CS-UY 4563 Machine Learning

Final Project Written Report

Credit Risk Prediction of Bank Customers

April 26, 2023

Professor Linda M. Sellie

Aruuke Bayakmatova, Andy Lee

1. Introduction

In the final project, we used a dataset from kaggle that contained payment and customer data to predict which customers were high-credit risks. The payment data included 12 features: customer id, number of times overdue types 1-3, total overdue days, number of times normal payment, credit product code, credit limit of product, account update date, current balance of product, highest balance in history, and date of recent payment. The customer data included 6 demographic features and category attributes that were already encoded. These features were used to indicate whether the customer was a low-credit or high-credit risk when they applied for banking products.

This is a binary classification problem, with the goal of classifying whether a bank customer will be a low-credit or high-credit risk. In this project, we used logistic regression, support vector machines (SVM), and neural networks. The data was split into 70% training, 15% test, and 15% validation sets.

- Train set - trained the model.
- Validation set - used to select the best hyperparameter, C.
- Test - evaluated the final model.

2. Data Preparation

Cleaning the Dataset

The raw dataset contained incomplete features with null entries and features that did not contribute to the credit risk prediction. Irrelevant features were dropped, null values were filled in with the median values of each feature.

In the SVM portion, the customer_id, update_date, reported_date, prod_code, and prod_limit were removed from the dataset since they did not contribute to the credit risk of the customer. Data values were scaled using StandardScaler.

In logistic regression, empty or duplicate rows were removed and date features in the string format were converted seconds after January 1, 1970. Additionally, PCA was employed to reduce the dimensionality of the dataset while retaining the most relevant information. PCA identifies the principal components of the data that capture the maximum amount of variation in the dataset. The number of principal components to be retained can be chosen based on the desired level of explained variance. By using the PCA function, an optimal number of components was selected to achieve the desired level of variance. The Standard Scaler was utilized to normalize the data from each set. Specifically, the data was scaled to have zero mean and unit variance, prior to implementing PCA. L2 normalization was used to prevent overfitting through adding a penalty term to the cost function, penalizing larger weights more than smaller weights. The best penalty was found by selecting a value which resulted in the best validation accuracy.

In neural networks, the 'update_date' and 'reported_date' features were converted from string to date-time format, and then combined to create the 'reported_updated_date' feature. In Perceptron, L1 normalization was used to add a penalty term to the cost function to shrink the weights towards zero. This was done to help reduce overfitting and improve the generalization performance of the model. Likewise, the best penalty was found by selecting a value which resulted in the best validation accuracy, and the results of L1 regularization were compared to the results without regularization. In Multi-Layer Perceptron, trained and evaluated on a dataset using different combinations of hidden layer sizes and activation functions (ReLU,logistic,tanh).

Models

Logistic regression

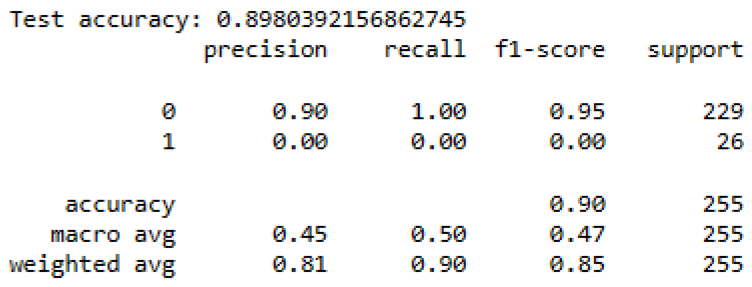
The first model used for credit-risk prediction was logistic regression. As previously mentioned, principal component analysis (PCA) was used to reduce the dimensionality of the logistic regression model while maintaining the most relevant information. L2 normalization was used to prevent overfitting and the best penalty was selected from the best validation accuracy.

Table 1 below shows the validation accuracies for the logistic regression model at different C values. The best performing C values were 1.0 and 100000.0 as these values had the same validation accuracy. The best C-value was identified as 1.0 and best validation accuracy as 0.8902.

Table 1: Validation accuracies of logistic regression model at different C-values

C-value	Validation Accuracy
0.00001	0.8902
1.0	0.8902
100000.0	0.8902

Figure 1 below shows the classification report for the best logistic regression model. The label “0” represents low-credit risk while “1” represents high-credit risk.



```
Test accuracy: 0.8980392156862745
              precision    recall  f1-score   support

           0       0.90      1.00      0.95       229
           1       0.00      0.00      0.00        26

    accuracy                           0.90       255
   macro avg       0.45      0.50      0.47       255
weighted avg       0.81      0.90      0.85       255
```

Figure 1: Classification report for the best logistic regression model

Figure 2 below shows the confusion matrix for the best logistic regression model. The figure shows that were: 229 correct low-credit risk, 26 incorrect low-credit risk, 0 incorrect high-credit risk, and 0 correct high-credit risk customer classifications.

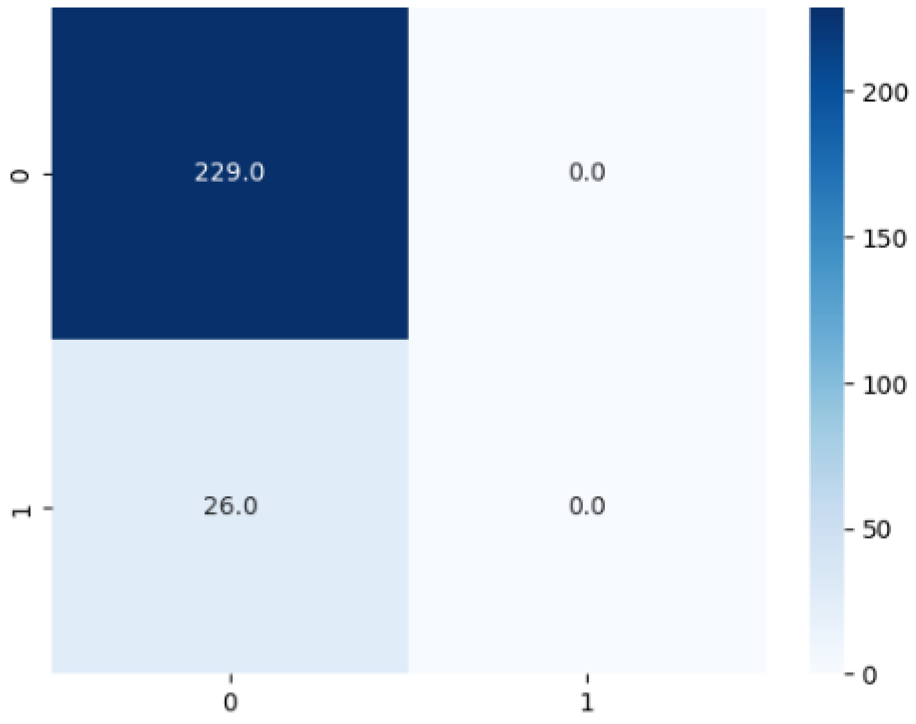


Figure 2: Confusion matrix for the best performing logistic regression model

Conclusion

As indicated by the results above, the best logistic regression model had a C-regularization parameter of 0.00001. However, as seen in Figure 1, the model struggles at correctly predicting high-credit risks with an abysmal 0% precision compared to 90% precision when predicting low-credit risks. The test accuracy was 0.8980 and the validation accuracy was 0.8902. There was no need to transform the data to higher complexities since the data appears to be linearly separable based on the high accuracies.

Support vector machine

Using the Sckit-Learn Python library, the data was trained using linear, polynomial, and radial basis function (RBF) kernels. The polynomial kernel was tested at various degrees and each kernel was tested at different C regularization parameters from 0.1 to 3. The test accuracies of each kernel were recorded and compared.

Linear Kernel

The linear kernel was tested at different C regularization values, 0.1, 1, and 10,000. L1 and L2 regularization was not applied. The test accuracy was 0.88 for all C-values. Figure 3 below shows the results of the linear kernel.

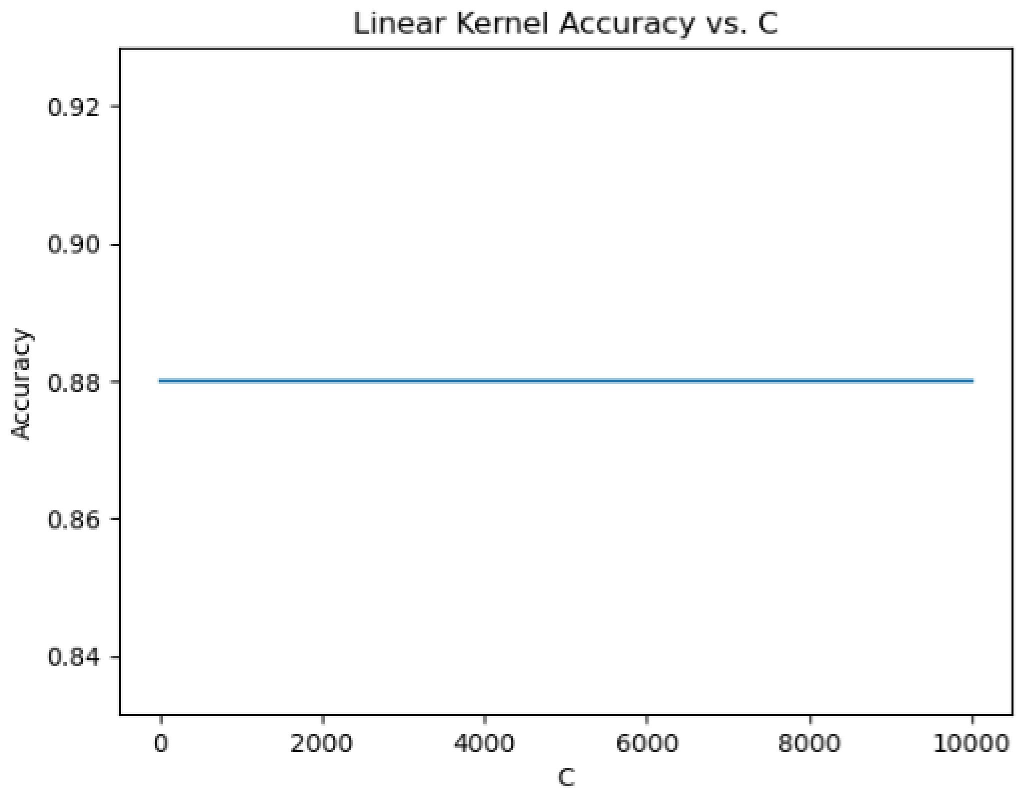
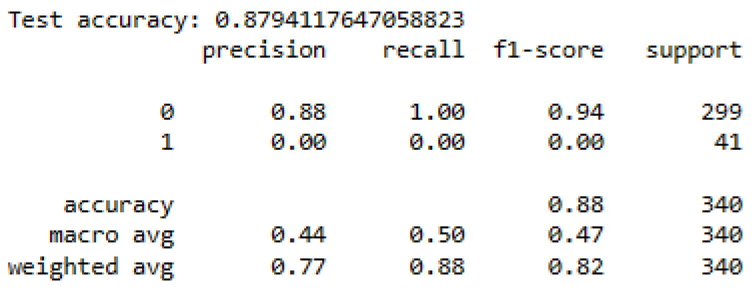


Figure 3: Linear kernel test accuracy at varying C parameters

Figure 4 below shows the classification report for the linear kernel. The kernel was very precise when predicting low-credit risk but had a terrible 0% precision when predicting high-credit risk.



```
Test accuracy: 0.8794117647058823
              precision    recall  f1-score   support

           0       0.88      1.00      0.94       299
           1       0.00      0.00      0.00        41

    accuracy                           0.88       340
   macro avg       0.44      0.50      0.47       340
weighted avg       0.77      0.88      0.82       340
```

Figure 4: Classification report for linear kernel

Polynomial Kernel

The polynomial kernel was also tested at different C regularizations of 0.00001, 1.0, and 10,000. The polynomial kernel was also tested at different degrees from 1 to 9. Figure 5 below shows the results of the tests. The best degree was 1 which suggests that the dataset was likely already linearly separable. The best validation accuracy was 0.8566 and the best test accuracy was 0.8794.

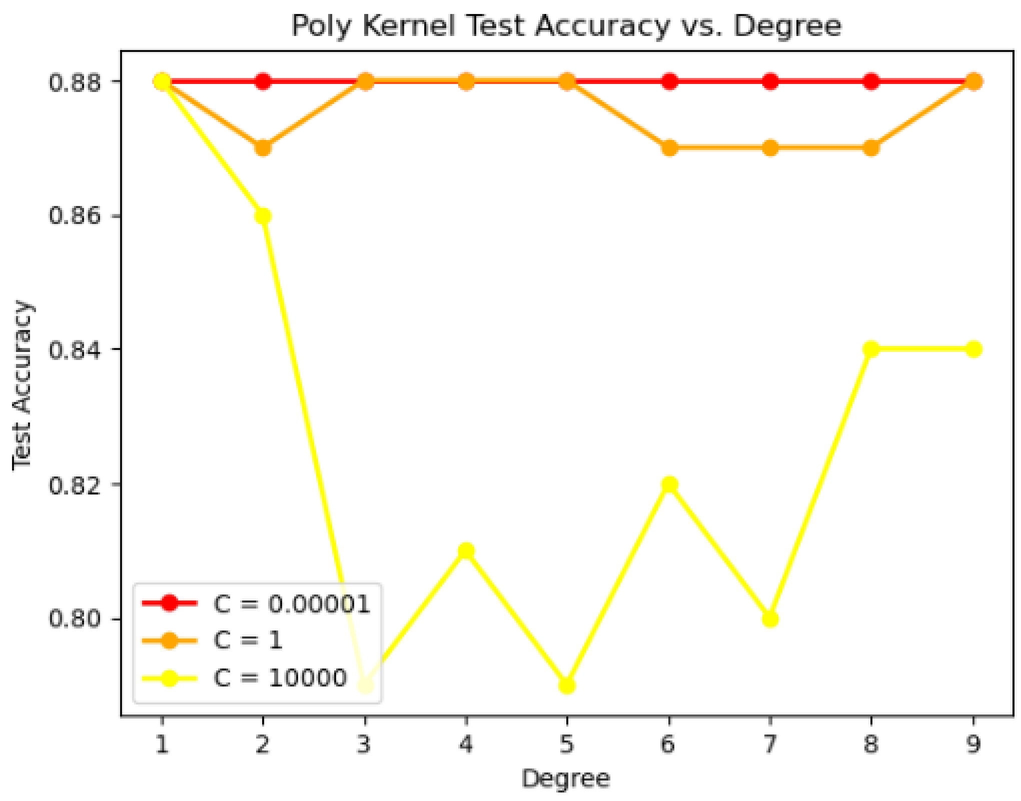


Figure 5: Polynomial kernel test accuracy at varying degrees and C regularization parameters

Since the best polynomial kernel was degree 1, its classification report was the same as the linear kernel's shown in Figure 4. Consequently, it suffers from the same issues of poor precision when predicting high-credit risk customers.

RBF Kernel

Like the previous two kernels, the RBF kernel was also trained on different C regularizations of 0.00001, 1.0, and 10,000. Figure 6 below shows the results of the training. The best C-value was 1.0 with a test accuracy of 0.882.

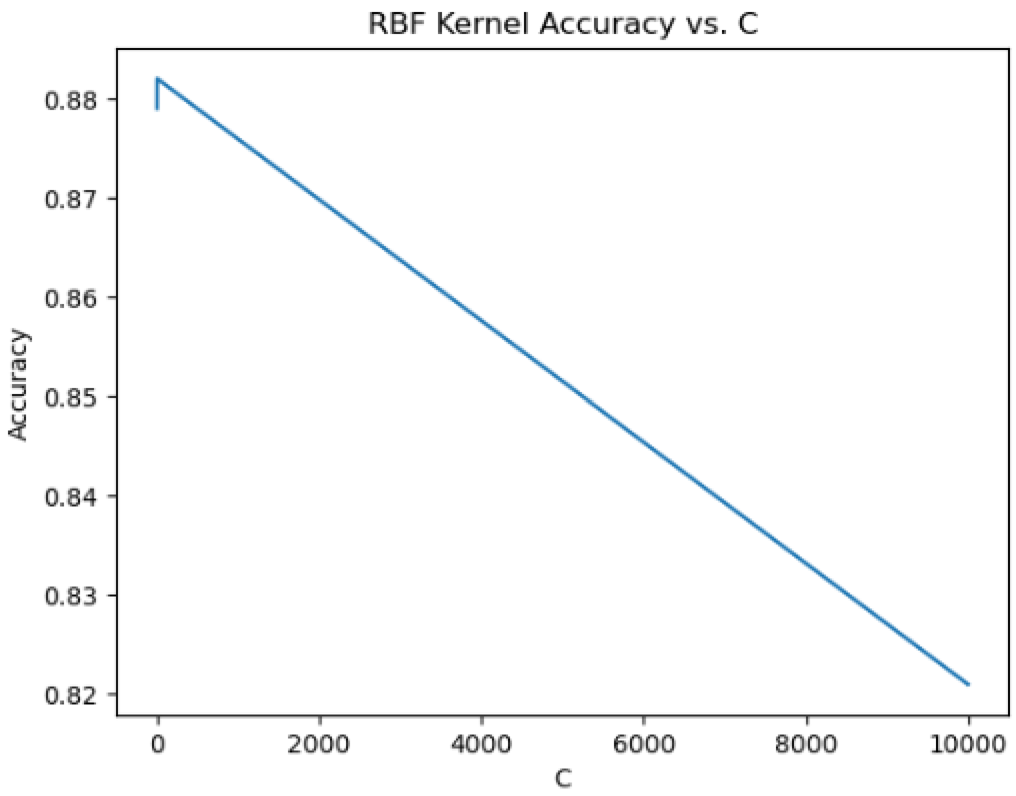
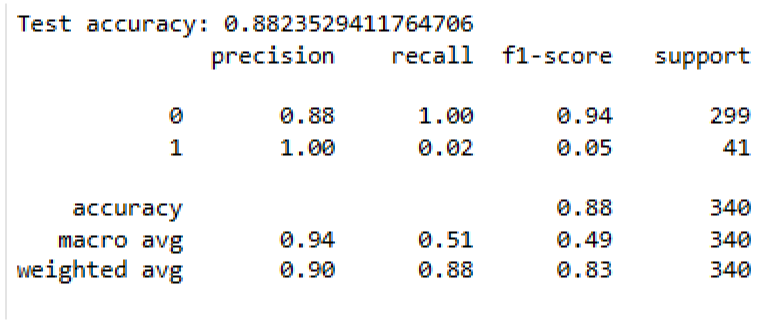


Figure 6: RBF kernel test accuracy at varying C parameters

Figure 7 below shows the classification report for the best-performing RBF kernel. The RBF kernel was great at predicting both low-credit and high-credit risk customers at 88% and 100% precision, respectively.



```
Test accuracy: 0.8823529411764706
              precision    recall  f1-score   support

           0       0.88      1.00      0.94       299
           1       1.00      0.02      0.05        41

    accuracy                           0.88       340
   macro avg       0.94      0.51      0.49       340
weighted avg       0.90      0.88      0.83       340
```

Figure 7: Classification report for RBF kernel

Conclusion

The high test accuracies of the linear and first-degree polynomial kernel of 0.8794 indicate that the dataset may already be linearly separable. However, the best-performing kernel was the RBF kernel as the linear and polynomial kernels performed very poorly when predicting high-credit risk customers at an abysmal 0% precision. The polynomial kernel did see some improvement in predicting high-credit risk customers at higher degrees. However, there was no significant correlation between high degrees and precision because at some higher-level degrees the precision of high-credit risk predictions was lower than at lower-level degrees. This suggests that the higher-degree models were fitting the noise in the model rather than the data. Therefore, the RBF kernel was the best choice since it: adjusts better to complex data, produces the best test accuracy, and has the highest precision percentages for predicting low-credit and high-credit risk customers. The RBF kernel had the highest test accuracy at 0.8824.

Neural Network

For the neural network architecture, single-layer and multi-layer perceptrons were used. Only the ReLU activation function was used in the single-layer neural network to find the best performing model. Logistic, ReLu, and tanh activation functions were applied to find the best performing model in the multi-layer neural networks. No regularization was applied to the neural networks.

Single-Layer Perceptron

The single-layer perceptron was used to represent a linear model. The input layer had 23 neurons corresponding to the 23 selected prediction features. There were no hidden layers given that this was a single-layer implementation. The output layer included just one neuron indicating the classification of high or low credit risk. Table 2 below shows that changing the C value did not affect the validation accuracy.

Table 2: Different C values and their validation accuracies

C Regularization Value	Validation Accuracy
0.00001	0.8934
1.0	0.8934
10000000000.0	0.8934

Table 3 below shows the best test accuracy, best C-value, and best validation accuracy. Since the validation accuracies were the same at every C-value that was tested, the smallest C-value was taken as the best value.

Table 3: Best C-value, test accuracy, and validation accuracy for single-layer perceptron

Best C-value	Best Test Accuracy	Best Validation Accuracy
0.00001	0.8824	0.8934

Multi-Layer Perceptron

In the multi-layer perceptron section, ReLU, logistic, and tanh activation functions were used on different numbers of hidden layers that each contained 1697 neurons representing the 1697 feature values. The first network had 2 hidden layers, the second network had 3 hidden layers, and the third network had 4 hidden layers. Each layer in each network was run three times using every activation function. The best activation function, best hidden layer size, and best validation accuracy was updated with each run to find the best model. The output layer for each network consisted of one neuron classifying the customer as a high or low credit risk.

Two-Layer Neural Network

Table 4 below shows the results for the first network which had two hidden layers each with 1697 neurons. The validation accuracy was 0.8934 for all three activation functions. This validation accuracy is the same as the validation accuracy calculated using a single perceptron.

Table 4: Activation functions and their validation accuracy for the first network

Activation Function	Validation Accuracy
ReLU	0.8934
Logistic	0.8934
tanh	0.8934

Three-Layer Neural Network

Table 5 below shows the results for the second network which had three hidden layers each with 1697 neurons. The validation accuracies were the exact same as in the first network.

Table 5: Activation functions and their validation accuracy for the second network

Activation Function	Validation Accuracy
ReLU	0.8934
Logistic	0.8934
tanh	0.8934

Four-Layer Neural Network

Table 6 below shows the results for the second network which had four hidden layers each with 1697 neurons. The validation accuracy for the ReLU activation function decreased by approximately 0.0037 while the other validation accuracies for logistic and tanh stayed the same.

Table 6: Activation functions and their validation accuracy for the third network

Activation Function	Validation Accuracy
ReLU	0.8897
Logistic	0.8934
tanh	0.8934

Conclusion

In conclusion, the best configuration was using ReLU activation on a two-layer neural network. This configuration yielded the best validation accuracy at 0.8934 and the best test accuracy of 0.8824. Figure 8 below shows the confusion matrix for the best configuration

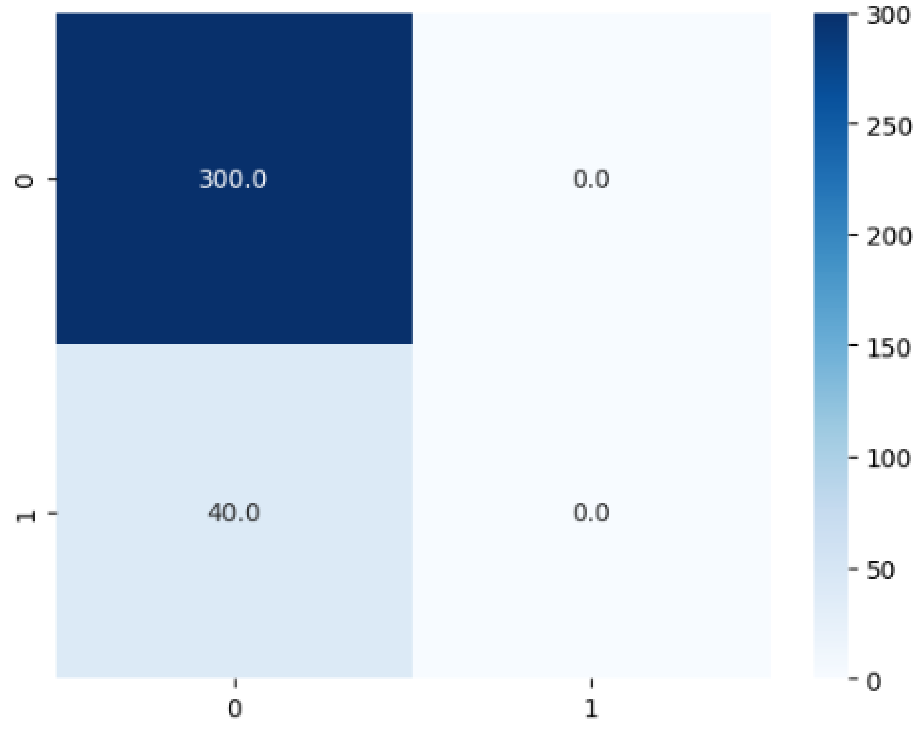


Figure 8: Confusion matrix for the best neural network configuration

As seen in the figure above, the neural network correctly classified 300 customers as low-credit risk but did very poorly with classifying customers as high-credit risks. There were 40 false positives or high-risk customers classified as low-risk. No customers were classified as high-credit risk so there were no false negatives. This meant there was 0% precision when predicting high-credit risk customers.

3. Final Conclusion

The best test accuracies obtained for our logistic regression, support vector machine, and neural network models are displayed in Table 7 below.

Table 7: Best test results across models

Logistic Regression	Support Vector Machine	Neural Network
0.8902	0.8824	0.8824

The dataset appears to be linearly separable based on the high test and validation accuracies of the linear kernel and of the polynomial kernel SVM when the polynomial degree was 1. The two kernels suffered from a 0% precision when predicting high-credit risk customers and this may have been because the original dataset was already standardized. As a result, two standardizations were performed on the data which may have yielded inaccurate results. This likely also affected the logistic regression and neural network since their precisions were also poor despite having high test and validation accuracies. Further investigation is needed to determine the root cause since the problem could stem from irrelevant features or a need for better regularization techniques. The RBF kernel did not experience precision issues with its built-in adaptability and as a result it was able to generalize to predict both low-credit and high-credit risk with high precision at 0.88 and 1.00, respectively.