Fys-Stk4155 Prosjekt3

Thomas Bergheim
Vidar Lunde Olaisen

December 2021

Contents

1 Abstract **iii**

2 Introduction **iii**

3 Theory **iii**

3.1 Classification problem . . . iii

3.1.1 Linear Regression . . . iii

3.1.2 Logistic Regression . . . iii

3.1.3 Neural Network . . . iv

3.1.4 Convolutional Neural Network . . . iv

3.1.5 Recurrent Neural Network . . . iv

3.1.6 Support Vector Machines . . . iv

3.1.7 Decision Trees . . . v

3.1.8 Random Forests . . . v

3.2 Estimates . . . v

3.2.1 R2-score . . . v

3.2.2 Mean Squared Error . . . v

3.2.3 Confusion matrix . . . vi

3.2.4 Accuracy score . . . vi

3.2.5 Information gain . . . vi

3.2.6 ROC . . . vi

3.2.7 Cumulative gains curve . . . vi

3.3 Training . . . vi

3.3.1 Number of features . . . vi

4 Methods **vi**

4.1 Variable selection . . . vi

4.1.1 Correlation . . . vii

4.1.2 Principal Analysis Component (PCA) . . . vii

4.1.3 Backward Elimination . . . viii

4.2 Neural Network . . . viii

4.3 k-Nearest Neighbors . . . viii

4.4 Decision trees . . . ix

5 Results **ix**

5.1 Neural Network . . . x

5.1.1 Removed variable . . . xii

5.2 k-Nearest Neighbors . . . xiii

5.2.1 Removed variables . . . xiv

5.3 Decision Tree . . . xv

5.3.1 Removed variables . . . xvi

5.4 Linear Regression . . . xvi

5.4.1 OLS . . . xvi

5.4.2 Ridge . . . xvi

5.4.3 Lasso . . . xvii

5.5 Bonus task . . . xvii

5.5.1 OLS . . . xvii

5.5.2 Ridge . . . xviii

5.5.3 Lasso . . . xix

6 Conclusions **xx**

1 Abstract

In this project we tried to find the best model for estiamting if a company is bankrupt or not from a dataset which is heavily skewed into containing a lot of samples that are not bankrupt. These samples are from https://www.kaggle.com/fedesoriano/company-bankruptcy-prediction and are sampled from companies in Taiwan between 1999-2009. We tried Linear regression, Neural Network, k-Nearest Neighbors and Decision trees, and we found that the k-Nearest Neighbors gives us the best usable model, it's also the easiest to understand for the user. We also tried to remove variables to see if that helps with the estimation, but we didn't get results that satisfied removed them. The estimates we used was MSE, R2, confusion matrix and accuracy.

2 Introduction

From the day Thomas was born, and maybe even before, there's always been transactions between individuals and groups. This can range from buying an ice cream from an ice cream truck to buying answers to an exam from a shady dealer online. These transactions can take place in many shapes and forms, but as we, as a society, has progressed, so has the refinement om these actions. We've gone from hand to hand transactions, towards more of an company to company transaction. Like when paying with bank card, there is a transaction between your bank and the sales company.

This has led us to an everyday where we're more and more reliant on companies to function properly, either with providing information, different types of wares or even loans. As such it's become more and more of a problem when companies are going bankrupt as we've becoming more and more addicted to them working.

Therefore we're going to go through in this report 95 different features of different companies, and see which ones of these we most effectively can use to estimate the probability of bankruptcy.

We will go through some theory of different ways to attack a classification problem, then we're going to go through our methods. Lastly we're going to review our results and come to a conclusion.

3 Theory

3.1 Classification problem

This whole problem is a classification problem, in other words the solution can be one of given classes, in our case bankrupt or not bankrupt, another classification example can be the grades we can get in this course, either A or B. This means that there are certain ways of progressing that makes more sense than others. Here are a few algorithms that can be used to estimate models.

3.1.1 Linear Regression

Linear regression isn't to great of a predictor as it gives us a predicted value that is continuous. It can therefore have values all over the place depending on the explanatory variable. It can work on the other hand by making all numbers higher than 1 equal to 1 and all lower than 0 be equal to 0.

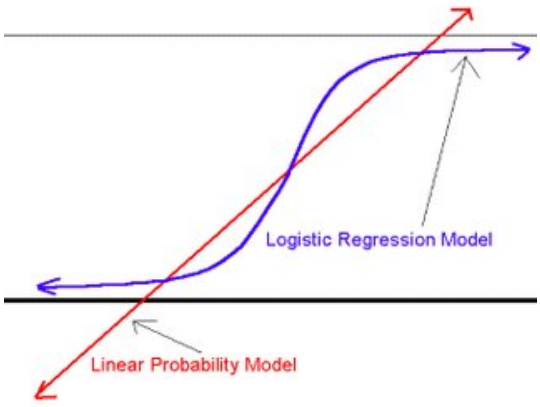

Figure 1: Linear vs Logistic Regression

3.1.2 Logistic Regression

Logistic regression creates a model where the explanatory variables are between 0 and 1 au-

tomatically thanks to the logistic function. It can also present imbalanced data better

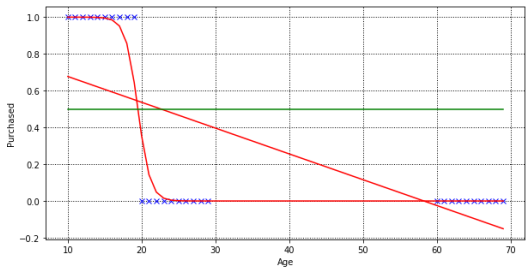

Figure 2: Linear vs Logistic Regression

3.1.3 Neural Network

Neural Networks work great at selecting a class for a future sample as it's basically a graph of connected perceptions. We can then easily use different functions to adapt the Neural Network into accomplishing what we want. We therefore can also specialise it in classification problems by using the Sigmoid function.

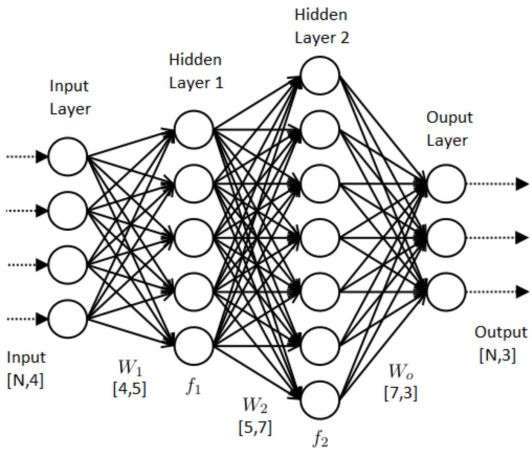

Figure 3: Neural Network

3.1.4 Convolutional Neural Network

While it's a Neural Network it's most commonly used to analyze visual imagery. The difference is that it in at least one layer computer convultional instead of matrix multiplication.

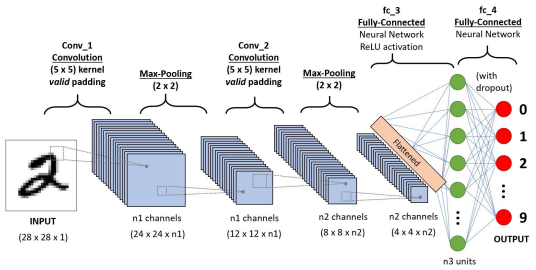

Figure 4: Convolutional Neural Network

3.1.5 Recurrent Neural Network

Recurrent Neural Network is more of a neural network used when memory is important. It can then be used for sentence building or image recognition. It uses the output of the last step to affect the current step.

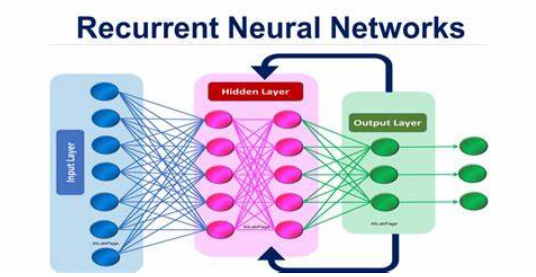

Figure 5: Recurrent Neural Network

3.1.6 Support Vector Machines

Support Vector Machines(SVM) is used for both regression and classification with the help of a linear classifier. It uses hyperplanes to find the seperator of the classes. It can then be used for all kinds of classification problems.

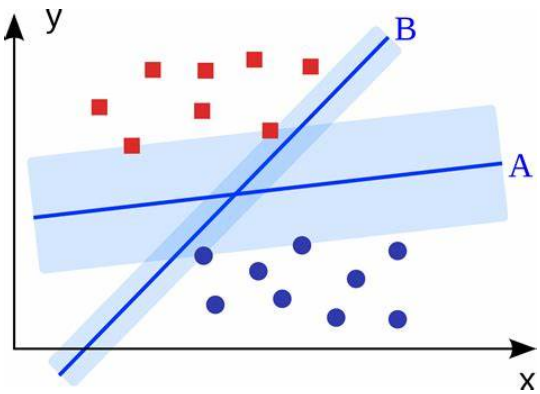

Figure 6: 6599 not bankrupt and 220 bankrupt

3.1.7 Decision Trees

A decision tree is a visual and algorithmic tool used to estimate values of samples. It does this by checking 1 and 1 value as one progresses down the tree to a leaf node where it's decided what class one belongs too. These are very relevant for classification problems as it can't estimate all the problems for a regression problem.

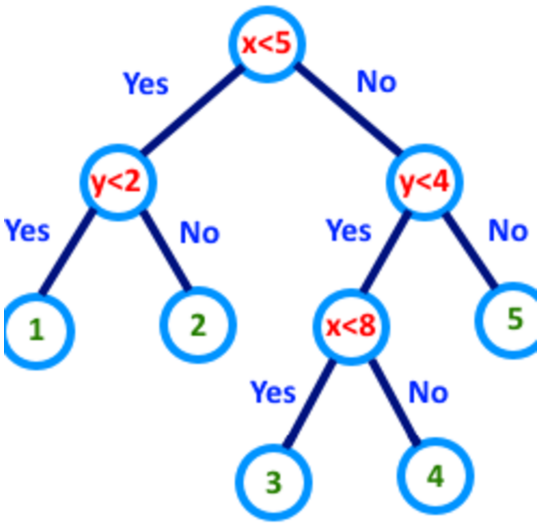

Figure 7: 6599 not bankrupt and 220 bankrupt

3.1.8 Random Forests

Random Forest are a number of different decision trees whos results are then stored, and the final result for the random forest is then the result which is repeated the most times.

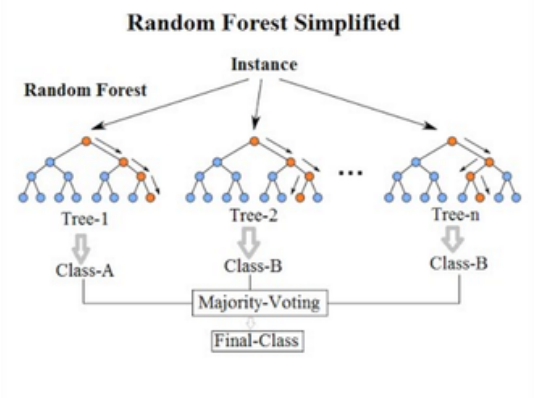

Figure 8: 6599 not bankrupt and 220 bankrupt

3.2 Estimates

Vi har gjort R2-score, MSE, Confusion matrix og accuracy score

There are several ways to see if our model is working well or not. As in project 1 we will look into some of the well known methods as R2-score, MSE, accuracy and others.

3.2.1 R2-score

The R2-score computes the coefficient of determination. The function will give us a measure of how well our future data are likely to be predicted. The score can never be better than 1.0, which will be a 100% prediction, but it can turn negative as the model can become worse.

3.2.2 Mean Squared Error

Another method to measure how well our model succeeded is to calculate the cost function also known as mean-square-error (MSE). The goal is to minimize the cost function as the MSE are the squared errors from the output values and predicted values. Minimizing this score is one of the important part while developing and researching our models.

3.2.3 Confusion matrix

The confusion matrix gives us a matrix of the complete performance of out model. As for a classification problem that is binary, we have the two classes. It can be 0, 1, true, fals, yes, no and so on. This will result in a 2x2 matrix of scores with true positive, true negative, false positive and false negative.

3.2.4 Accuracy score

The accuracy will tell us how many correct prediction our model made. The sum of right predictions are divided by the number of possible data points to provide a accuracy score. The accuracy score can be directly interpreted as how well the model predicted the class belonging.

3.2.5 Information gain

Information gain measures how well a given attribute separates the training examples according to their target classification. This method is relevant in selecting paths in a decision tree.

3.2.6 ROC

The receiver operating characteristic curve (ROC) is used to display the diagnostic ability of a binary classifier system as its discrimination threshold is varied. It plots the true positive rate against the false positive rate.

3.2.7 Cumulative gains curve

The cumulative gains curve shows the percentage of the overall number of cases in a given category gained by targeting a percentage of the total number of cases.

3.3 Training

When training a model to hopefully be accurate in predicting if a company is bankrupt or not, one has to take some decisions that will affect the model. These decisions can range from the ratio of training set and test set to the value of eta(learning steps size). The reason one has to take these decisions is because they don't have a certain answer we can use in all situations, and therefore the values has to be selected individually according to our own understanding of whats the most correct value.

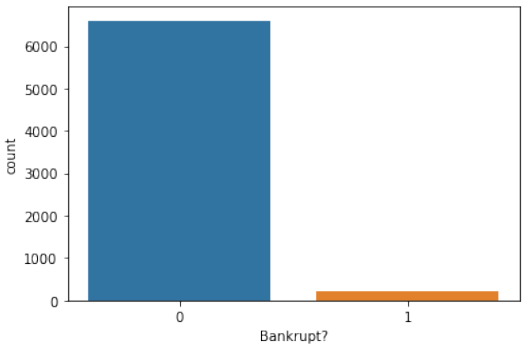

Figure 9: 6599 not bankrupt and 220 bankrupt

3.3.1 Number of features

The data we've gathered contains 6819 samples where each ones contain 95 features. With so many features, there are a few questions we have to answer. Do we overfit the model by keeping all the features? Do they all affect the model enough to keep? The any of the features correlate to a high degree?

4 Methods

4.1 Variable selection

Having many features collected in the data can both be a blessing and a curse. While it most often gives us a more flexible model that's more accurate, it also can make the model over complex. This can especially hurt when trying to get an overview of the effect the different features has, either with a plot or with numbers. An over complex model can also be over fitted to the training data and therefore provide a worse model than one with the same data but fewer features.

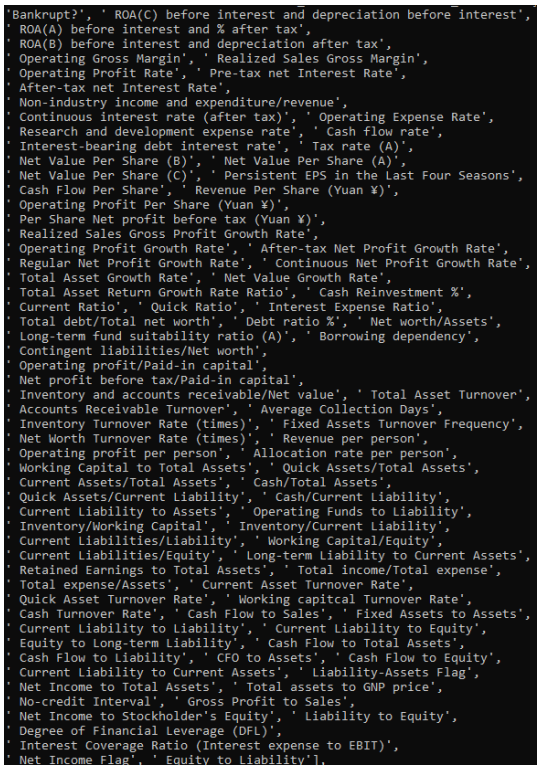

Figure 10: All the features of the data, including the response feature

4.1.1 Correlation

One way of removing correlation and therefore potentially removing overfitting can be Correlation. By using a correlation algorithm, we find the features that provide similar information and remove one of them. When we're doing this, we keep most of the information provided by the different features, while removing unimportant features. This is however a tradeoff where we will most likely lose some information that can be vital later.

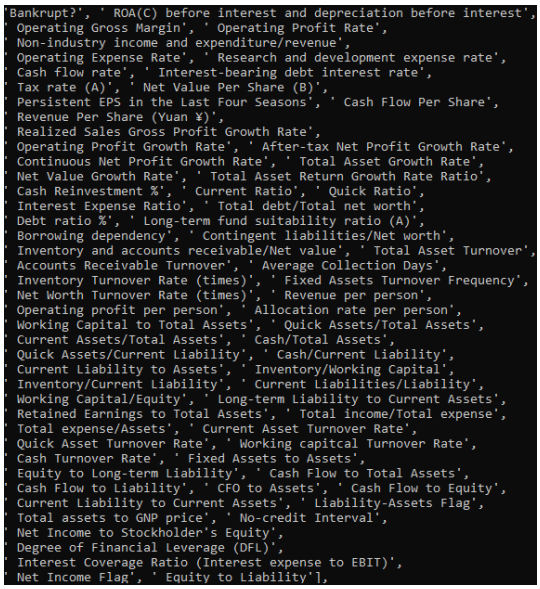

Figure 11: All the features of the data after doing Correlation

Figure 12: The shape of the data

4.1.2 Principal Analysis Component (PCA)

Principal Analysis Component(PCA) is an algorithm that tries to create new features based on the old ones. This is a very risky approach to fix the problem of to many features as it outright removes the original features to create the number of wanted features. The wanted number of new features can be decided by the user, and one does not simply know the right amount without experimenting.

Figure 13: All the features of the data after doing PCA

Figure 14: The shape of the data

Converting our data to PCA with two features gives usgives us

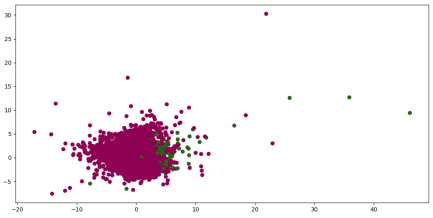

Figure 15: The plot of the data after PCA

```
[0.13346977 0.07213753]
```

Figure 16: Accuracy of the two features

4.1.3 Backward Elimination

Backward Elimination is probably one of the more casual and normally used ways of reducing features. With this one we remove one variable at the time according to it's affect on the accuracy of the model on the test data. We're doing this with the help of something called the p-value. The p-value is a variable that calculates the importance of different features in a model where the higher value the less importance it has. We then remove one, compute a new model and the repeat until each of the features left has a p-value lower than the value the user decides.

```
'Bankrupt?', ' ROA(C) before interest and depreciation before interest',
' ROA(A) before interest and % after tax',
' ROA(B) before interest and depreciation after tax',
' Pre-tax net Interest Rate', ' After-tax net Interest Rate',
' Tax rate (A)', ' Net Value Per Share (A)', ' Net Value Per Share (C)',
' Revenue Per Share (Yuan ¥)', ' Operating Profit Per Share (Yuan ¥)',
' Debt ratio %', ' Net worth/Assets', ' Borrowing dependency',
' Contingent liabilities/Net worth',
' Net profit before tax/Paid-in capital', ' Total Asset Turnover',
' Accounts Receivable Turnover', ' Fixed Assets Turnover Frequency',
' Revenue per person', ' Operating profit per person',
' Current Assets/Total Assets', ' Cash/Current Liability',
' Current Liability to Assets', ' Operating Funds to Liability',
' Current Liabilities/Liability', ' Working Capital/Equity',
' Current Liabilities/Equity', ' Retained Earnings to Total Assets',
' Working capitcal Turnover Rate', ' Cash Turnover Rate',
' Fixed Assets to Assets', ' Current Liability to Liability',
' Current Liability to Equity', ' Cash Flow to Equity',
' Current Liability to Current Assets', ' Liability-Assets Flag',
' Net Income to Total Assets', ' Net Income to Stockholder's Equity',
' Net Income Flag', ' Equity to Liability'],
```

Figure 17: All the features of the data after doing BE

```
New amount of features:  41 , Shape:  (6819, 41)
```

Figure 18: The shape of the data

4.2 Neural Network

Feed Forward Neural Networks(FFNN) with back propagation, also called Neural Network, is a machine learning algorithm that vaguely tries to reassemble how the neurons in the brain work together. By working together in the same way as neurons the algorithm is trying to evolve into the best fitting model.**[1]**

The structure of the NN contains several layers and it's algorithm is built up of 2 processes. **[1]**

For the structure of the FFNN is the input layer, output layer and in between one or several hidden layers depending on the complexity the programmer wants. For the hidden layers and output layers we add weights and bias to tune the network.**[1]**

The first processes is forward propagation, which calculates the values according to the different weights, biases and the input. While the second process is backward propagation, here we calculate the error of the model, in other words how correctly it is in guessing the output of the input. We also update the weights and the biases in an effort of trying to decrease the error of the model. **[1]**

4.3 k-Nearest Neighbors

k-Nearest Neighbors is an algorithm created for selecting classes based on the proximity of nearby data samples. It does this by using a distance equation to find the closest ones, these can be many and can depend on the type of problem. For regression problems one might use Euclidean distance

$$1 \text{ dim} \longrightarrow \sqrt{(x_1 - x_2)^2}$$
$$2 \text{ dim} \longrightarrow \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$
$$3+ \text{ dim} \longrightarrow \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + ...}$$

Another way is Hamming distance for classification problems where it counts the total difference between two same length problems bit

for bit

"Sw**aa**ag" against "Sw**ee**ag" = 2
"**Thug**life" against "**Simp**life" = 4
"Yol**o**" against "Yol**t**" = 1
"**Tupac**" against "**Jesus**" = 5(Some will say 0)

It's normal for the k to be an odd number as it hinders the problem of having an equal amount of point from the classes closest to the new sample. Also the larger the number the lower is the chance for noise to occur.

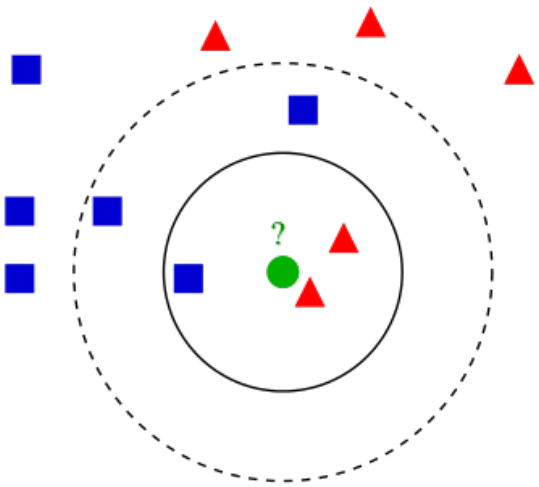

Figure 19: k nearest neighbors 2d plot

4.4 Decision trees

A decision tree is a model that is built up by several layers where each layer is made up of either a decision node, chance node or an end node.

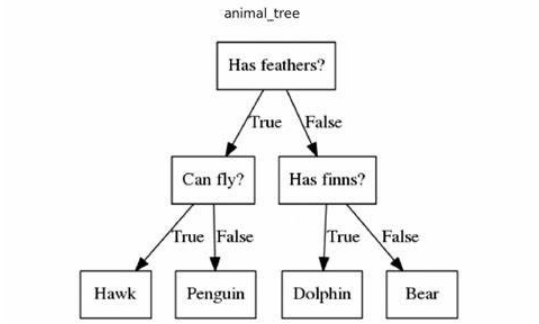

Figure 20: The shape of the data

It quite commonly used because of it's readability. Where other algorithms has to plot in higher dimensions when given many variables, decision trees only has to add more nodes. It's therefore quite simple for a new sample to be decided by following the tree down to it's root according to what values one has.

They are however quite hardcoded from the traininsample, therefore a small change in the trainingsample will change the whole model. It's also not to accurate compared to other predictors.

5 Results

The default value of overfitting with every company assigned not bankrupt we would get an accuracy of 0,967737205 $\sim$ 96.77%. We will use this as an startingpoint in seeing if our models improve or not. If every company was given the class bankrupt we would get 0,0322627951$\sim$ 3.23%. This is accuracy on the whole set.

It should be said that even if the model has worse accuracy than just saying that all the companies are up and running the model might still be an improvement. As we're training on a trainingset and testing on a testset it's more accurate if we have more equal size for class 0(not bankrupt) and 1(bankrupt)

We tried 6 different algorithms to get a model that could be used to predict bankrupts, these were

- Neural Network
- k-Nearest Neighbor
- Decision Tree Linear Regression
- Ridge Regression
- Lasso Regression

and we used 4 different estimations

- Accuracy score
- MSE
- R2 score
- Confusion Matrix

We used Backward Elimination as feature reduction to see if it improves as it's the most common type.

5.1 Neural Network

```
Batchsize: 10 Eta: 0.1
Accuracy: 0.969208211143695

Batchsize: 50 Eta: 0.1
Accuracy: 0.969208211143695

Batchsize: 100 Eta: 0.1
Accuracy: 0.969208211143695

Batchsize: 500 Eta: 0.1
Accuracy: 0.969208211143695

Batchsize: 10 Eta: 0.01
Accuracy: 0.969208211143695

Batchsize: 50 Eta: 0.01
Accuracy: 0.969208211143695

Batchsize: 100 Eta: 0.01
Accuracy: 0.969208211143695

Batchsize: 500 Eta: 0.01
Accuracy: 0.969208211143695

Batchsize: 10 Eta: 0.001
Accuracy: 0.969208211143695

Batchsize: 50 Eta: 0.001
Accuracy: 0.969208211143695

Batchsize: 100 Eta: 0.001
Accuracy: 0.969208211143695

Batchsize: 500 Eta: 0.001
Accuracy: 0.969208211143695
```

Figure 21: The Accuracy of Neural Network

The Accuracy of the neural network stayed at 97% for every size of Eta, and Batch size. This is pretty close to the same result as just guessing every sample ain't bankrupt.

```
Batchsize: 10 Eta: 0.1
MSE: 0.030791788856304986

Batchsize: 50 Eta: 0.1
MSE: 0.030791788856304986

Batchsize: 100 Eta: 0.1
MSE: 0.030791788856304986

Batchsize: 500 Eta: 0.1
MSE: 0.030791788856304986

Batchsize: 10 Eta: 0.01
MSE: 0.030791788856304986

Batchsize: 50 Eta: 0.01
MSE: 0.030791788856304986

Batchsize: 100 Eta: 0.01
MSE: 0.030791788856304986

Batchsize: 500 Eta: 0.01
MSE: 0.030791788856304986

Batchsize: 10 Eta: 0.001
MSE: 0.030791788856304986

Batchsize: 50 Eta: 0.001
MSE: 0.030791788856304986

Batchsize: 100 Eta: 0.001
MSE: 0.030791788856304986

Batchsize: 500 Eta: 0.001
MSE: 0.030791788856304986
```

Figure 22: The MSE of Neural Network

We stumble upon the same problem here as with the accuracy, that the value stays the same at 0.031.

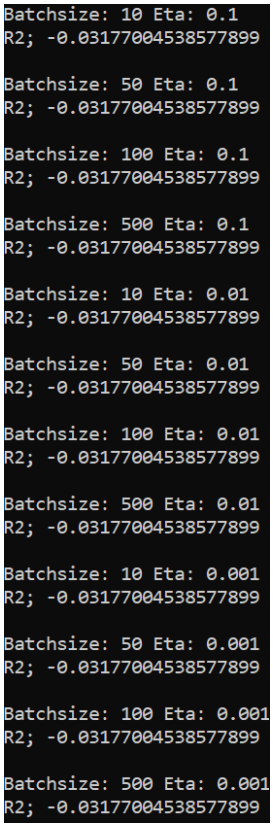

Figure 23: The R2 of Neural Network

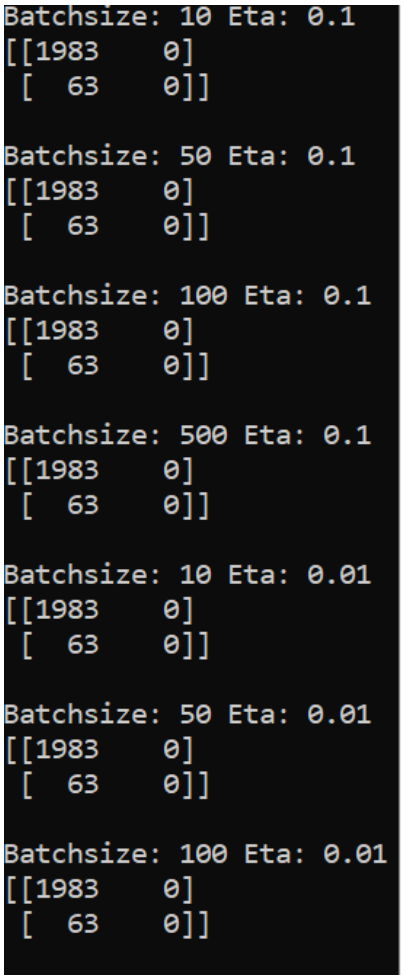

Figure 24: The Confusion Matrix of Neural Network

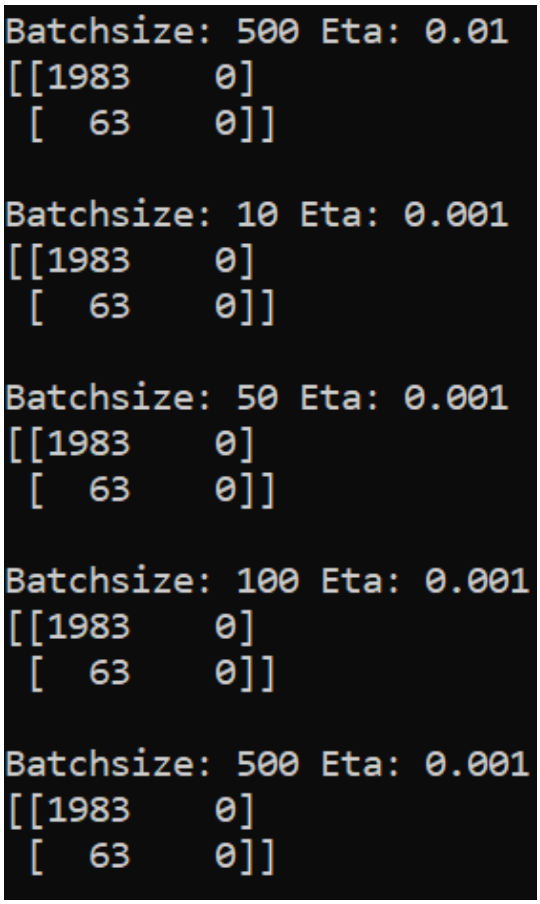

Figure 25: The Confusion Matrix of Neural Network

5.1.1 Removed variable

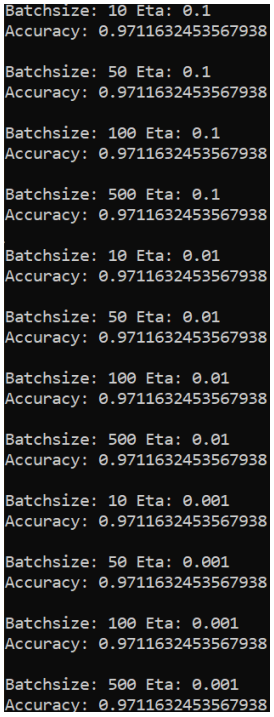

Figure 26: The Accuracy of Neural Network

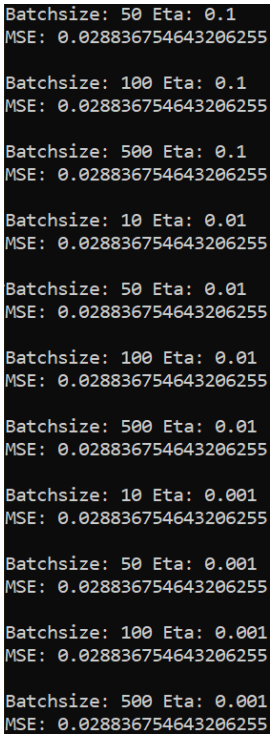

Figure 27: The MSE of Neural Network

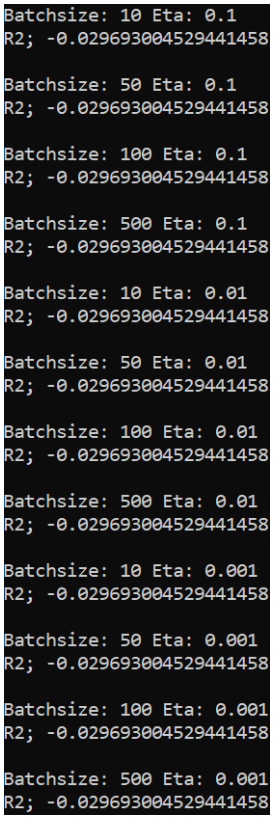

Figure 28: The R2 of Neural Network

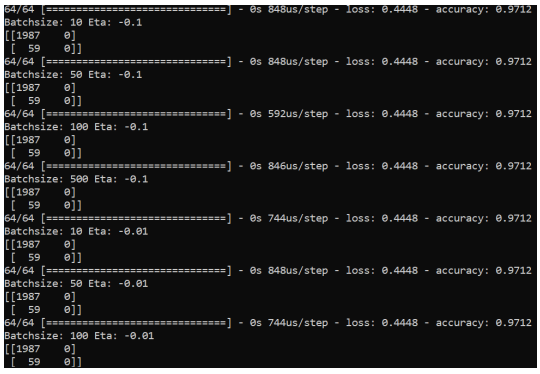

Figure 29: The Confusion Matrix of Neural Network

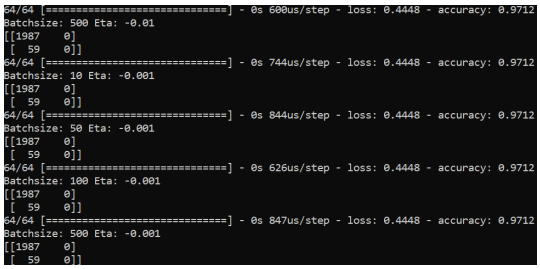

Figure 30: The Confusion Matrix of Neural Network

5.2 k-Nearest Neighbors

The second approach we tried was k-Nearest Neighbors with this we achieved some results that we could expect

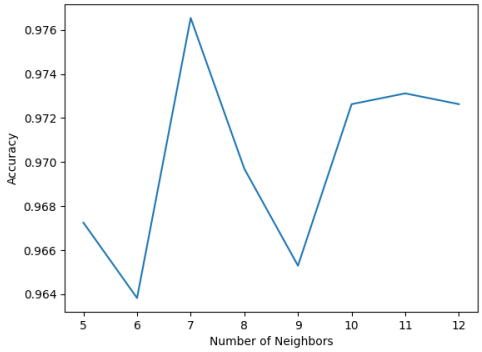

Figure 31: Accuracy of kNN based on number of neighbors

Here the accuracy of the model depended on the number of neighbors, but not in a way that made to much sense. We can perhaps see it stabilising around k = 10 in a higher accuracy than 96.77%

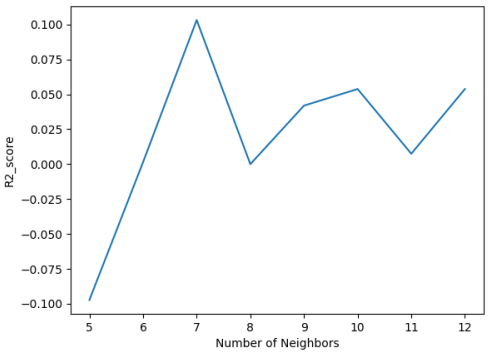

Figure 32: R2 score of kNN based on number of neighbors

The R2 score is getting a bit in the positive when we're increasing the number of neighbors.

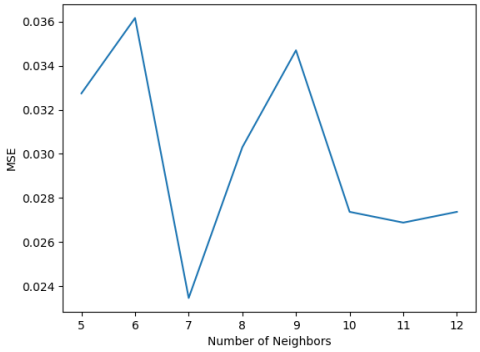

Figure 33: MSE of kNN based on number of neighbors

```
Number of Neighbors =  5
[[1968   15]
 [  52   11]]

Number of Neighbors =  6
[[1966    3]
 [  71    6]]

Number of Neighbors =  7
[[1987    4]
 [  44   11]]

Number of Neighbors =  8
[[1978    4]
 [  58    6]]
```

Figure 34: Confusion Matrix of kNN based on number of neighbors

```
Number of Neighbors =  9
[[1965    4]
 [  67   10]]

Number of Neighbors =  10
[[1984    1]
 [  55    6]]

Number of Neighbors =  11
[[1981    8]
 [  47   10]]

Number of Neighbors =  12
[[1984    1]
 [  55    6]]
```

Figure 35: Confusion Matrix of kNN based on number of neighbors

5.2.1 Removed variables

We used Backward Elimination to reduce the model to 41 models, in a hope to increase the output of the model.

```
New amount of features:  41 , Shape:  (6819, 41)
```

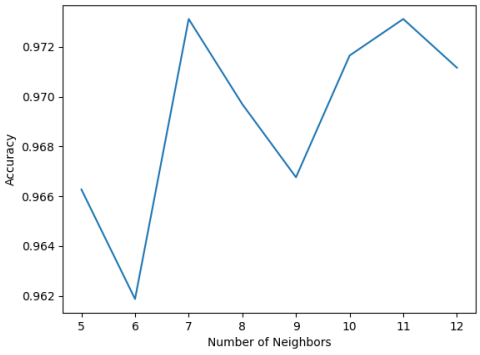

Figure 36: Accuracy of kNN based on number of neighbors

The accuracy decreases from before the BE

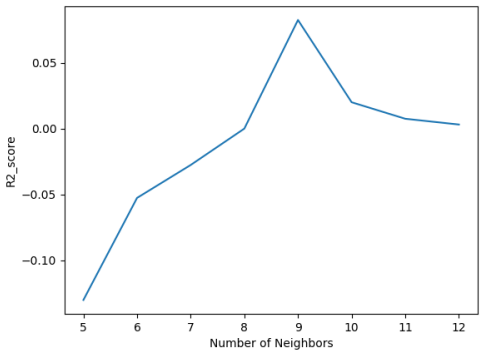

Figure 37: R2 score of kNN based on number of neighbors

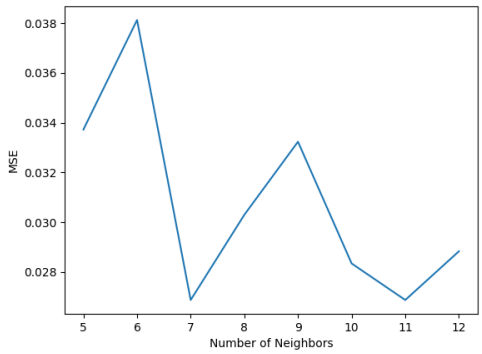

Figure 38: MSE of kNN based on number of neighbors

```
Number of Neighbors =  5
[[1963   20]
 [  49   14]]

Number of Neighbors =  6
[[1959   10]
 [  68    9]]

Number of Neighbors =  7
[[1982    9]
 [  46    9]]

Number of Neighbors =  8
[[1978    4]
 [  58    6]]
```

Figure 39: Confusion matrix of kNN based on number of neighbors

```
Number of Neighbors =  9
[[1965    4]
 [  64   13]]

Number of Neighbors =  10
[[1980    5]
 [  53    8]]

Number of Neighbors =  11
[[1980    9]
 [  46   11]]

Number of Neighbors =  12
[[1982    3]
 [  56    5]]
```

Figure 40: Confusion matrix of kNN based on number of neighbors

5.3 Decision Tree

```
Accuracy:  0.9496578690127078
```

Figure 41: Accuracy of decision trees

The accuracy of the decision tree on the test set is around 95%

```
R2 Score:  -0.6868621376942101
```

Figure 42: R2 score of decision trees

The model is actually in the negative when it comes to R2 score

```
MSE:  0.050342130987292275
```

Figure 43: MSE of decision trees

```
[[1920   63]
 [  40   23]]
```

Figure 44: Confusion matrix of decision trees

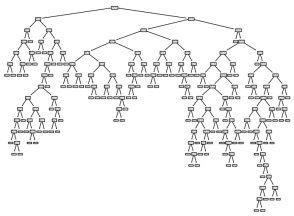

Figure 45: Decision tree

5.3.1 Removed variables

```
Accuracy:  0.9521016617790812
```

Figure 46: Accuracy of decision trees

```
R2 Score:  -0.6049756261556563
```

Figure 47: R2 score of decision trees

```
MSE:  0.047898338220918865
```

Figure 48: MSE of decision trees

```
[[1932   51]
 [  47   16]]
```

Figure 49: Confusion matrix of decision trees

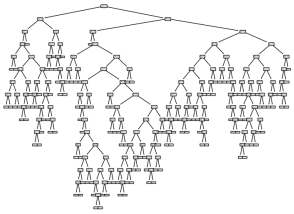

Figure 50: Decision tree

5.4 Linear Regression

5.4.1 OLS

We also want to look at how the linear regression models solves the problem. The first regressor is the Ordinary Least Square (OLS). The model assume that all of the features has the same influence. This means our prediction is the average predicted value from all of the features. The accuracy for the model is plotted as a function of polynomial degree.

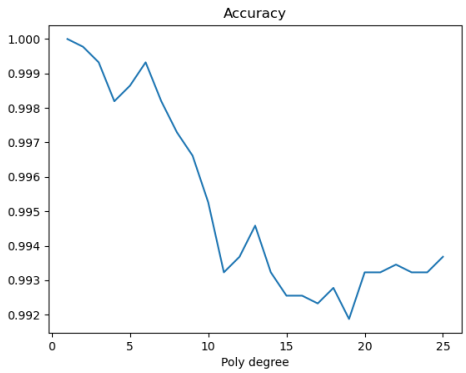

Figure 51: OLS accuracy

The plot show how the accuracy is decreasing as the polynomial order increase. The influence of increasing the polynomials decrease after a certain amount of order. The result tend to show that a larger degree makes a worse prediction than a low. This either means that the OLS is a bad choice for solving this problem at higher orders or that there is something wrong during our calculations. Either way the model has a good score of a above 99% which seems a bit too good to be true as linear regression should not be a good model for solving these types of problem.

5.4.2 Ridge

Ridge regression is used as our next model.

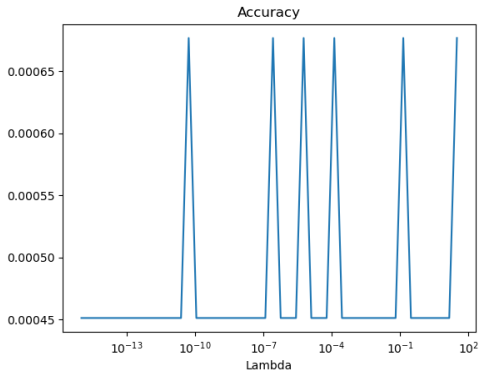

Figure 52: Ridge Regression accuracy

The result from this model is not behaving good. The score seems to fluctuate between two states. Even though the results are poor the score of the accuracy is very low as expected.

5.4.3 Lasso

As our last regressor we tried out Lasso regression.

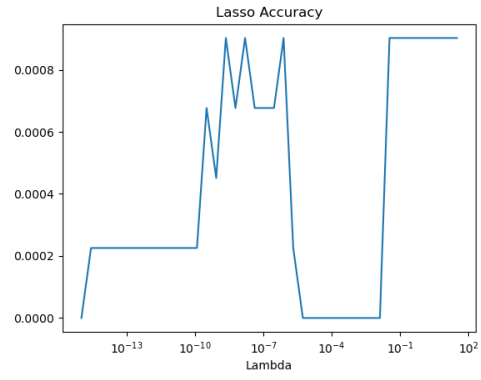

Figure 53: Lasso Regression accuracy

Lasso is also having troubles as Ridge. Both encounter warnings during the calculations and the results are varying. On the other hand the accuracy is low as we expected it to be.

5.5 Bonus task

For the bonus task we also look into how well the regressors fits and predict our data by evaluating its score with MSE, R2 and bias-variance.

5.5.1 OLS

First out are the results from the OLS

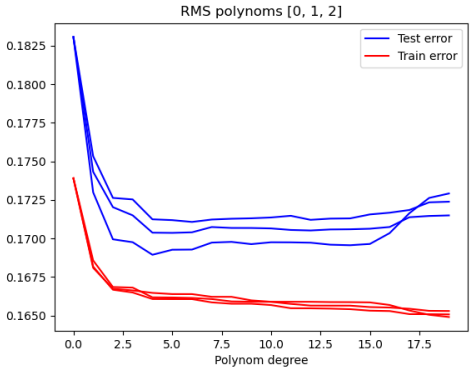

Figure 54: OLS RMS feature 1,2 and 3

The model is behaving good for some of the features in our data set. As for feature 1,2 and 3 the train error and test error behaves just as expected and wanted. The training error is decreasing and continue to decrease slowly. The test error is also decreasing at first, but then increase as the order of polynomial gets too high.

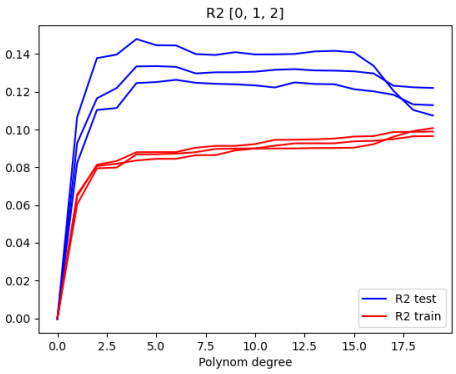

Figure 55: OLS R2 feature 1,2 and 3

The R2 score for the same features tend to show a increasing then steady score which means out model becomes stable at predicting after a certain amount of polynomial order.

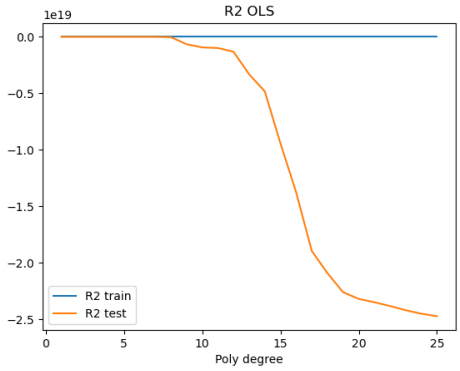

Figure 56: OLS R2

We have now added all of the features in our model. These results becomes bad, and plotted together they don't give us a lot of information. But if we plot them separate there might be some trends.

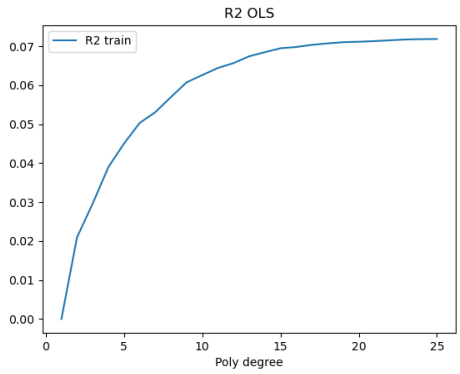

Figure 57: OLS R2 train

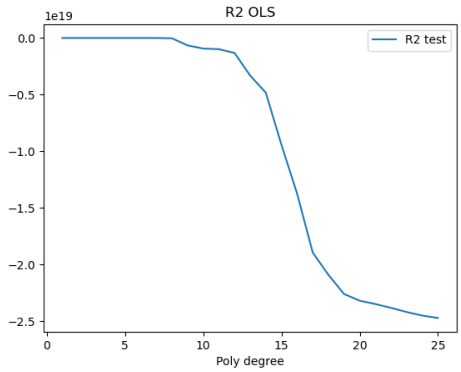

Figure 58: OLS R2 test

From the result is seems that a higher order makes the model fit the problem better and better but it becomes worse in predicting.

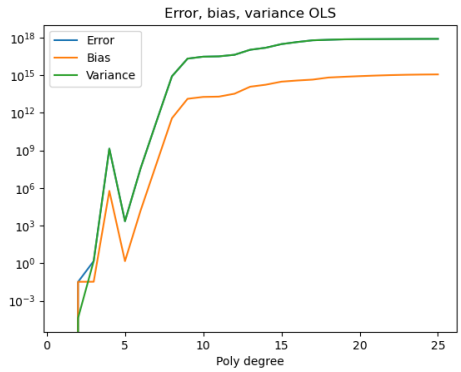

Figure 59: OLS error bias variance

At the end we look at the bias variance trade off. This behaviour is strange and is not giving a lot of good information. The Error is the same as the variance through almost all of the plot and the bias is following the variance.

5.5.2 Ridge

We also look at the same results for Ridge and Lasso.

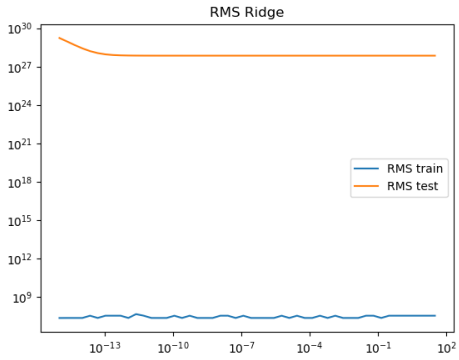

Figure 60: Ridge RMS

The two RMS plotted together does not give away a lot of information.

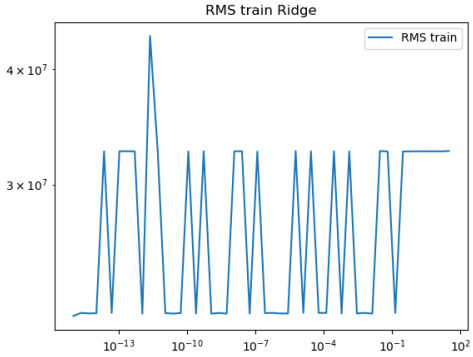

Figure 61: OLS RMS train

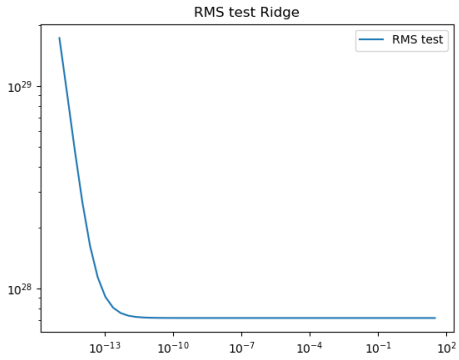

Figure 62: OLS RMS test

Again we encounter some weird fluctuating results in the calculations. The result of the RMS for the training data is too bad to valuate. The test RMS on the other hand tends to do as expected by dropping down as the lambdas increase.

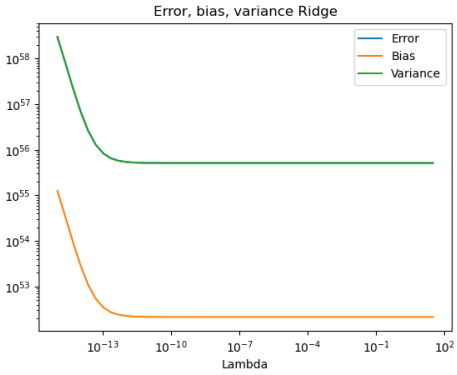

Figure 63: Ridge error bias variance

The error bias variance is also not telling us much for Ridge regression.

5.5.3 Lasso

At last we look at the Lasso regression.

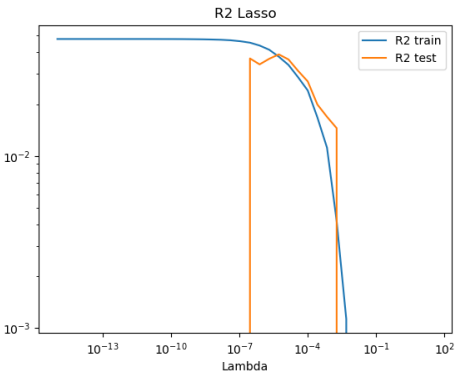

Figure 64: Lasso R2

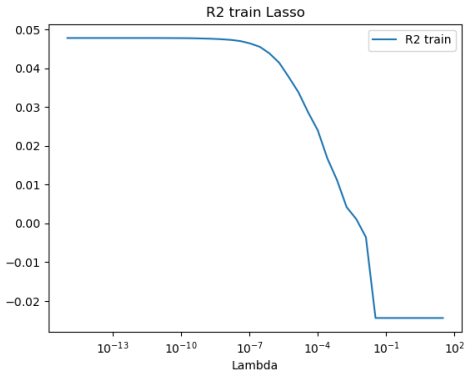

Figure 65: Lasso R2 train

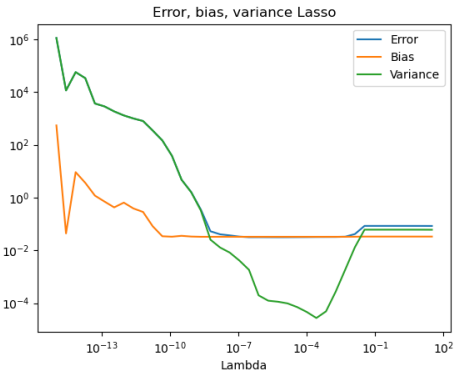

Figure 67: Lasso error bias variance

The error bias variance tends to behave somewhere as we want to with the variance decreasing and then increasing as the lambdas rise. On the other hand it is strange that the bias flattens out and the error keeps following the variance.

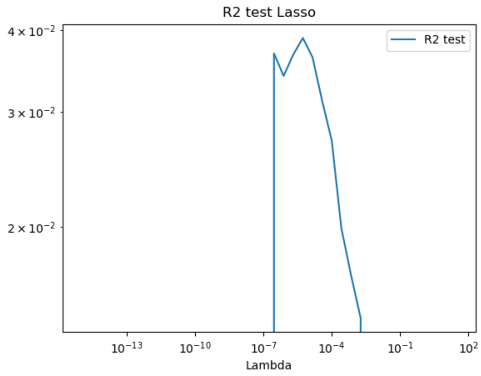

Figure 66: Lasso R2 test

The R2 score is starting high for the training data and then plunging into negative values at a high rate. The score tell us that the model is decreasing in predicting as the lambdas keep increasing. The test result is not valid to discuss. There is something really wrong.

6 Conclusions

We've discovered several interesting things about the probability of a company going bankrupt and the optimal model for discovering such a scenario without know prior.

When it comes to features, we've discovered that there ain't really a good way to remove enough of them to provide a simpler model without hurting the accuracy to much. This has many reasons, one of them is that the correlation between them isn't big enough and there isn't a small set of feature which affects to model in such a degree that we can remove the rest.

The methods used are usable in a scenario as this and provides us with different models.

The OLS has a high score at predicting, but it might not be a true prediction or a valid result due to different possible errors. The OLS has a better score than we expected. For Ridge and Lasso the computations was behaving strange with convergence warnings and overflow. The results were not too good either. This might come from a bad dataset, wrong size of training vs test set, wrong use of the regressors and many other factors. The result is varying much through the choice of

polynomials, random seed, step size, choice of axis and other vaiables.

NN isn't a to good predictor when we've god such a unbalanced samples. When we trained our model we only got a model which predicted every company to be doing fine. As the hit percentage of this is so high, it makes the self training of NN work against it's purpose as it couldn't find a way to improve the results.

kNN gave us a more acceptable answer than NN as it's gotten more variance. It works quite well in predicting the state of the companies when given the task, it does however not not improve or decline when increasing the number of nearest neighbors, k. In theory it will however in the end, with high enough k, just estimate every company, sample, to be doing fine. We therefore can't increase the k to much, and we can see that the decline in MSE is at it's lower at 7/11 which makes sense.

The decision tree is probably to easiest to understand and use by eyesight. It's quite large as expected of the high feature number, but at the same time it's always easy to read each node for itself. It didn't provide the best model as expected as it's quite stern in it's decisions.

From the results we arrived to the conclusion that the kNN is the best to use in this assignment in disclosing if a company is bankrupt or not as it has a high accuracy, while at the same time being able to predict companies going bankrupt.

References

[1] Project2, *Neurale Nettverk*, Vidar Lunde Olaisen, Thomas Bergheim, Oslo, *https://www.overleaf.com/project/6167f4a28b15e3ffa0aab71c* best edition, 2021