity CDS spread between 2008 and 2018



For understanding the effect of the sector, it is possible to compare CSCO and IBM companies with other different that are operating in North America and had 'A' rating at 01-01-2009, but are operating in Finance sector. At Figure 4.7, 5 years CDS spread of 2 companies from technology sector is shown along with 2 companies from finance sector.

Figure 4.7: 5 year maturity CDS spread between 2008 and 2018 for companies from finance and technology sectors

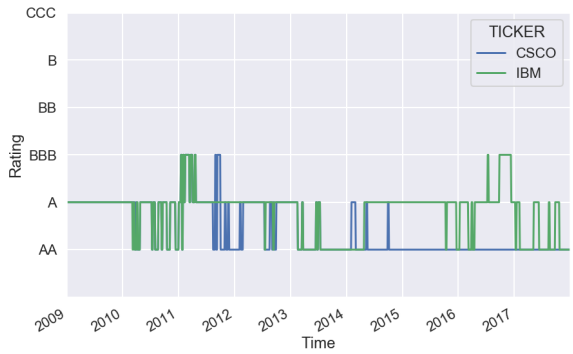


As seen at Figure 4.7, different sectors has its own dynamics. At the 2009 where economical crisis had effects on the market, companies working in finance sector had relatively high CDS spreads while technology companies was not effected as severe. This shows importance of sector in CDS proxying.

Rating is another important sign giving information about CDS spread of a counterpaty. It is not a constant variable but it might change throughout time as credit rating agencies updates ratings. Companies having a good rating tend to have lower CDS spread. However it is a categorical variable and is not able to capture all the information.

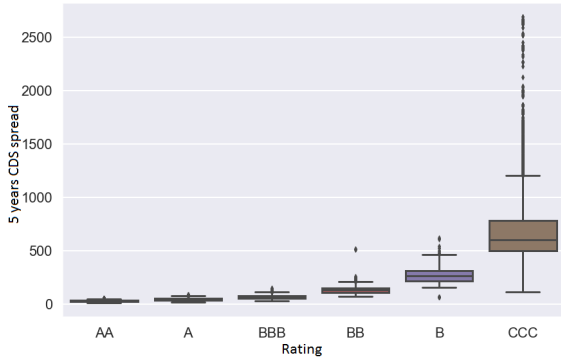
At Figure 4.8, the change in the ratings for CSCO and IBM companies can be seen. It is possible to track the change in CDS spread by looking at rating changes. For example, CSCO had rating 'BBB' at the beginning of 2012. If we compare it with the Figure 4.6, it is seen that in fact CDS spread of CSCO was higher in this period. Also, in late 2012, where CSCO had a rating of 'AA', its CDS spread has also decreased.

Figure 4.8: Credit rating changes for IBM and CSCO



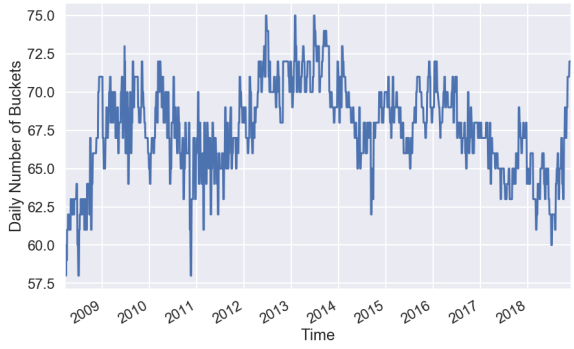
Another way to see the relationship between the ratings and CDS spread is through a comparison using box plots. Figure 4.9 shows the relations between the rating and CDS spread for the year 2017. As expected, we can directly see that there is a strong relationship between CDS spread and credit ratings.

Figure 4.9: CDS spread ranges for different credit ratings



As explained at related works section, current algorithms that are commonly used for CDS proxying, separates data into buckets where sector, region and rating is the same. Since companies can change rating over time, the number of buckets can change daily. Figure 4.10 shows the change in the number of buckets on a daily basis. This figure shows that on average there are approximately 70 buckets for 410 counterparties. This shows that on average there are 5 different companies in each bucket.

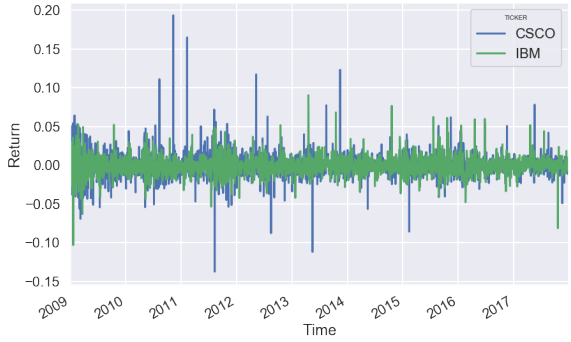
Figure 4.10: Daily number of buckets



Since, there are much more buyer and seller, stock market is much more liquid compared to CDS market. Also, as was discussed earlier, it might be possible to infer information about default probability of a company by using its stock price returns. However returns by themselves might be quite noisy. Figure 4.11, shows the change in daily stock returns for our example companies. Here it is seen that, it is very hard to infer information by only using stock return. However

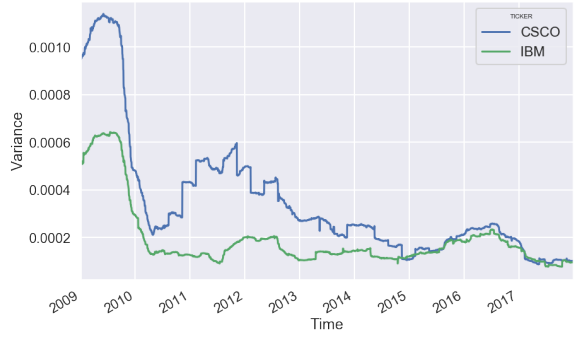
as explained at work done by Raymond Brummelhuis and Zhongmin Luo [18], it might be possible to aggregate stock returns for obtaining meaningful feature sets.

Figure 4.11: Daily stock returns



There might be several ways of aggregating stock returns. This will further be explained at the Section 4.3 where different feature sets that we engineered are explained thoroughly. One of the aggregations also forms the motivation of this work. As a general trend, it was observed that if the volatility of a public company's stock price is high, then CDS spread of the company was also high. This relation was also explained at [31] by European Central Bank. For seeing this relationship, 1 year volatility of stock returns for IBM and CSCO companies were shown at the Figure 4.12.

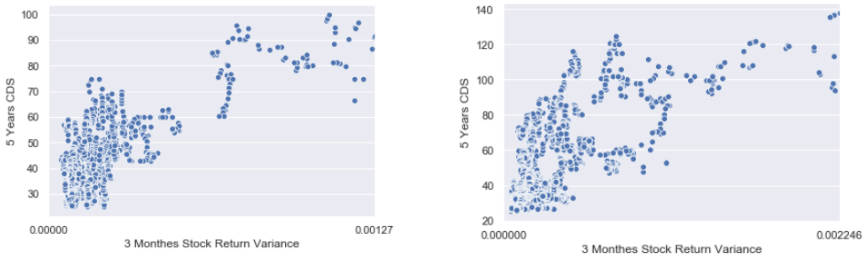
Figure 4.12: 1 year stock return variance calculated on rolling window



When Figure 4.6 and Figure 4.12 are compared, it is observed that there is a correlation between the variance of stock returns and the CDS spread of a

company. For better observing this relation, 3 months stock return variance was shown with CDS spreads as a scatter plot. Here every data point represents a day.

Figure 4.13: Relation between 3 months stock variance and CDS spread

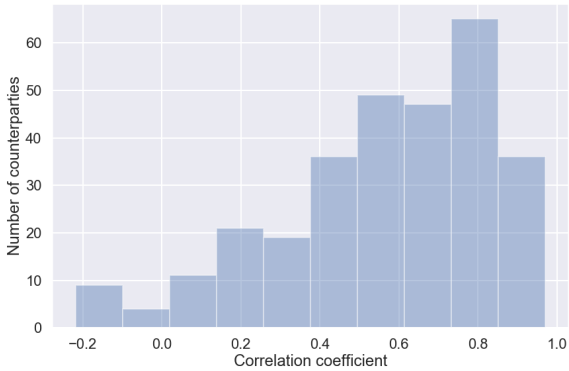


(a) IBM(correlation 0.70)

(b) CSCO(correlation 0.64)

As observed at the Figure 4.13, there is a positive correlation between stock return variance and daily CDS spread. The correlation cofficient was observed to be 0.70 for IBM and 0.64 for CSCO. To see if this is a fact for all different companies, the histogram of correlations of 3 month stock return variance and 5 years CDS spread for all 410 public companies is given at Figure 4.14. As seen here in fact for the majority of the counterparties, there is a significant correlation which creates the motivation to use stock return variances for predicting CDS spread.

Figure 4.14: Histogram of correlations between 3 month stock return variance and CDS spread for companies in our data set



As a conclusion, data investigation shows that both S1 data set (i.e sector, region, rating) and S2 data set (daily stock prices) gives information about CDS spread for a given counterparty. Thus both data sets were merged and used for feature engineering.

4.3 Feature engineering and analysis

At the earlier sections, the analysis of the both S1 and S2 data sets was done thoroughly. For extracting information from these data sources, one should process the raw data by creating structured features. Only this way, it would be possible to train machine learning models. For this reason, throughout the project we have developed 3 main feature sets from the raw data. In this section we will explore and argument for those feature sets.

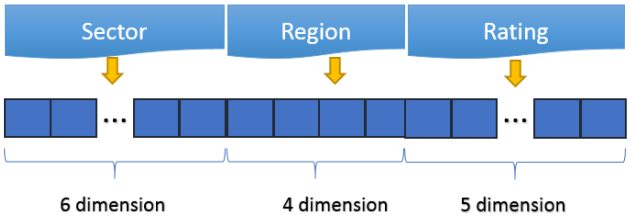
Please note that in our study, we have preferred to create feature sets by doing literature review and discussing with finance experts. An alternative approach could have been to use more machine oriented feature selection such as forwards/backward selection. However, we did not prefer this approach since we had the chance to use domain knowledge of experts. Moreover, it is possible to argue that feature engineering makes some models more interpretable which is one of the main concerns for financial institution in Machine Learning applications.

4.3.1 Feature Set 1 (FS1): sector, region, rating

After creting the golden data source by merging S1 and S2 data sets, there exists 7 different sectors, 5 different regions and 6 different ratings in our data base. Sector and region are both nominal variables thus should be processed by one-hot encoding. On the other hand, it is possible to assume rating as an ordinal variable. However as shown in the Figure 4.9, it is hard to define the relationship between different ratings. Interpreting a linear increase between rating notes could easily end up with a highly biased model. Moreover, our experiments has shown lower performance for treating the rating as an ordinal variable. Thus, the first feature set that we have used is one hot encoded version of **sector, region and rating** fields. Note that as rating might change, this is a daily feature so there is a data point for every day.If the model uses last n days for training and if there are k training companies, training set will be composed of $n \times k$ data points.

Figure 4.15 shows the FS1 vector. Note that for columns to be totally independent there are 1 less dimensions than there are categories for each sector region and rating fields. This is mainly for following the multicollinearity assumption of linear regression.

Figure 4.15: FS1



Even though, intrinsically FS1 cannot capture all the variance in the CDS spreads, it is generally a good measure for estimating the level of the curve especially when the model is trained daily. This is the reason why FS1 is the feature vector that is used for training Cross-Section and Intersection methods as explained at the work done by Nomura [9].

In our work, we used FS1 for implementing Cross-Section, Intersection methods and for implementing simple classification algorithm.

4.3.2 Feature Set 2 (FS2): sector, region, rating and stock variances

At the data investigation process, a significant correlation between the variances of daily stock price returns and CDS spreads has been observed. Therefore, another feature set that we have designed includes variances on different periods. For this, first daily stock returns was calculated using the following equation.

$$R_t = \frac{P_t}{P_{t-1}} - 1$$

Wher P_t is the stock price for the day t and R_t is the return.

As we have already shown at Figure 4.11, daily stock returns is very noisy and do not reflect the long term trend about how secure a counterpary is perceived. If

Figure 4.12 and Figure 4.11 compared, it is obvious that instead of using returns as raw, calculating variance of the returns is a better idea. This is the reason why the our approach was to calculate long term variances in a rolling window for stock returns. Variance calculation was done using the next equation.

$$\sigma_t^2 = \frac{\sum_{i=t-n+1}^{t} (R_i - R_\mu)^2}{n}$$

where n is the period and R_μ is the average return over the period and t is the date that feature vector is representing. As a result of experiments best performing and emprically reasonable windows were chosen to be 22, 64, 124, and 254 working days corresponding to respectively 1 month, 3 months, 6 months and 1 year periods.

Note that both rating and the stock variances change daily. Thus, similar to FS1, FS2 is also a daily feature meaning that there would be n data point for a given company, if the model uses last n days for training.

Since, return variances on different periods are expected to have a high correlation, it is relevant to see the correlation matrix for these extra features.

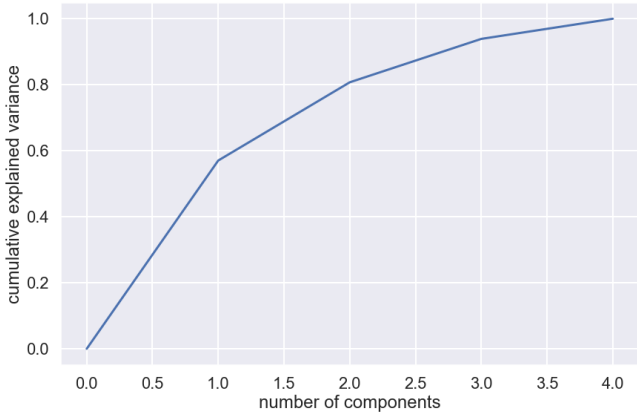
	1M	3M	6M	1Y
1M	1.0	0.59	0.44	0.31
3M		1.0	0.75	0.53
6M			1.0	0.71
1Y				1.0

Table 4.1: Correlations between the features

As seen at the Table 4.1, there is a high correlation between variances on different period. This is expected because the variance calculated on the last 6 months of period includes the days on which 3 months variance is calculated.

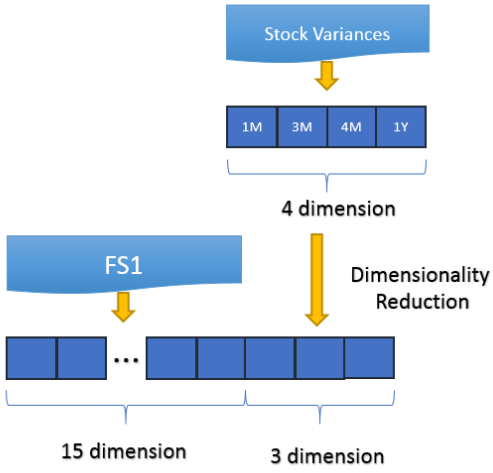
To understand FS2 further, Principal Component Analysis is carried out for stock variances. This is also important to obtain orthogonal fields as explained at Section 2.2.1. To decide if we need all 4 principal components, we have analyzed the variance explained by each principal component. After the analysis, it was decided to keep first 3 principal components which stands for 94% of the total variance as seen at Figure 4.16.

Figure 4.16: Explained cumulative variance on different principal components of FS2



After this analysis the formation for FS2 can be seen at Figure 4.17.

Figure 4.17: FS2

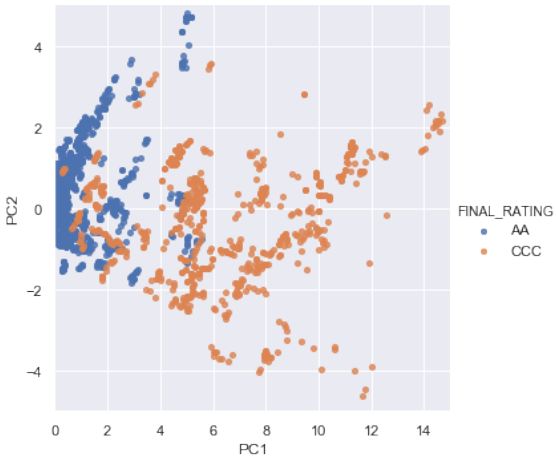


For analyzing FS2 even further, it was decided to visualize projection of data points on the first 2 principal components. For the visualization purposes, firstly we have filtered data points between 2010-01-01 and 2010-06-01. Secondly we

have filtered data points where the related counterparty had a rating of either 'AA' or 'CCC'. This is done because we expect a bigger difference for the counterparties that have bigger gap of ratings. After filtering, we have data about 142 companies for 108 working days.

Projection of the filtered data points can be seen at Figure 4.18. Here, coloring is done depending on rating. Analyzing the figure, it is possible to argue that companies that have 'AA' rating differ from companies with 'CCC' rating in terms of daily stock return. Actually, this result is expected since generally stock prices for low rated companies tend to be more volatile.

Figure 4.18: Projection of FS2, colored with rating



If we color the same graph depending on region and sector we obtain Figures 4.19 and 4.20. These figures show that stock returns capture the differences between basic information for companies and supports motivation for using FS2.

Figure 4.19: Projection of FS2, colored with region

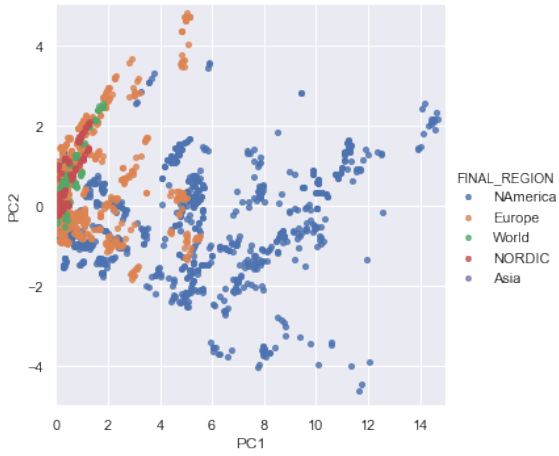
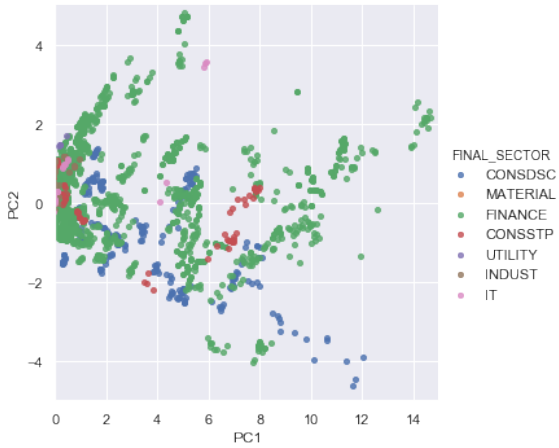


Figure 4.20: Projection of FS2, colored with sector



Another way to analyze FS2, is to do Linear Discriminant Analysis. This way it is possible to see if FS2 can distinguish between different companies in the same bucket.

Continuing the example at Section 4.2, the LDA projection of stock return variances in the technology sector, North America region and 'A' rating was visualized for the period between 2010-01-01 and 2010-06-01 at Figure 4.21.

Figure 4.21: FS2 LDA projection for data betweeen 2010-01-01 and 2010-06-01 in the bucket (Technology, North America, 'A')

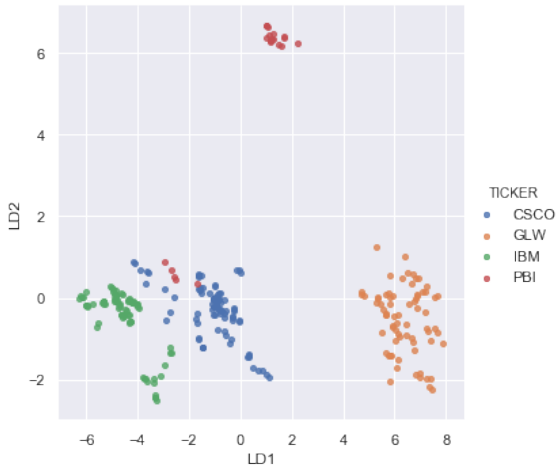
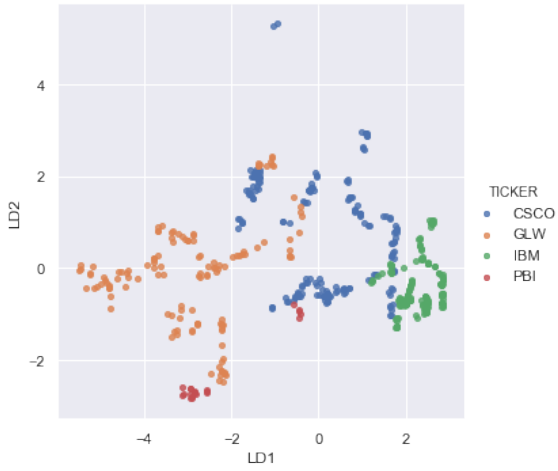


Figure 4.21 shows that each company has a characteristic stock performance which makes them easier to distinguish. Note that if the period we are analyzing is longer, probably it will be harder to distinguish different companies by just looking at their stock return variances. This is because in the long period companies might change their behaviour. For analyzing this, LDA projection for the same bucket was done on a longer period as shown at Figure 4.22.

Figure 4.22: FS2 LDA projection for data representing whole 2010



When we compare Figure 4.21 and Figure 4.22, we see that in the longer periods it becomes harder to distinguish companies based on their stock return variances. This is the reason why constructing the whole CDS spread time series for the whole year is very hard for a model without any recalibration. The best approach is to periodically calibrate the model. This is also why currently common models such as cross-section model are recalibrated daily.

Finally, another interesting point is to see if it is possible to distinguish different ratings using LDA dimensionality reduction. The Linear Discriminant Analysis above was done by optimizing on different company ids (tickers) and we had seen that stock variances is a discriminant factor for different companies. It is also interesting to see if stock variance is a discriminant factor for different ratings.

Since the number of companies in the Technology, N.Amer, 'A' bucket is relatively smaller, for visualization purposes of this analysis, we have run LDA dimensionality reduction on all the companies that is working at finance sector in North America.

Figure 4.23 shows the result of the projection. Here it is seen that FS2 definitely gives information about the rating of a counterparty. This shows that FS2 is a good measure for calculating CDS spread. Moreover, even though optimization is done on discriminating rating, data points belonging to the same companies are still clustered which can be seen at Figure 4.24.

Figure 4.23: FS2 LDA projection colored on credit rating

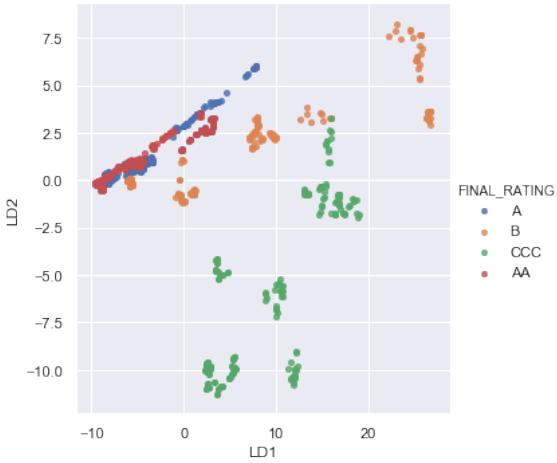
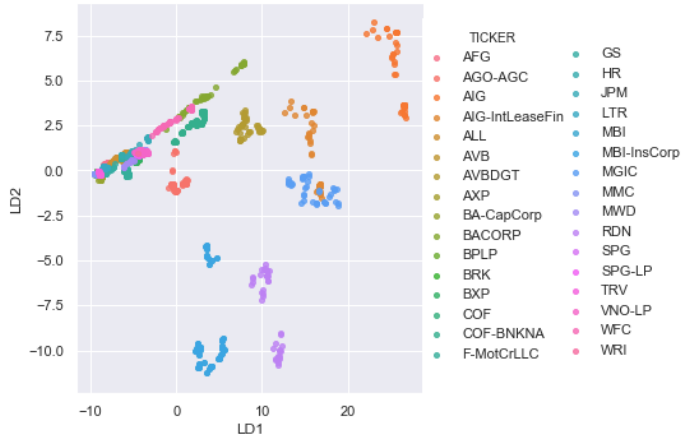


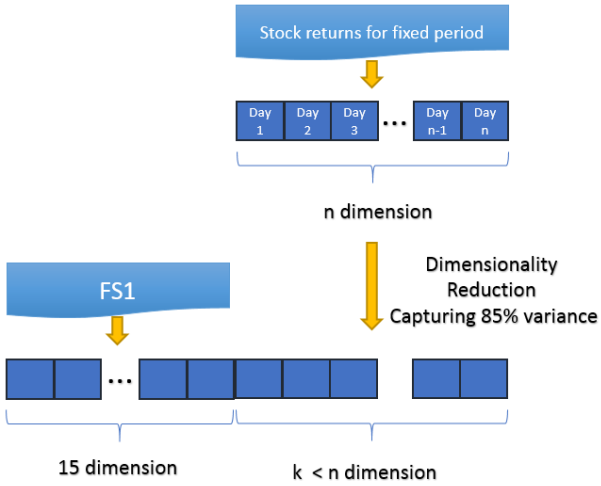
Figure 4.24: FS2 LDA projection colored on ticker name



4.3.3 Feature Set 3 (FS3): sector, region, rating, daily stock returns for fixed period

The main idea behind FS2 was that, at a given day, similar companies would have similar stock return variances. Another way to look at this is that, for a fixed period, similar companies would perform similarly in stock market. For example, if we expect that 2 companies are similar, we would expect that both performed similar last year. Thus, another feature set that we have designed includes stock price returns for a specified period. The period could be chosen freely, however throughout this project it was always chosen the most recent n days before the calibration(training) day. For example, if the calibration date is 2017-01-01, and length of the period is chosen to be 241 days, the feature vector FS3 includes all the daily returns for the period of 2016-01-01 and 2016-12-31 creating 241 dimensions. In practice, 241 dimension was never used as it is. Always dimensionality reduction was done using Principal Component Analysis or Linear Discriminat Analysis. Since, it is required to use sector,region and rating as input, FS1 is appended to the stock returns for obtaining FS3 feature set.

Figure 4.25: FS3



FS3 is not a daily feature vector. By the nature of FS3, there should be only 1 data point for a counterparty. This is because a company only performs ones at the stock market for a given period. However, for this period, the company might have had changed ratings. This means that there might be several FS1 vector to concatenate. For the example above, for the period of 2016, the stock returns create 241 dimension. However if the company changed ratings in this period which rating should we concatenate?

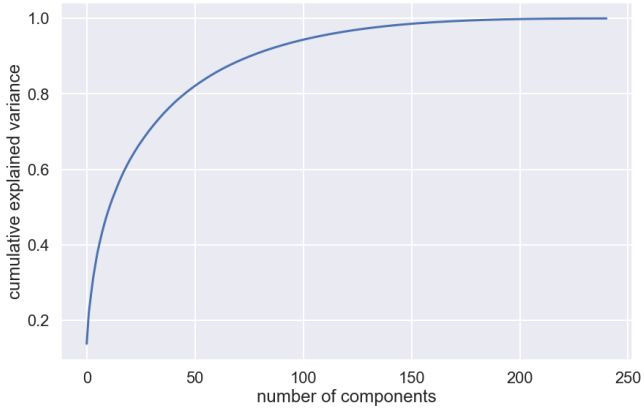
For tackling this problem, we have used a form of data augmentation. For example, if in the 250 days period, a company had rating 'A' for 50 days and 'BBB' for 200 days, we create 5 vectors having the same tail (daily stock returns for 250 days) where 1 of them has FS1 vector head with rating 'A' and 4 of them has FS1 head with rating 'BBB'. Note that in this case all the 4 vectors representing 'BBB' rating would be exactly the same. For creating some variance, we add Gaussian noise on their tails as well. The noise is added before dimensionality reduction and the variance of the noise is chosen to be 1/10 of the variance of the stock returns for the given period for that company. This way, we could obtain multiple data points for each company so it will be possible to train models with FS3.

To understand FS3 further, we have run Principal Component Analysis and Linear Discriminant Analysis as we have done for FS2. For this, we have created stock returns vector from 2016, for each company. As mentioned above, this

vector has 241 dimensions (there are working 241 days) and there are 410 data points for 410 companies.

After PCA, to decide if we need all 241 principal components, we have analyzed the variance explained by each principal component.

Figure 4.26: Explained variance on different components of FS3

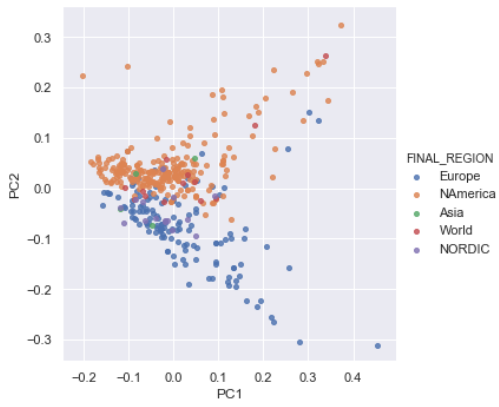


As seen at Figure 4.26, the first 55 principal component was enough for covering 85% of variance. Note that this number is only true for this period and at the actual running the model is calibrated periodically which means the the number of components covering the 85% of the variance might change.

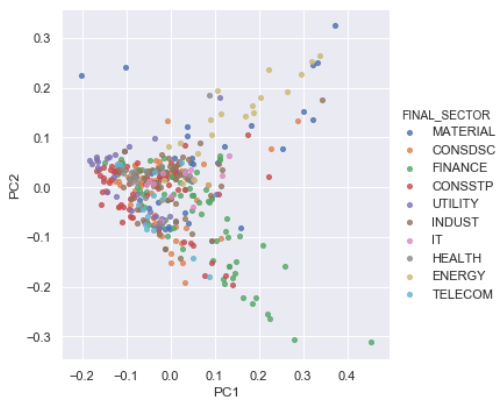
To visualize the data, the projection of data points on first two principal components was plotted at Figure 4.27, by using different coloring, based on sector region and rating. Note that for FS3, the first 2 principal components only represents 27% of the variance. Thus, the projection is much less representative compared to the projection we did for FS2. This is the reason why it is not possible to observe any grouping depending on sector and rating. However as seen at Figure 4.19, first two principal components captured about region information.

Figure 4.27: FS3 projection on two first Principal Components colored by (a) rating, (b) region and (c) sector fields

(a)



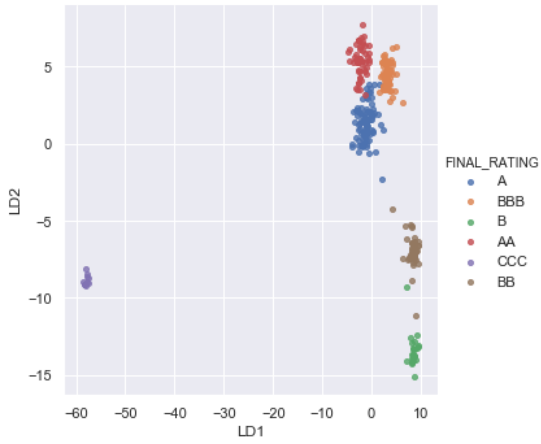
(b)



(c)

Linear Discriminant analysis is much more representative for understanding FS3. If we run LDA on this data by optimizing the separation between ratings we obtain the following projection as given at Figure 4.28. Here every data point represents a counterparty and it is seen LDA on FS3 creates very good separation between different rating groups. Moreover, companies with rating 'AA' are close to the companies with rating 'A' and 'BBB' respectively. This shows that companies that have close ratings actually performs similar. This supports the motivation for FS3.

Figure 4.28: FS3 LDA projection



Chapter 5

Implementation

This section briefly describes the implementation of the project as well as external software libraries that was utilized.

5.1 Software

All the programming in this project was done using Python 3.6.2 programming language. As of 2019, Python is one of the most popular programming languages for machine learning applications. There are many packages specialized for implementing machine learning models for Python.

In this project, we have leveraged especially 3 machine learning libraries. These are *keras* and *scikit-learn* libraries. Both of these libraries provide easy implementation and quick prototyping.

In addition, we have heavily used *pandas 0.24.2* library for data manipulation. Visualizations were done using *matplotlib 3.1.0* and *seaborn 0.9.0* libraries. Statistical analysis was done using *statsmodels 0.10.0*. The show cases and proof of concepts were implemented using *Jupyter Notebook*. Nordea's internal *Bitbucket*

accounts were used for the version control together with *git*. We used *Microsoft Teams* and *facebook messenger* for the communication.

5.2 Structure of the project

There are 4 main folders in the project repository. Those are "Notebooks", "Models", "DataProcessor" and "Utils".

"Notebooks" folder includes all the Jupyter Notebook implementations that includes all the visualizations and models that were experimented in this project. Here, it is possible to see the flow of different classification and regression models. The naming convention was chosen so that the reader would understand which machine learning algorithm was implemented, which engineered feature used and what type of evaluation were used to measure the effectiveness of the model. For example 'DecisionTree_FS3_All_mse.ipynb' file under the folder 'Notebooks/Classification/DecisionTree' shows the proof of concept for Decision Tree algorithm that was trained on the whole training data set using FS3 feature set and the model evaluation was done using mean squared error measure.

"Models" folder includes scripts for models that has a specific nature and which was not exactly implemented in scikit-learn.

"DataProcessor" folder includes scripts for slicing and dicing the data. This way Jupyter Notebook showcases was kept simple and independent from data manipulation.

"Utils" folder includes all the helper functions and classes. For example, standard settings for plotting graphs were handled by Python classes written under this folder.

Chapter 6

Model evaluation and running flow

6.1 Model assessment and selection

As it will be explained in the following sections, CDS spread time series construction problem was handled using either by assigning proxy companies or by predicting directly CDS spreads. Thus both classification models and regression models were experimented. This is the reason why different evaluation methods were examined to compare classification and regression models.

6.1.1 Classification evaluation

There is no other company who reflects another company's CDS dynamics perfectly. Thus, it is hard to confirm whether or not a proxy classification is correct. Assume that we need to find a proxy company for IBM. We ask our model for a good classification and our model returns Google as a proxy. This classification looks good as both firms generally perform similar. However, it is impossible to say if this classification is actually right or wrong. Thus, throughout this

project we had to come up with different evaluation structures for classification models.

As explained in [7], one possible way to check if an approximation model actually runs successfully, is testing the model with data belonging to a company that the model already knows. The idea here is that the best approximation of a given company is itself. Of course one should be careful not to mix training and testing data. For this, after extraction, data is split into test and training sets. For metric measurement K-fold cross validation is used. With this approach, if the data is collected daily, and if the number of different companies we have in our data set is C for d days, the total data would consist of $C \times d$ data points where each point belongs to a company. After this, we randomly split the data into train and test sets. Please note that after the split, in both training and test sets, data points from the same company will exist as shown in the figure 6.1. After training, if we give a test data point originally belonging Google company, we should actually expect the model to classify the point to Google class. This measurement should be iterated k times as K fold cross validation convention states. This evaluation will be called **E1**. Throughout this work, for E1, we have reported *accuracy*.

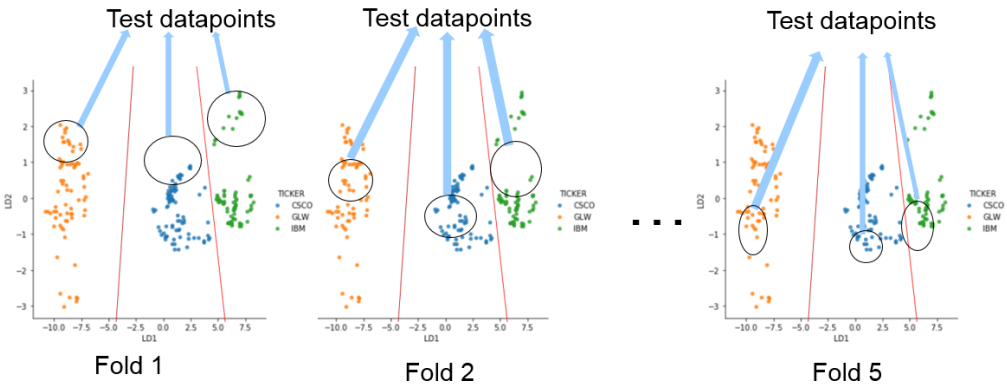


Figure 6.1: Visualisation of an example bucket classification using E1 evaluation using FS2

The fundamental idea of classifying a proxy is to use proxy company's historical CDS time series instead of the actual. Thus, if the classification is good then the time series of the classified company is expected to be closer to the actual. So, E1 by itself cannot capture how close the proxy time series is. Moreover, for E1 evaluation, we are splitting data randomly. This means that data points from consecutive days could end up being separated. However, this 2 data points are

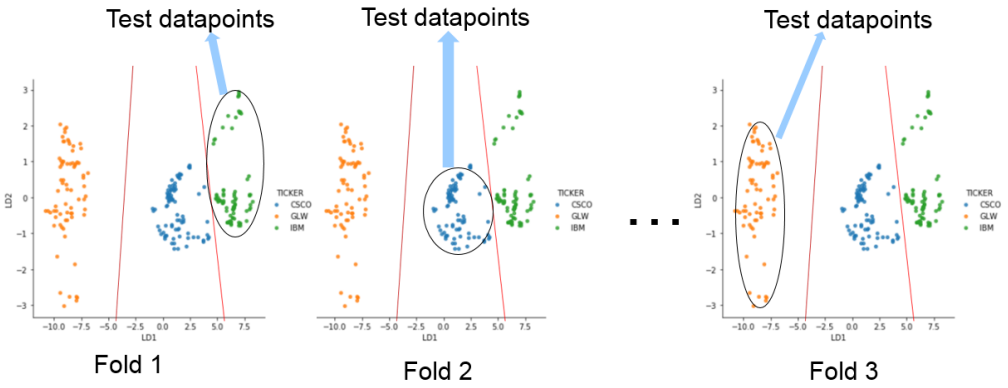
expected to be really similar since market dynamics would not differ too much. This is why it is an easy task for a machine learning algorithm to perform well on E1 evaluation. Even so, in this work we have decided to do E1 evaluation to replicate [7]. This way we wanted to cross check the arguments explained in that paper.

For capturing changes in the market, the classification should be run daily. This would create a time series of predictions. After obtaining the daily proxies, it is possible to replace them with actual spreads of the assigned proxy at the prediction day. By doing this, we could construct time series which then will be used as the proxy spread for the company of which CDS spread is unknown. Following the example above, we could daily give IBM's features (such as stock prices or ratings as will be explained in the following sections) and get the daily proxy companies. For example one day, the most similar company to IBM might be Google while the other day it is Apple. By concatenating CDS spreads of Google and Apple respectively, we could form a CDS spread series. Now, it is possible compare the constructed CDS spread series with IBM's actual historical CDS spread. The difference between those two graphs would then show the success for our model. If the actual graph and the constructed graphs are close than our models assigned good proxies. The evaluation of closeness between the prediction time series and the actual time series will be called **E2**. Since a good proxy time series would be close distance and would have similar trends, for E2 we have reported several metrics including *mean squared error*, *R squared* and average *Pearson's correlation* between the prediction series and the actuals. As seen in figure 6.2, for E2 evaluation we split the companies into test and train and evaluate different test companies at every cross validation fold. Note that even if the classification algorithm runs perfectly, the constructed graph will never be exactly the same as test company's actual CDS spread time series. In other words, it is impossible to make the errors between the predicted and real values zero just by assigning a company to another one. This is because no two companies have perfect correlation but they are unique. However, it is possible to say that if we make good classification then the difference between constructed and real graphs should be small.

6.1.2 Regression evaluation

Contrary to classification, in regression methods instead of assigning a proxy company, we directly predict the CDS spread. Thus, we could construct the proxy graph using the predictions. This means that E2 evaluation is valid for regression models. This way, we would be able to compare classifications models with regression models. Note that even though it is meaningful, such a comparison is not always fair. This is because regression methods are optimized

Figure 6.2: Visualisation of an example bucket classification using E2 evaluation using FS2



to reduce mean squared error while classification algorithms are optimized to maximize accuracy. However, it is interesting to see if assigning proxies actually construct close to real graphs.

6.2 Model running flow

This work is based on the following idea. We have the training companies for which we know daily CDS spreads, daily stock prices, daily ratings as well as the sector and region. We are trying to daily estimate some test companies' CDS spreads using only sector, region, rating and stock prices between first day and last day of 2017. In order to do this we are training models; use those models for a period of time for predicting the CDS spreads and recalibrate the model with the extra data obtained. For example, if the calibration frequency is 1 month, used data length is 2 years the model runs as follows. At the beginning of the year, the dataset consisting last 2 years information is gathered. A machine learning model is trained with the collected data, for 1 month period everyday CDS spread of the unobservable company is predicted using one of the features explained earlier till the next calibration date.

For this work we have developed many different models. The main difference between the models is whether the model is a classification type or a regression type. As explained earlier, classification models are assigning a daily proxy to

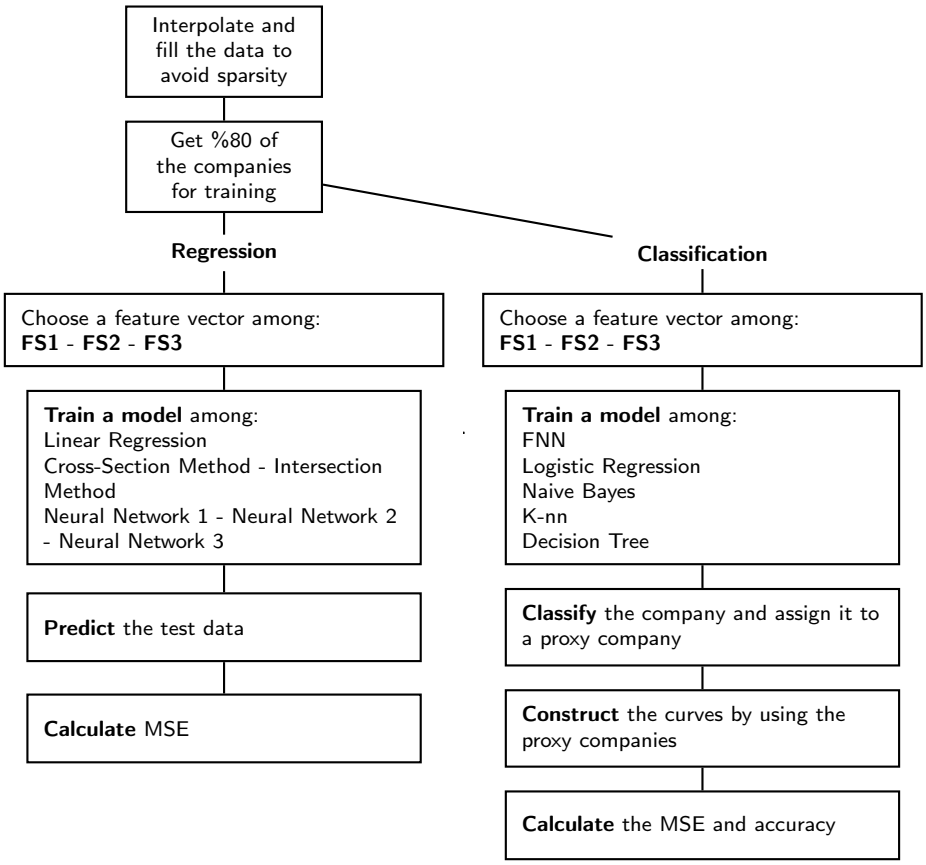
our test company while regression models directly estimate CDS spread. Thus, fundamentally, all the models have one final aim: to construct the most realistic time series to the test company's CDS spread.

For classification models, we run a prediction everyday for classifying the daily data of the unobservable company to one of the companies in our dataset. Finally instead of using the class, we use the actual CDS spread of the proxy company. This way it is possible to construct CDS spread time series using classifications.

For regression models, the flow is simple Since regression models directly estimates CDS spreads, for these models it is natural to construct the times series directly only using estimations.

Another important separation between the models is whether the model is an **in-bucket** model or **out of bucket** model. For in-bucket models we group the training data into buckets representing companies which are in the same sector, region and which have the same rating. For every group a separate machine learning model is trained. This way, for example classification in bucket models can never predict a proxy company which is not in the same sector, region and rating with the test company. On the other hand, for out-of-bucket models, there is only one machine learning model trained with the all training data. These models, as well uses sector region and rating information using FS1 feature vector, however they are not forced to do a classification to a company in the same bucket. In the following figure a high level work flow we have run in this project can be seen.

Figure 6.3: High level workflow of the proxy assignment process



Chapter 7

Experiments & results

In previous chapters, we have presented our approach, data and feature sets. In this chapter, different methods and experiments conducted throughout this project will be explained along with the results.

The structure of this chapter is as follows. We first describe classification models that we have implemented. Next, we describe the regression experiments including conventional methods and experiments that we have done with our feature sets. Finally, we describe Deep Learning models in a separate section. For all the models we have reported E2 evaluation(comparison of time series). We have also reported E1 evaluation(accuracy comparison) for classification algorithms where it is applicable.

7.1 Classification experiments

One way to construct CDS spread time series is to find the most similar company to the unobservable and assume that its CDS spread would behave the same on the prediction day. For example, if we have Google in our dataset but not IBM, we could probably use Google's CDS spread instead of IBM's. This is a classification problem. The model should be able to classify IBM data to Google

class.

The proxy classification is a dynamic process thus everyday, the model predicts a proxy company. For example, if we are to predict CDS spread series of 2017 for IBM, then everyday in 2017, model classifies the daily data to one of the companies. This forms a time series of proxy predictions which would be converted to CDS spread series. Note that, for consecutive days we expect the models to classify daily input data to the same class (proxy company) since generally daily data is similar in consecutive days. However, if there is a strong change in the data of that day, the assigned class might change as well.

As explained earlier, we have engineered 3 different feature sets. From all feature sets, for FS1, having E1 evaluation is irrelevant. This is because FS1 includes only sector region and rating fields. However, at the classification date, there are generally more than 1 company which has the same sector, region and rating fields. This means that there is no further information in FS1 which would help the model to distinguish between 2 different companies in the same bucket. Thus only a random selection could be possible. However, this still may not be a totally wrong approach. A random selection from the bucket would surely catch the dynamics in the bucket and the time series constructed by using these random predictions would not be far from the actual CDS spread time series. Thus, it was decided to create a random selection pipeline to use as a benchmark.

Secondly, for FS3, E1 evaluation does not apply as well. This is because FS3 represents stock history of a company and there is only one data point for a given company. Thus it is impossible to split data points for measuring E1. However, E2 evaluation was exercised as was done to all the different models discussed in this work. This is possible because for E2 evaluation, there are different counterparties in test and train sets.

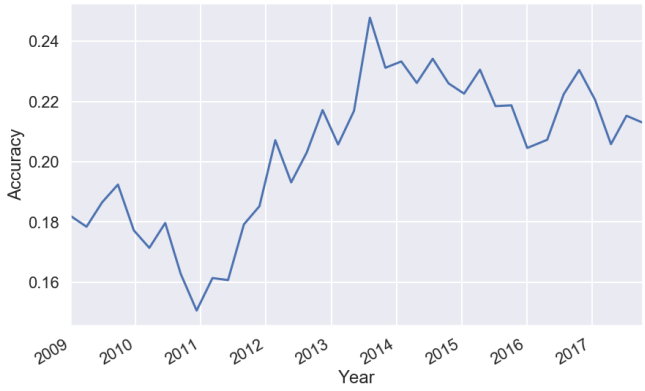
7.1.1 Classification using FS1 (benchmark)

The most simple way to do a classification using FS1 is basically assigning a random proxy to our unobservable counterparty. This random proxy should be from the same bucket meaning that from the same region, sector and rating.

Using E1 evaluation, if we set the period to 2017-01-01 to 2017-03-31 (65 working days, 1 quarter), the number of data points will be 26650 since there are 410 companies in our dataset. Using 5 fold cross validation, if we run the E1 evaluation as explained in Section 6.1, we get 0.21 accuracy. This value is expected since on average there are 5 companies in each bucket.

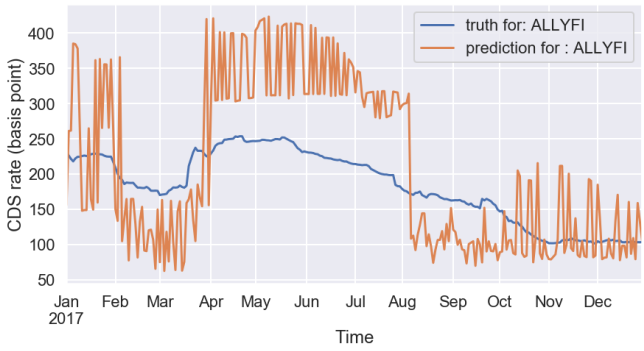
If we run E1 evaluation on different periods of same length between 2009 and 2017, we obtain Figure 7.1. Here you can see that, for different periods, random classification model performed similarly. The performance of course depends on the average number of companies in each buckets. As we had shown at Figure 4.10, the number of buckets changes with time thus the average number of companies in each bucket also changes.

Figure 7.1: Accuracy of random classifier on different periods



One flaw of this approach is that if we assign a new company everyday there will be jumps at the graph we construct. Figure 7.2 shows time series constructed by the predictions given by random classification approach. As seen between each time steps there are significant jumps.

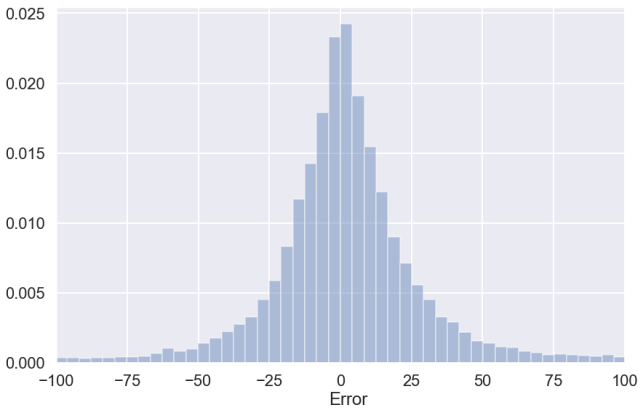
Figure 7.2: An example time series predicted by random classifier



To understand if the model has a significant bias,it is important to visualize error

distribution. The histogram showing the distribution of errors (aggregating all the folds in cross validation) is shown at Figure 7.3. It is seen that the error distribution is close to normal thus the model does not have a significant bias.

Figure 7.3: Error distribution



7.1.2 Classification using FS2

As explained before, FS2 includes stock return variances as well as containing FS1. For this feature type, we have experimented different classification algorithms including Logistic Regression, Decision Tree, K-Nearest Neighbors and simple 2 hidden layer, fully connected Feed Forward Neural Networks. All of these models were both experimented in-bucket and out-bucket. For in-bucket experiments, every single bucket has its own model. When the test data comes in, the model of the bucket that the test data belongs to, returns the prediction.

In-bucket models

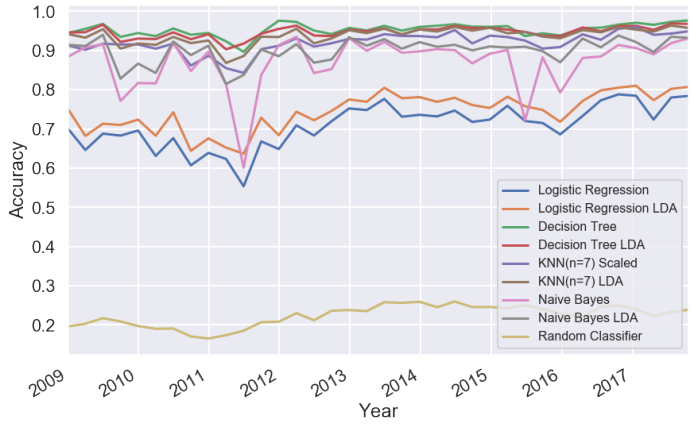
By its nature, for the first evaluation E1, in-bucket models has a comparative advantage. This is because in these models, each model should only learn to classify between different counterparties inside the bucket. For training in-bucket models using FS2, the training data is grouped by sector region and rating fields. In each bucket, a separate classifier is trained using only the stock variances. Here, we exclude sector, region, rating fields because, in each bucket these fields are same. As for all the classification models, the training is done using the target as the company id (ticker).

Note that since the rating is a daily parameter, for a model that was trained using multiple days, the same company can be present on different buckets. This is the case if the company changed rating during the period. For example, for daily feature sets such as FS1 and FS2, if a company had rating 'A' for 50 days this many data points goes to the bucket 'A'.

At Figure 7.4, the prediction accuracy for different periods can be seen. Here, you can observe that especially, KNN and Decision Tree models performed well for the E1 evaluation. This is expected since FS2 does not differ too much between different days if the days are close.

If we analyze the figure even further, we see that all different models performed pretty well over all. This is actually expected since this part of the work, was the replication of Zhongmin [7]. After the experiments we obtained the same outputs with this paper . However please note that in [7], E2 evaluation was not shown and success in E1 may not always mean success in E2.

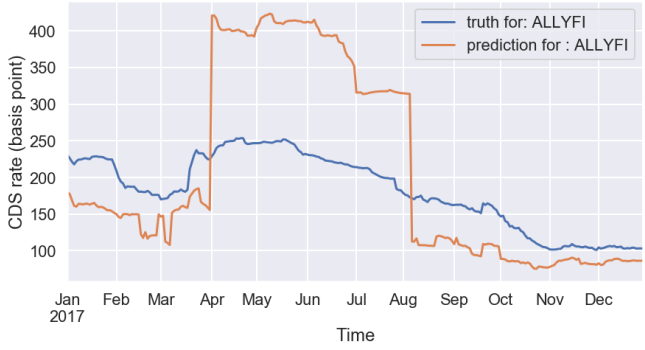
Figure 7.4: Accuracy of different **in-bucket** models on different periods



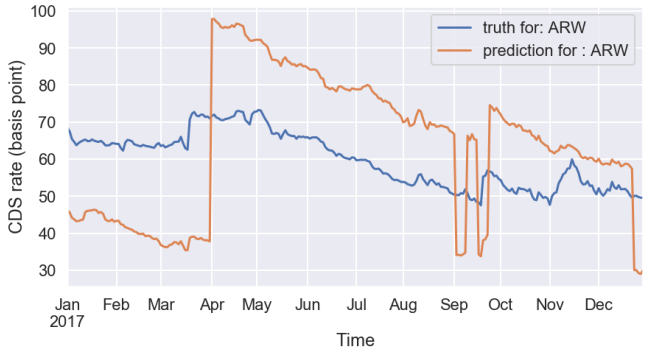
If FS2 classification models are compared to Random Classifier, it is seen that these models are much more insistent on their predictions. This means that the proxy company does not change everyday but the model insist on its decision on consecutive days. This creates comparatively stable time series predictions. Figure 7.5, shows example time series constructed by the predictions from in bucket Decision Tree model. Here the model changed the proxy company a few times through whole year. If we compare this with Figure 7.2, it is seen that having a classification is much more convenient compared to random guess. On the other hand, these models are still found to be vulnerable to sudden jumps

whenever model prediction changes as seen at Figure 7.5 (b).

Figure 7.5: Time series constructed by daily in-bucket Decision Tree classification predictions



(a)



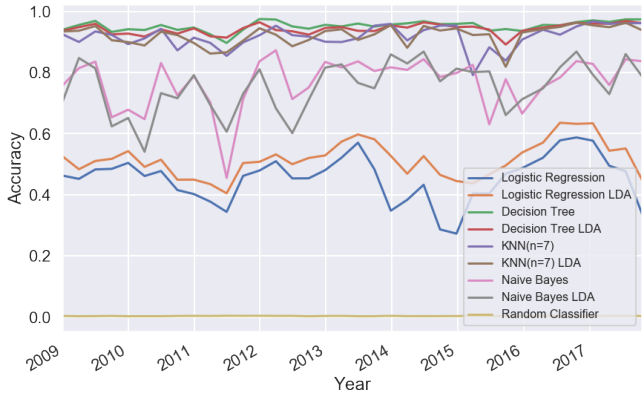
(b)

Out-bucket models

Out-bucket models have both an advantage and a disadvantage. This is because they are solving a harder classification problem since there are 410 different counterparties. This definitely makes it more challenging for the accuracy evaluation. On the other hand, this can be an advantage for the E2 evaluation. For out-bucket models, it is possible to classify to a target company which is not exactly in the same bucket. So if the test company is in the wrong bucket somehow, i.e the rating is about to change, then a well trained out-bucket model can predict better proxy compared to in-bucket models.

The Figure 7.6, shows the accuracy measure on different periods for out-bucket models. Similar to in-bucket models, Decision Tree and KNN algorithms performed better on E1 evaluation.

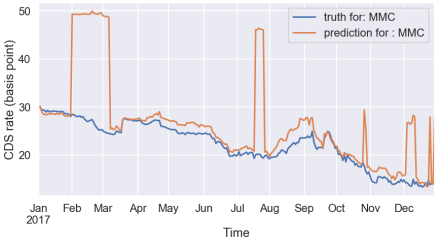
Figure 7.6: Accuracy of different **out-bucket** models on different periods



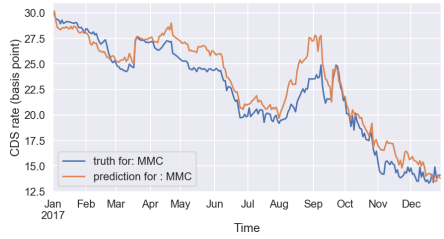
As described before, E1 measures how well a model can find the company that the test data is belong to. However FS2 describes historical stock return variances which does not change significantly between consecutive days. This is the reason why simple algorithms like KNN and Desicion Trees performs better on this task. On the other hand, E2 is a better evaluation method since it compares actual predicted time series.

Surprisingly, after constructing time series, we have seen that the models that performed good on E1 evaluation did not performed well on E2. This shows that our assumption on E1 was not correct. Therefore, even if a model is good at finding which company the test data belongs to, it might not be a good model for finding the best proxy. For understanding this better, several predicted time series by out-bucket Decision Tree model which performed relatively good on E1 evaluation and out-bucket Logistic Regression model which performed poor on the E1 evaluation was shown at Figure 7.7. It can be seen that Logistic Regression was much more insistent in its prediction which created better graphs.

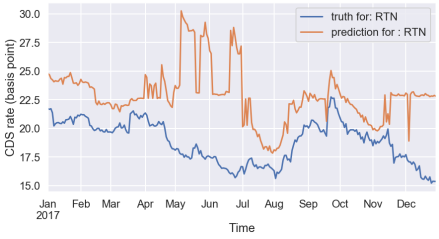
Figure 7.7: Different predictions by Decision Tree(left) and Logistic Regression(right



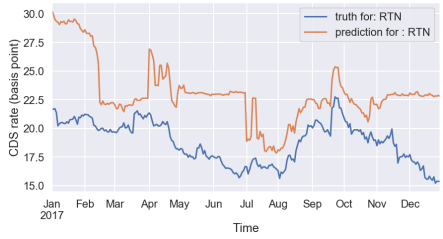
(a)



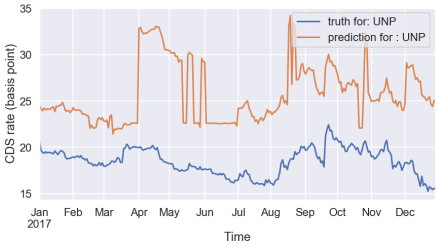
(b)



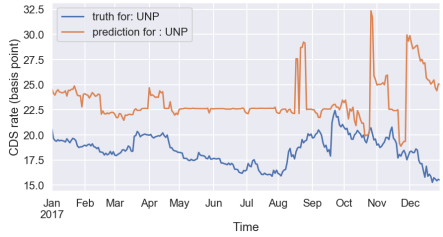
(c)



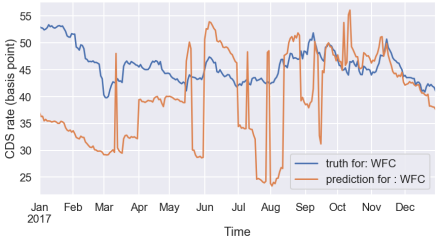
(d)



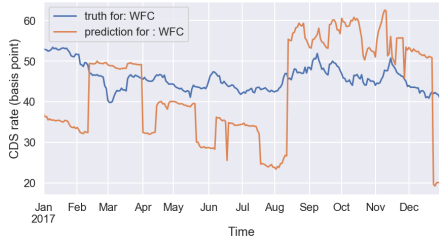
(e)



(f)



(g)



(h)

Table 7.1 shows the performance of different Machine Learning algorithms using E1. This table is basically obtained by processing Figure 7.6 and Figure 7.4. Accuracy performance is shown as percentage accuracy and standard deviation is the standard deviation of the accuracy measured in different periods. 5 fold cross validation was applied at each period. Also please note that FNN model is too slow to train thus it was impossible to measure by training every 3 months between 2009 to 2018. We have only measured it in 2017.

E2 evaluation performance for different classifiers is shown at Table 7.2. For this evaluation, as explained in section 6.1, counterparties are divided into test and train counterparties (5 fold cross-validation, different counterparties at every fold). Models are trained using only train counterparties and daily CDS spread time series of 2017 were constructed for test companies. In this table it is possible to see average correlation between predicted and real time series as well as mean squared error. The last column represents the number that shows how many times a model changed its prediction. Note that the best model would have low mean squared error, high correlation and low prediction change.

Table 7.1: Performances of the classifiers using E1 (using 5 fold cross validation)

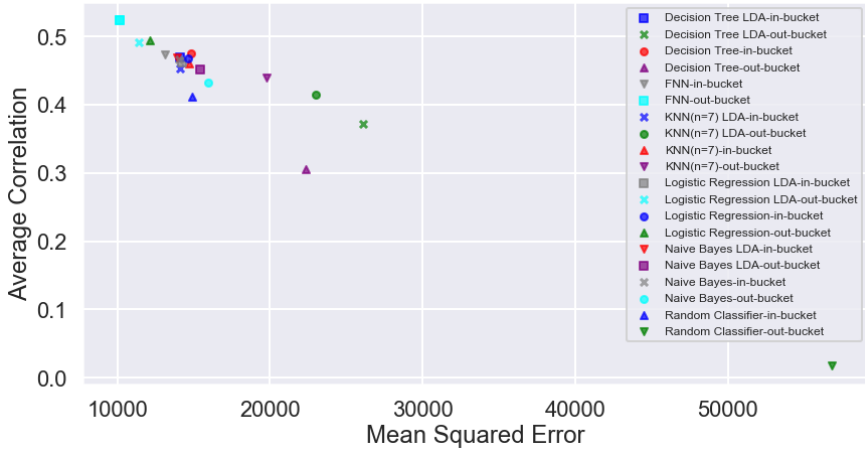
Name	Feature	Type	Acc.	Acc. Stand. Dev.
Random Classifier	FS1	In Buc	22.3	2.8
KNN	FS2	In Buc	92.0	2.6
KNN(+LDA)	FS2	In Buc	93.8	2.1
Logistic Regression	FS2	In Buc	70.8	5.5
Logistic Regression(+LDA)	FS2	In Buc	74.4	4.8
Naive Bayes	FS2	In Buc	87.0	6.6
Naive Bayes(+LDA)	FS2	In Buc	90.0	3.1
Decision Tree	FS2	In Buc	**95.3**	1.6
Decision Tree(+LDA)	FS2	In Buc	94.7	1.4
FNN	FS2	In Buc	75.1	1.8
Random Classifier	FS2	Out Buc	2.2	0.4
KNN	FS2	Out Buc	91.2	2.7
KNN(+LDA)	FS2	Out Buc	93.1	2.8
Logistic Regression	FS2	Out Buc	55.2	6.4
Logistic Regression(+LDA)	FS2	Out Buc	58.8	5.7
Naive Bayes	FS2	Out Buc	68.6	11.1
Naive Bayes(+LDA)	FS2	Out Buc	68.4	12.4
Decision Tree	FS2	Out Buc	95.2	1.7
Decision Tree(+LDA)	FS2	Out Buc	**95.3**	1.8
FNN	FS2	Out Buc	70.1	1.3

Table 7.2: Performances of the classifiers using E2 (using 5 fold cross validation)

Name	Feature	Type	MSE(10e3)	Corr.	Pred. Change
Random Classifier	FS1	In Buc.	14.9	0.40	137.2
KNN	FS2	In Buc.	14.7	0.46	13.0
KNN(+LDA)	FS2	In Buc.	14.1	0.45	9.8
Logistic Regression	FS2	In Buc.	14.6	0.47	10.5
Logistic Regression(+LDA)	FS2	In Buc.	14.2	0.46	10.7
Naive Bayes	FS2	In Buc.	14.0	0.47	9.8
Naive Bayes(+LDA)	FS2	In Buc.	13.9	0.47	9.6
Decision Tree	FS2	In Buc.	14.1	0.47	9.8
Decision Tree(+LDA)	FS2	In Buc.	14.3	0.46	10.5
FNN	FS2	In Buc.	**13.3**	0.47	9.5
Random Classifier	FS2	Out Buc.	56.8	0.02	194.4
KNN	FS2	Out Buc.	19.7	0.44	12.2
KNN(+LDA)	FS2	Out Buc.	23.0	0.41	12.0
Logistic Regression	FS2	Out Buc.	12.2	0.49	7.7
Logistic Regression(+LDA)	FS2	Out Buc.	11.4	0.50	8.5
Naive Bayes	FS2	Out Buc.	15.9	0.43	9.9
Naive Bayes(+LDA)	FS2	Out Buc.	15.4	0.45	9.8
Decision Tree	FS2	Out Buc.	22.3	0.30	10.6
Decision Tree(+LDA)	FS2	Out Buc.	26.1	0.37	11.3
FNN	FS2	Out Buc.	**10.1**	0.52	7.3

Finally, for a visual comparison, the scatter plot showing the average correlation and mean squared error is shown in Figure 7.8

Figure 7.8: Scatter plot comparing different classifiers



7.1.2.1 Classification using FS3

Feature Set 3, constructs the idea of comparing the past stock market performance for different companies. The idea here is that similar companies perform similarly on stock markets at a given fixed period. For example, it is possible to match companies that performed similar last year. This idea is the same with FS2, however instead of using variances, here we directly use raw stock returns.

As explained at Section 4.3.3, there exists only one FS3 data point for each counterparty. This means that before any manipulation, if we have 300 counterparties in our training data set, we would also have 300 data points. One major issue in this approach is that training a 300 category classification algorithm with 300 data points is impossible. For this reason, a data augmentation process was implemented to obtain 25 data points for each counterparty. This process is run as the following. For example assume that we are using a fixed period of 250 days for creating FS3 data points and a given company in our training data had rating 'A' for 200 days and had rating 'B' for the rest. In this case, we create 25 extra data points for this company where 20 of them is treated as rating 'A' and 5 of them as rating 'B'.

As explained before, FS3 vector has relatively high number of dimensions which has potential noise. For avoiding any multicollinearity, Principal Component Analysis was applied to reduce the number of dimensions.

Finally, for FS3 models E1 evaluation was not exercised. Instead, model comparison was realized only using E2 evaluation. This is because for FS3, before data augmentation a counterpary only has 1 data point which makes it impossible for test/train separation.

In-Bucket models

Similar to the FS2 classification examples, FS3 in-bucket classification only returns a company which has the same rating, sector and region with the test company. For this, in each bucket a separate model was trained only using the data which belongs to the bucket.

Out Bucket models

As it was done for FS2 models, we have experimented out-bucket models for FS3. For these models, there is only one Machine Learning model which captures all

the data instead of training a separate model for each bucket.

Table 7.3: Performances of the FS3 classifiers using E2 (using 5 fold cross validation)

Name	Feature	Type	MSE(10e3)	Corr.	Pred. Change
KNN(+PCA)	FS3	In Buc.	19.5	0.53	4.2
Logistic Regression(+PCA)	FS3	In Buc.	23.1	0.52	4.6
Naive Bayes(+PCA)	FS3	In Buc.	14.1	0.61	4.6
Decision Tree(+PCA)	FS3	In Buc.	23.8	0.38	5.7
FNN	FS3	In Buc.	13.5	0.62	4.2
KNN(+PCA)	FS3	Out Buc.	23.0	0.41	12.0
Logistic Regression(+PCA)	FS3	Out Buc.	11.4	0.50	8.5
Naive Bayes(+PCA)	FS3	Out Buc.	15.4	0.45	9.8
Decision Tree(+PCA)	FS3	Out Buc.	26.1	0.37	11.3
FNN	FS3	Out Buc.	10.1	0.52	7.3

Table A3.1 shows the mean squared error of different classifiers that uses FS3. Here it is possible to observe that the average correlation for these models are actually higher. This is because using FS3 creates more robust models in the sense that the models tend to keep its prediction. On the other hand mean squared error of these models are much higher.

Figure 7.9: A time series constructed by logistic regression classification predictions with FS3

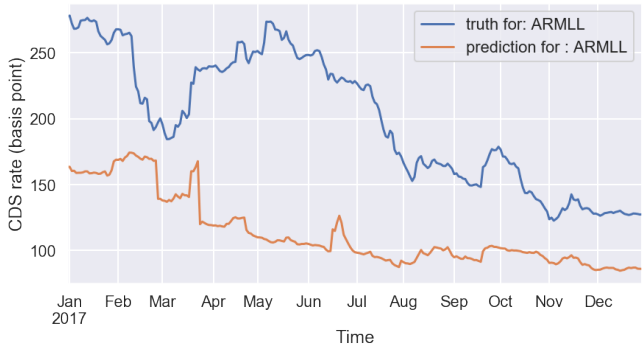


Figure 7.9 shows an example graph predicted using logistic regression trained with FS3 data. Please observe that in this time series classified target only changes few times. While this created a time series which is visually close to the

actual series, mean squared error is still expected to be high for the first half of the year. The performance of different models can be seen at the scatter plot in the Figure 7.10.

Figure 7.10: Scatter plot comparing different classifiers. Average correlation is the correlation between predicted time series and the original time series

Note that for all models, error distribution was observed as normal distribution. At the Appendix A3 error distribution for best performing models of FS2 and FS3 can be seen.

7.2 Regression experiments

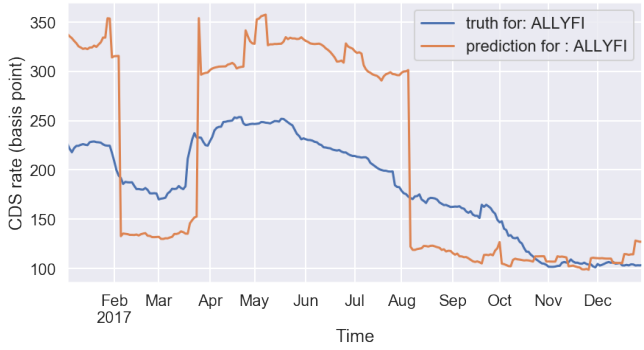
7.2.1 Intersection method

Intersection method is one of the two methods actively used in finance sector with *cross-section method*. For creating a baseline in this work we have implemented both of these models. The main idea for intersection method is to get the mean spread inside all the buckets and use this value as the prediction for the test company. Please note that the mean value is updated daily thus we expect the model to capture the trend in the market.

One of the main flaws of this approach is that if a company changes ratings, its bucket also changes. This creates a jump in the time series. Figure 7.11 shows

an example prediction by intersection method. Here, the company had a rating change at 2017-02-06 from 'B' to 'BB'. Another change occured at '2017-03-27' from 'BB' to 'B'. Finally the rating changed one last time from 'B' to 'BB'. If we observe the figure, we see that jumps coincides the rating changes. Even though the actual time series has an increasing trend on the given days, the change is much smooth compared to the prediction.

Figure 7.11: A time series constructed by intersection method



As explained at the study done by Nomura [9], another important flaw of intersection method is that time series representing each buckets has sharp changes. For example when a company joins or leaves a bucket, than the mean spread of the bucket that it belongs to, jumps drastically. To observe this paradigm, the time series representing mean spread for the bucket (Finance, North Amer., 'B') is shown at the Figure 7.12.

Figure 7.12: Average spread for the bucket (Finance, North Amer., 'BB')

The figure shows some sudden jumps on the average spread time series for the bucket. This is because ratings change for training companies as well. For example the company 'ALLYFI' had changed buckets on 2017-02-06 which changed the mean in both its new and old bucket.

The performance of different methods is shown at the Table 7.4. As seen intersection method is strong in terms of having small mean squared error however it is prone to steep changes.

7.2.2 Cross-section method

Cross-section method is another method which is commonly used in financial institutions for Credit Default Swap approximation problem. Here, a linear regression model is trained daily using the FS1 feature as explained at Section 3.

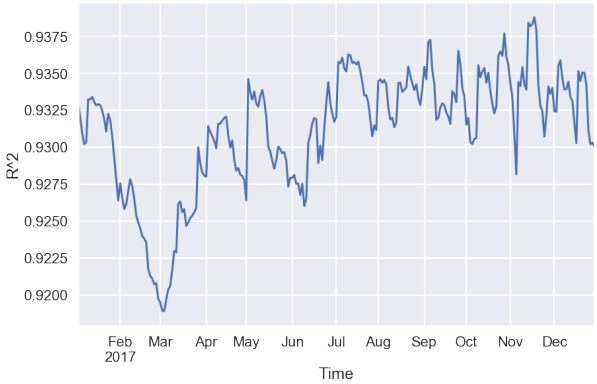
Since, an independent model is trained every single day, linear regression coefficients also change daily. Figure 7.13 shows the change in parameters representing, ratings, region and the constant(global mean).

Figure 7.13: Change in coefficient representing the rating, region and global

As explained at Section 3, according to Cross-Section method, CDS spreads are predicted by multiplying the coefficients seen in Figure 7.13.

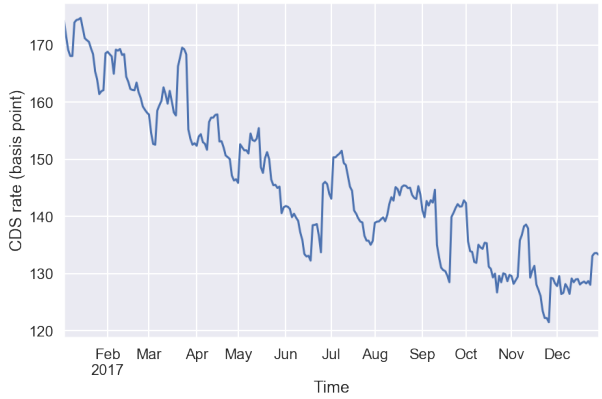
Even though, Cross-Section method is a simple linear regression in its essence, its performance is actually good. This can be seen on the variance captured by the regression. For showing this, daily training R-squared looks as the Figure 7.16.

Figure 7.14: Daily R-squared at 2017



Finally for a comparison between Intersection method and Cross-Section method, the time series for (Finance, North Amer., 'BB') predicted by Cross-Section is shown at Figure 7.15. Here it is seen that Cross-Section is more stable than Intersection.

Figure 7.15: (Finance, North Amer., 'BB')

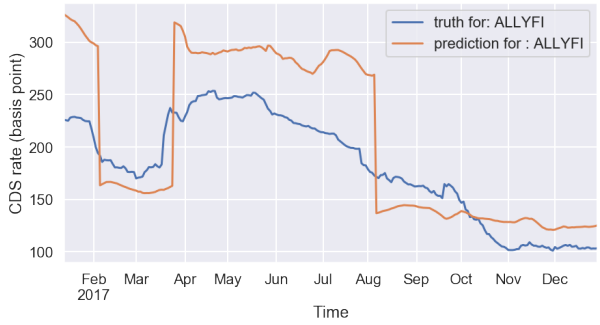


7.2.3 Regression using FS2

As explained before Cross-section method only uses FS1 feature. However, it actually very easy to implement Cross-Section logic by FS2 and FS3 feature sets that were used for classification experiments. In this section we will describe FS2 regression. Doing regression with FS2 is exactly the same procedure with Cross-Section method. Only difference is adding more information (i.e stock variances). However, by adding counter-party specific information, we hoped to capture counterparty specific information.

As seen at Table 7.4, when the experiments were run, it was observed that the results are actually very similar to the Cross-Section method. Even though a slight decrease in mean squared error was obtained, it was not possible to obtain significant improvement. This is because even though stock variances are used, linear regression is still dominated by the FS1 fields.

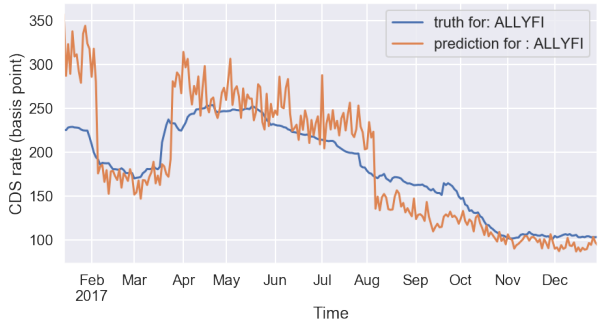
Figure 7.16: An example prediction by using FS2 in linear regression



7.2.4 Regression using FS3

As mentioned at the earlier section, the performance of linear regression using FS2 could not create a big improvement. The next step was using raw stock price returns with FS3. For this experiment we have run linear regression similar the way we have done for FS2 vector. This model significantly improved mean square error in 5 fold Cross Validation.

Figure 7.17: An example prediction by using FS3 linear regression



Performance obtained by all the different models can be seen at Table 7.4.

Table 7.4: Performances of the regressions and feature sets

Name	Feature	Type	MSE	Avg. Corr.
Intersection Method	FS1	In Buc.	10.9	0.62
Cross-section Method	FS1	Out Buc.	5.8	0.65
Linear Regression(+PCA)	FS2	Out Buc.	5.7	0.65
Linear Regression(+PCA)	FS3	Out Buc.	4.8	0.67

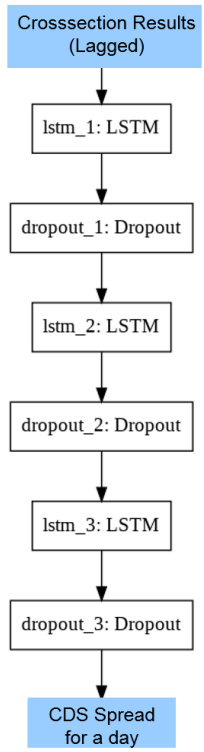
7.3 Deep Learning experiments

In the sections above we have analyzed many different Machine Learning models. In those models we had to fit a regressor or a classifier on daily, weekly or monthly basis because the market conditions are changing and it is not possible to capture these changes by one linear regressor or classifier some of whom only has hundreds of parameters. Moreover all the models that we had tried suffered from the same problem. When the company changes buckets, prediction for that company changes too much compared to the real. Therefore, we decided to experiment deep neural networks solutions that has the capacity to capture smooth changes and longer periods.

7.3.1 Deep Learning Model 1: LSTM + Cross-Section

If you observe previous figures, you can directly see that all the different models creates jump overshoots whenever the input company change ratings. Even though very simple models such as Cross-Section Method performs well in terms of having low mean squared error, constructed time series never looks natural because of all the jumps. This is because all the models are run daily and they don't infer from previous predictions. This is the reason why Recurrent Neural Network models are promising.

Figure 7.18: Model plot of the LSTM network



In the first Deep Learning experiment, we have feeded LSTM network with the predictions obtained by Cross-Section method. By using these predictions as input, LSTM already have a good baseline but only needs to learn the transition between ratings. Figure 7.18 shows the general flow for the model.

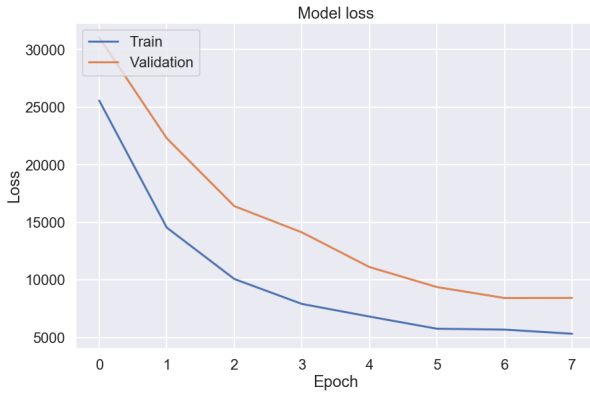
After the experiments we have seen that by using relatively simple architectures, it was possible to improve Cross-Section model. In this experiment our model has 4 LSTM layers and 1 dense layer. For regularization purposes Dropout layers were included between all layers. The summary of the model architecture can be seen at Figure 7.19 .

Figure 7.19: LSTM model architecture

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 21, 50)            10400
_________________________________________________________________
dropout_1 (Dropout)          (None, 21, 50)            0
_________________________________________________________________
lstm_2 (LSTM)                (None, 21, 50)            20200
_________________________________________________________________
dropout_2 (Dropout)          (None, 21, 50)            0
_________________________________________________________________
lstm_3 (LSTM)                (None, 50)                20200
_________________________________________________________________
dropout_3 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 51
=================================================================
Total params: 50,851
Trainable params: 50,851
Non-trainable params: 0
_________________________________________________________________
```

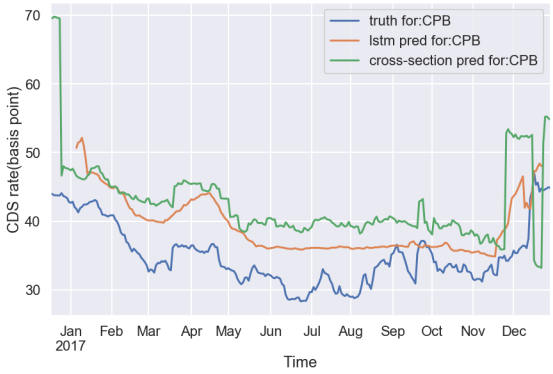
Since the input of the model is already predictions, the model starts with a lower loss and converges very quickly. Training loss and validation loss in the first fold of the cross validation can be seen at Figure 7.20.

Figure 7.20: LSTM training and validation loss per epoch

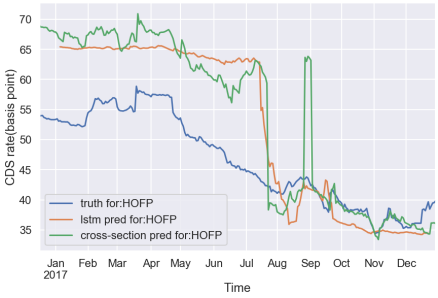


The comparison of Cross-Section predictions and LSTM predictions are done at Figure 7.21.

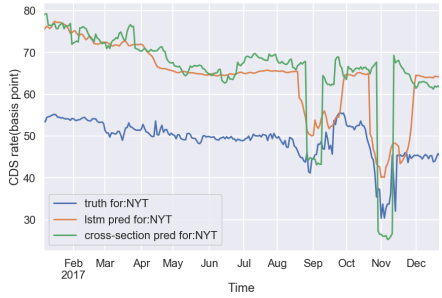
Figure 7.21: LSTM predictions



(a)



(b)



(c)

Since the model architecture is simple, training time was actually manageable for comparing in 5 fold cross validation. Table 7.5, shows the performance for this model. There is gain in mean squared error and it is safe to say that this model is outperforming Cross-Section method.

Table 7.5: Performances of the regressions and feature sets

Name	Feature	Type	MSE	Avg. Corr.
LSTM + Cross-section	-	Out Buc.	4.3	0.67

7.3.2 Deep Learning Model 2: Convolutional

In this part, a Convolutional Neural Network architecture is experimented. The idea was to use the spatial relationships among the 1 dimensional sequential data both for stock returns and ratings of one company. We added convolutional layers to extract the features in each of these time series. Later the outputs from both of these convolutinal layers of stock prices and the ratings of each company are added in the latest layer.

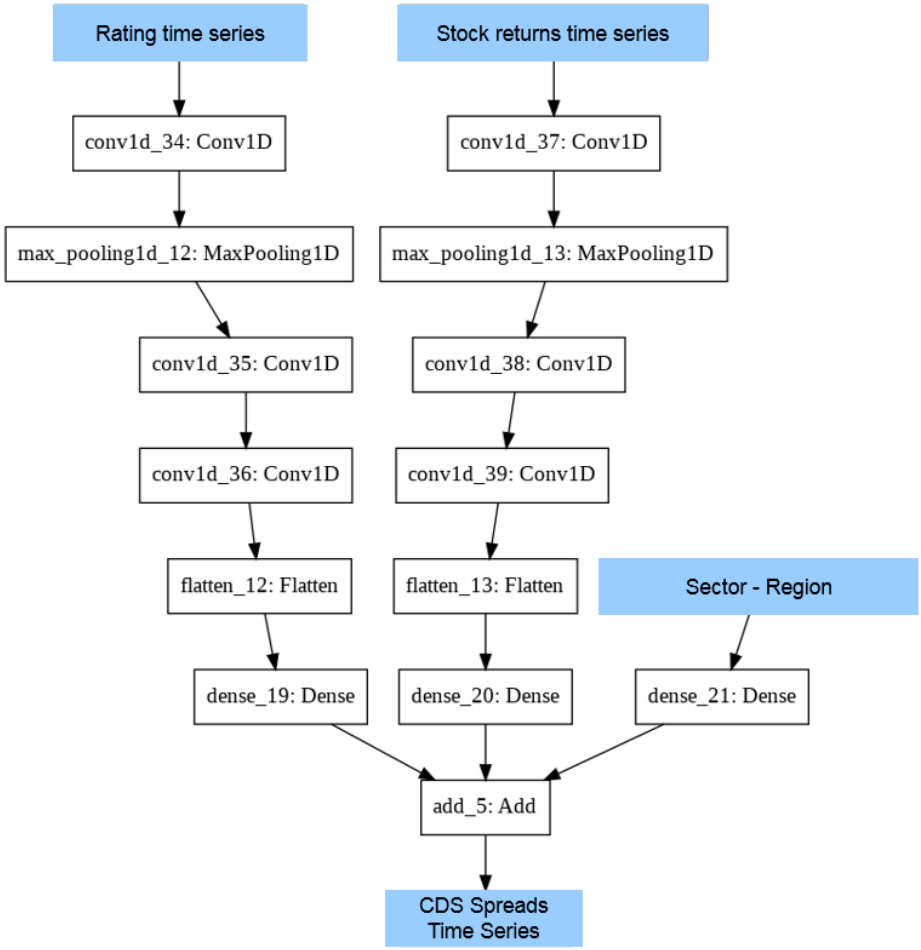
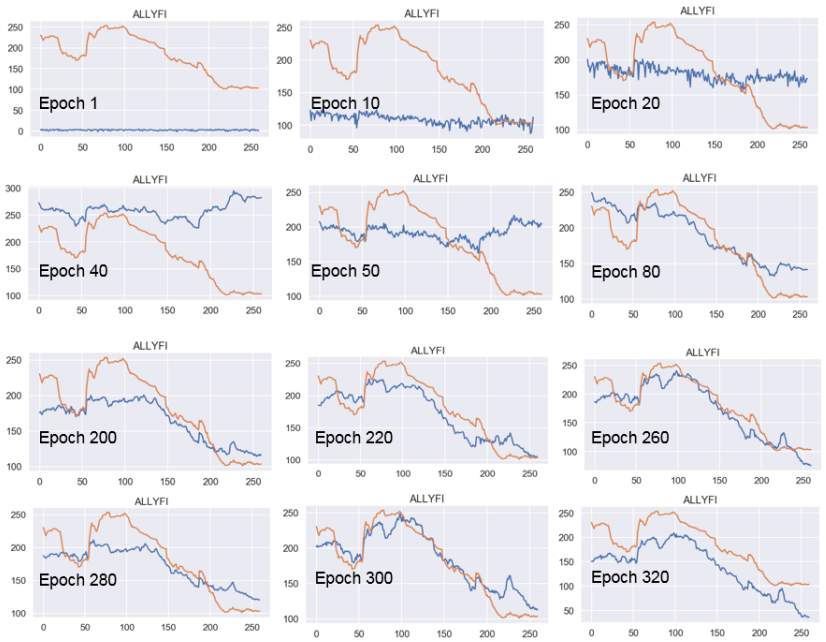


Figure 7.22: Model summary of convolutional neural network design of Deep Learning Model 2

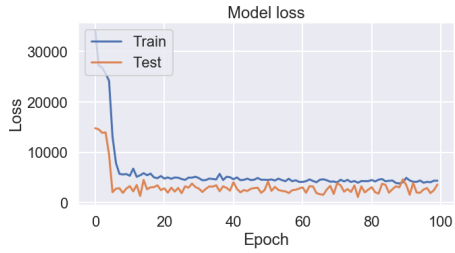
In figure 7.23 we see a qualitative result of the method we used in this experiment. We investigate the same company we used for some other models in this project. The company was taken from the test batch. We can see the progress in the network, epoch after epoch until we reach around epoch 300. Then we observe that the model is getting overfit. The curves are getting further away than each other and MSE is increasing.

Figure 7.23: Predicted and original CDS spreads of company ALLYFI



Training and validation loss for this network are shown at Figure 7.24. The hyperparameters like number of layers, the stride number, filter size are chosen with a trial and error method. The model is run for 320 epochs. Since we have not observed a significant over fitting by training too much, we have not employed regularizations such as dropout.

Figure 7.24: Losses per epoch for convolutional neural network design



Finally, Table 7.6, shows the performance of this model. Note that since training takes too much time, 5-fold Cross Validation could not be applied.

Table 7.6: Performances of the regressions and feature sets

Name	Feature	Type	MSE	Avg. Corr.
Convolutional	-	Out Bucket	6.7	0.62

7.3.3 Deep Learning Model 3: Multiple Input Single Output

In this network, we have decided to feed in all the historical CDS spreads of all the training companies. This is because there may be a relationship between the CDS spread of the company that we are interested in and all the other companies in the world. Therefore, all the other companies' CDS spread can be used as a feature. There can also be a dependency between the ratings yesterday, last week or last month and the CDS spread today. Similarly, we may observe dependency between historical and future stock returns and the CDS spread today. Note that we can actually use the future data for our experiment. This is because what we are doing basically is a historical mapping between historical features to historical CDS spreads. Thus the only information not available is CDS spreads. In figure 7.25 a multiple input single output network design is shown. The inputs we have utilized for this architecture is explained in table 7.7.

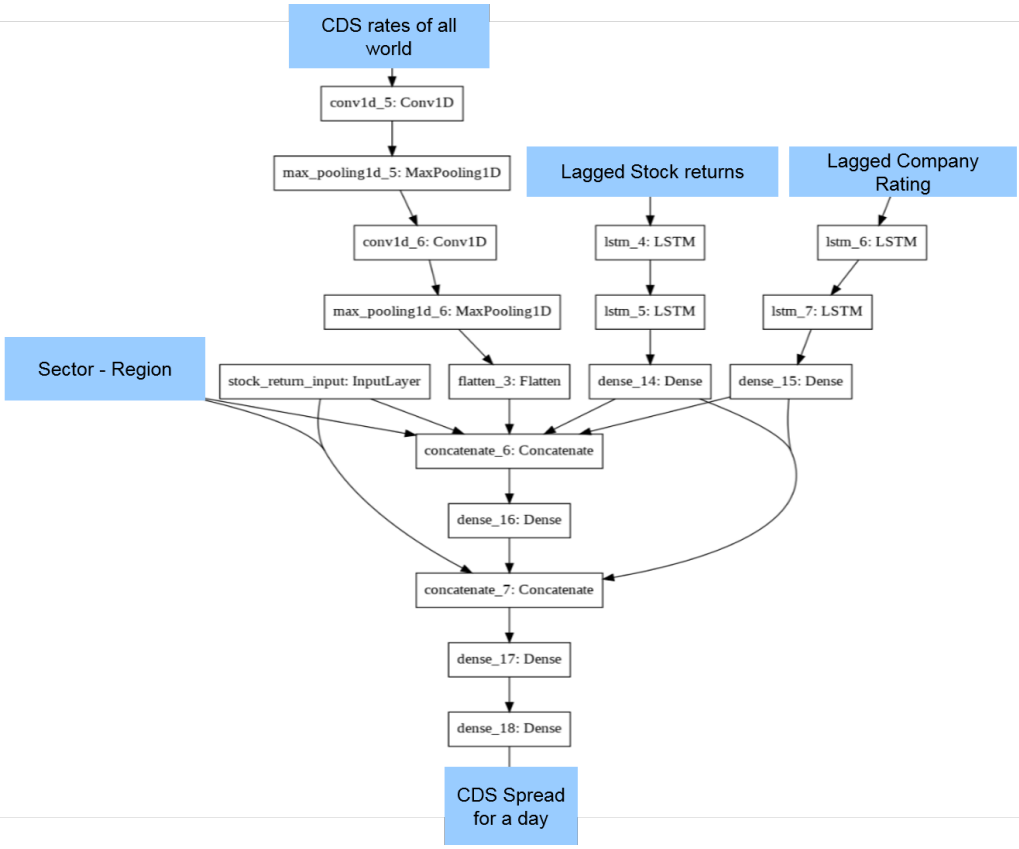


Figure 7.25: Multiple input single output Neural Network Design

	Explanation	Number of dimensions
Sector-region	One hot encoded version of sector and region	10
CDS spreads of all world	All CDS spreads in the world from **training** companies	322
Lagged stock returns	RNNs needs the lagged version of sequential data. Here, not only the past is used but we can use the future stock prices as well since we already know all the historical stock prices. Therefore, they can be defined as $S_{t-10}..S_t..S_{t+10}$.	20
Lagged company rating	RNNs needs the lagged version of sequential data. This is the numerical version of the ratings. AA is 7, A is 6, CCC is 1. Therefore, they can be defined as $R_{t-10}..R_t..R_{t+10}$.	20
'Target', 'y' or 'dependent variable'	The CDS Spread of the company that we are interested	1

Table 7.7: Explanations of the inputs for the architecture in figure 7.25.

The stock prices and the ratings are sequential data. Therefore, we can use them in LSTM part of the network. CDS spreads of all world might include positional relationships between the companies. That is why we decided to utilize convolutional networks in the related part of the network.

Figure 7.26: Mean squared error loss function of the features shown in both figure 7.25 and table 7.7 by using multiple input - single output LSTM-Convolutional combined neural network

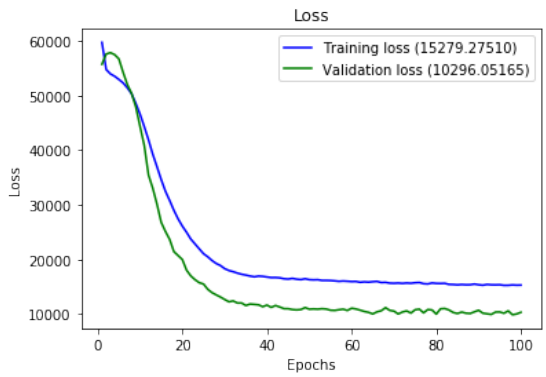


Table 7.8: Performances of the deep neural network regressions and feature sets

Name	Feature	Data	MSE	Avg. Corr.
LSTM + Conv1d	-	In Buc.	11.0	0.61

Since the training was slow enough, we have not employed a regularization technique in this model. The combinations of the layers seen in figure 7.25, were chosen with a trial and error process. As seen from both figure 7.26 and table 7.8 this model did not give a better result than the simpler models. This may be because the data set was big and model was more complex. Therefore, the model needed much more time to be trained on CPU. Even though the input data includes much more information than we need, the model complexity and the hardware limitations does not allow us to capture the patterns we capture with the simpler models.

Chapter 8

Discussion

In this work, we have aimed to leverage Machine Learning methods for solving historical CDS proxy construction problem using publicly available financial data. There are two possible approaches to solve this problem. If we don't have the CDS spread for a company, we could either find a proxy company and assume that CDS for unobservable is close to the proxy company's CDS or we could directly estimate CDS spread with a regression algorithm.

On the **regression** side, currently, the most common method for solving CDS spread construction problem is *Cross-Section method* which is fundamentally a linear regression trained on sector, region and rating fields of a counterparty. Apart from this model, *Intersection method* is also very common. Even though these models are stable and proven to be performing "good enough" [9], using modern Machine Learning methods there is room for improvement as stated by Zhongmin, [7] and by our foundings.

On the **classification** side, to our knowledge, the only Machine Learning based research on our problem was done by Zhongmin [7]. Here authors suggest classification algorithms to construct CDS spread time series. However, they have only used a simple evaluation method by measuring the classification accuracy. In short, they left out some data points belonging to a company and measured how well a classification model could find to which company the data belongs in the first place. Even though authors mentioned that they believe this method

would perform better than Cross-Section method, they have never constructed time series and thus could never compare.

To understand how well classification algorithms perform in comparison with the regression methods, we have started replicating a subset of the methods explained at Zhongmin, [7]. There, the authors engineered some features which makes the data points unique to each company such as stock returns. Following this idea, we have also constructed feature sets using sector, region, rating and historical stock prices for a company. On top of using these fields (FS1), the first feature vector we engineered includes historical stock price return variances (FS2). Moreover, we have also engineered another feature vector that includes directly stock returns (FS3).

After measuring the performances of the models, similar to Zhongmin, [7], we also observed that Decision Tree, KNN and FNN models were very successful in finding to which company a given data point belongs to. At first, it is odd to see that simple models such as KNN could perform over 90% accuracy on a classification problem which includes more than 400 classes. However, when some thought is put on the evaluation method, the case is clarified. It is actually quite normal that feature vectors for a company on different days are pretty close. This is because some of these features are stochastic processes which can be autocorrelated. For example, 1 year variances of the stock returns are highly autocorrelated as seen in figure A.1. Thus, it is easy for KNN to classify the true classes by abusing these autocorrelations in the features. Then, *how frequent should the samplings be in order to avoid the illusion of the good accuracies because of the autocorrelated data?* This seems like a relevant question at a first glance but what we are after is not the accuracies but the MSE errors we obtain using E2 evaluation. Since the test and training companies are separated and the time series are constructed with the training companies, this will not be an issue.

Another question is *Will a model that performed well on classification, beat the regression methods after constructing the time series?* In their work (Zhongmin, [7]), the actual CDS spread proxy time series were never constructed. Therefore the next action for us was creating the time series. For this, we have first simulated the exact environment, where the models would be used. The test period was chosen to be 2017. Using cross validation, in a loop, we assumed we have test companies for which we don't know CDS spreads. Next, we have constructed CDS spread time series by using both classification models and regression models. Finally, we have observed that the classification models that have performed well on the first evaluation could not create good time series. Actually, none of the classification models could beat Cross-Section model. Moreover, only Feed Forward Neural Network classification could perform similar to Intersection method. However, the performance gain is insignificant and FNN is much

slower in terms of training time especially when we think that Intersection and Cross-Section methods are trained for each day's data separately.

After observing classification models, we have concluded that it is really hard to train a classification model which would build time series close to truth. According to our results, after constructing the time series, higher accuracy does not necessarily give a lower MSE. This makes sense since at a given day, there are no two companies that would exactly perform same in CDS market. Lastly, the time series constructed using classifications have a tendency to include significant jumps whenever the predicted proxy changes. This created poor quality graphs. For example, if you observe Figure 7.5 (a), you can see whenever the input company changes ratings, the model also changes prediction that creates jumps in the graph. Even if the input company does not change ratings, the classification models are not stable. This can be best seen at Figure 7.5 (b) where the model changed predictions without a change in rating.

Here, the second phase of the project started where we have searched opportunities in developing regression models for predicting daily CDS spreads. Similar to the classification comparison, we have simulated different models in the year 2017 and compared results with Cross-Section benchmark. Our first attempt to improve current regression methods was to insert extra information. Feature vectors FS2 and FS3 that we have engineered for classification models include extra information on top of the fields used by Cross-Section method. Thus we first run the same model logic with these methods. At the regression experiments carried out with the engineered features, in fact we observed that new models outperformed Cross-Section method. Table 7.4 shows different models compared by 5 fold cross validation (different companies tested at every fold). Here we can see that regression model trained using historical stock return variances(FS2) and daily stock returns(FS3) both beat the current model. Especially, the model trained with FS3 decreased the mean squared error significantly compared to current model which proves that using stock prices as inputs improves CDS spread prediction performance.

Even though the models that we have developed using FS2 and FS3 improves Cross-Section method, all the models still suffer from the same flaw. Whenever a company changes ratings, these models change their predictions significantly. This is because none of those models infer information from previous predictions. Therefore, daily prediction are totally independent. This was the reason why we decided to keep improving our models. In recent years, Deep Learning models, especially Convolutional and Recurrent networks were proved to be performing very good on time series prediction. Since CDS proxy construction problem is a time series prediction problem in its essence, we have implemented 3 different Deep Learning models.

In section 7.3.1, we experimented an LSTM model by using cross-section method's output. Therefore, this is a 2 stage model. This model improved MSE significantly. Also we observed that the predictions were much more stable thanks to the memory unit cells in LSTM. As seen in Figures 7.21, Cross-Section model is vulnerable to changes in buckets when used on its own. Whenever a new counterparty enters or exits the bucket, the prediction for that bucket changes severely and this is even worse in the intersection method. Here it is seen that LSTM model was more resilient in changes. After our attempts, we can safely say that this LSTM model is a direct contribution to the field.

In section 7.3.2, we started experimenting more developed architectures. We have successfully implemented a Deep Learning architecture to capture more complex patterns. As seen in figure 7.3.2, this network maps a company's stock return time series, rating time series and sector-region into its CDS spread time series. In figure 7.23 we observe the improvement in one test company's predictions but after 300^{th} epoch, the quality start to get worse again. We did not get the best MSE score from this model but qualitative results show stable predictions as seen in figure 7.23.

In section 7.3.3, we developed a model where we tried to incorporate the general situation of the world in a given day. Here, the CDS spreads of the training companies were given as a feature. However, the number of dimensions in the input vectors increased. The model had to be more complex to catch the patterns among all the other companies as well. This led to a decrease in performance.

Overall, we get the best performance when we use both cross-section and LSTM at the same time. This model proved to be stable and easy to apply in the industry.

Chapter 9

Conclusion & future work

9.1 Conclusion

In this project, we have implemented 37 different Machine Learning models for constructing historical CDS spread time series. The current methods that are in use in financial sector are relatively simple models and only use little information such as sector, region and rating of a counterparty.

After assigning proxies to the unobservable counterparties by using classification methods, we concluded that it is difficult to construct a CDS spread time series based on the assigned proxy counterparties. We also found that there is an inevitable trade off between catching the trend in the market and stability of the constructed time series.

Secondly, this project investigated the effects of adding extra features extracted from stock market data. Our experiments showed us, adding extra information of historical stock price returns increased performance of currently used Cross-Section model.

Finally, the results of Deep Learning experiments showed that a much greater performance can be obtained. In order to do this, we have proposed a simple LSTM model which is trained on the predictions of Cross-Section method. This

model could outperform current common CDS spread construction methods among all of our experiments.

9.2 Future work

One potential improvement is to have more *meaningful input data*. In this work we have used stock prices for CDS spread estimation because it is easily accessible. However, there are different data sources which could provide even better results such as quarterly P&L reports. We believe that it could be possible to create even better models with this data.

Another way to improve the results would be to have a *better neural network architecture* that has the capacity to capture even more complex patterns. Because of the time and computational power limitations we could not investigate other usages of convolutional networks. However, we consider that especially having other counterparties CDS spread as input to a Recurrent-Convolutional Neural Network is a promising idea according the promising results that we obtained from section 7.3.3.

Appendix A

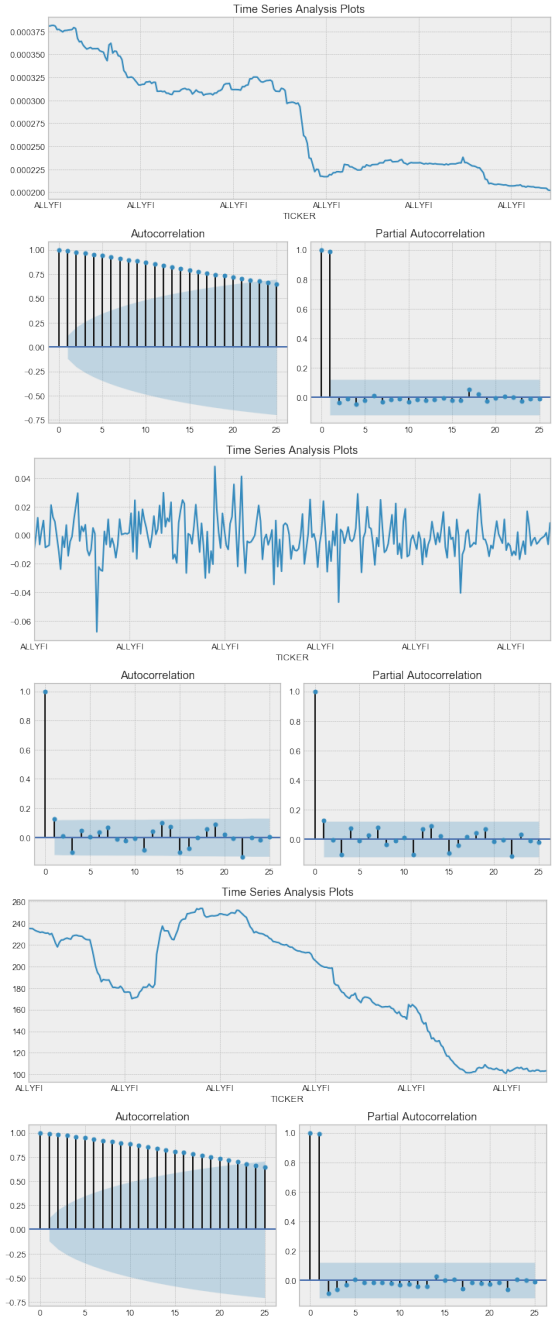
Appendix

A1 Abbreviations

Abbreviations	explanation
FS1	(sector, region, rating)
FS2	(sector, region, rating, $\sigma_{1M}, \sigma_{3M}, \sigma_{4M}, \sigma_{1Y}$)
FS3	(sector, region, rating, stock returns)
E1	Evaluation of the accuracies
E2	Evaluation of the mean squared errors
S1	The raw data source which includes TICKER-DATE-COUNTRY-SECTOR-REGION-RATINGSPREAD6M-SPREAD1Y-SPREAD2Y/3Y/5Y/7Y/10Y/15Y/20Y/30Y
S2	The raw data source which includes TICKER-DATE-CLOSING PRICE-VOLUME
Out-bucket models	Models that are trained by all the companies in the world
In-bucket models	Models that are trained by all the companies in one bucket

A2 Autocorrelations of some features

Figure A.1: Auto correlations of 1Y variance stock returns, daily stock return and CDS spreads of company 'ALLYFI' are shown respectively



A3 Error distribution

Figure A.2: Error distribution for in-bucket FNN classification (FS3)

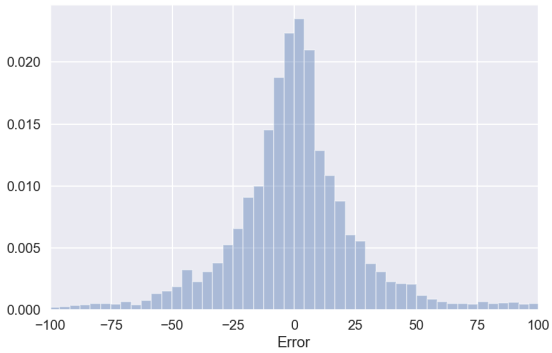


Figure A.3: Error Distribution for out-bucket FNN classification (FS2)

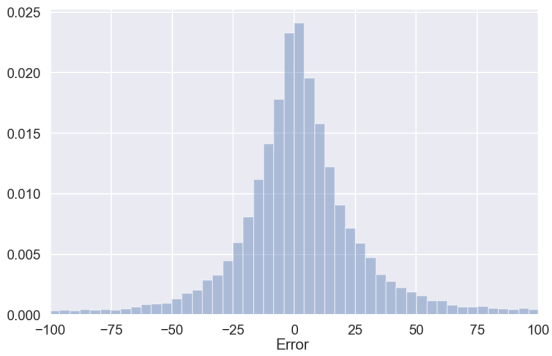


Figure A.4: Error distribution for LSTM + Cross-Section

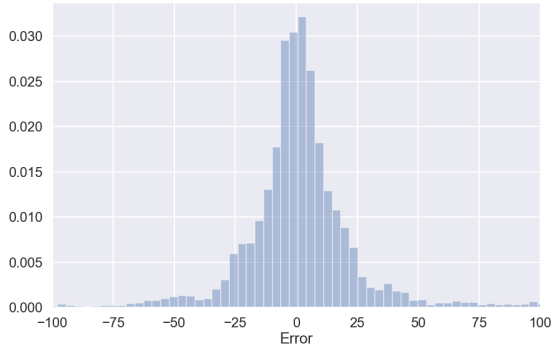


Figure A.5: Error distribution for Cross-Section

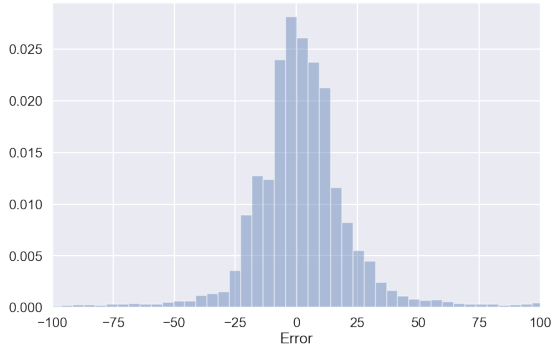


Table A3.1: Performances of the FS3 classifiers using E2 (using 5 fold cross validation)

Name	Feature	Type	MSE(10e3)	Corr.	Pred. Change
Random Classifier	FS1	In Buc.	14.9	0.40	137.2
FNN	FS2	Out Buc.	10.1	0.52	7.3
FNN	FS3	Out Buc.	10.1	0.52	7.3
Intersection Method	FS1	In Buc.	10.9	0.62	-
Cross-section Method	FS1	Out Buc.	5.8	0.65	-
Linear Regression(+PCA)	FS2	Out Buc.	5.7	0.65	-
Linear Regression(+PCA)	FS3	Out Buc.	4.8	0.67	-
LSTM + Cross-Section	FS1	Out Buc.	4.3	0.67	-
Convolutional	-	Out Buc.	6.7	0.62	-
LSTM + Conv1d	-	Out Buc.	11.0	0.51	-

Bibliography

[1] The basel iii accord. `https://basel-iii-accord.com/`. Accessed: 2019-03-13.

[2] Quandl u.s stock prices data.

[3] Yahoo finance, 2018.

[4] N. S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[5] F. J. Anscombe. Graphs in statistical analysis. *The American Statistician*, 27(1):17–21, 1973.

[6] E. B. Authority. Technical standards in relation with credit valuation adjustment, risk. 2013.

[7] Brummelhuis, R. and Z. Luo. Cds rate construction methods by machine learning techniques. 2017.

[8] R. Chalapathy and S. Chawla. Deep Learning for Anomaly Detection: A Survey. *arXiv e-prints*, page arXiv:1901.03407, Jan 2019.

[9] Chourdakis, Kyriakos and Jeannin, MARC and Mcewen, James. A cross-section across cva. 2013.

[10] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals Eugenics*, 7:179–188, 1936.

[11] B. for International Settlements. Cds market size, 2018.

[12] K. P. F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[13] P. Ganesh and P. Rakheja. Deep Neural Networks in High Frequency Trading. *arXiv e-prints*, page arXiv:1809.01506, Sep 2018.

[14] Gillian Tett. The dream machine: invention of credit derivatives. 2006.

[15] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[16] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *ArXiv*, abs/1207.0580, 2012.

[17] y. Hochreiter, S., Schmidhuber, J. Long short-term memory. neural computation.

[18] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[19] A. Karpathy. The unreasonable effectiveness of recurrent neural networks), 2015.

[20] LEANDRO S. MACIEL, ROSANGELA BALLINI. Design a neural network for time series financial forecasting: Accuracy and robustness analisys. 2010.

[21] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521:436–44, 05 2015.

[22] Z. Liang, H. Chen, J. Zhu, K. Jiang, and Y. Li. Adversarial Deep Reinforcement Learning in Portfolio Management. *arXiv e-prints*, page arXiv:1808.09940, Aug 2018.

[23] C. M. Lior Rokach. *Data Mining with Decision Trees*. 2014.

[24] H. Liu. Leveraging Financial News for Stock Trend Prediction with Attention-Based Recurrent Neural Network. *arXiv e-prints*, page arXiv:1811.06173, Nov 2018.

[25] M. LLC. Perceptron: The artificial neuron (an essential upgrade to the mcculloch-pitts neuron), 2018.

[26] M. LLC. Plot neural network layout, 2018.

[27] MathWorks. Counterparty credit risk and cva, 2019.

[28] K. N. S. S. R. Merity, S. Regularizing and optimizing lstm language models. 2017.

[29] C. Olah. Understanding lstm networks, 2015.

[30] M. Pykhtin and S. Zhu. A guide to modeling counterparty credit risk. July/August 2007.

[31] B. Raunig and M. Scheicher. *A Value At Risk Analysis of Credit Default Swaps.* European Central Bank.

[32] Richard White. The pricing and risk management of credit default swaps with a focus on the isda model. 2013.

[33] F. F. S.M. Focardi. The mathematics of financial modeling and investment management. 1997.

[34] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[35] R. O. . E. Sundberg. A study of the basel iii cva formula, 2017.

[36] M. N. S. Tue Herlau and M. Mørup. *The Problematic Case of Clearinghouses in Complex Markets.* Georgetown Law Journal, Vol. 101, 2013.

[37] M. N. S. Tue Herlau and M. Mørup. *Introduction to Machine Learning and Data Mining.* Technical University of Denmark, 2018.

[38] S. University. Cs231: Convolutional neural networks for visual recognition, year = 2019, url = http://cs231n.github.io/convolutional-networks/, urldate = 2019-06-25.

[39] Wikipedia. Credit valuation adjustment, 2018.