CS412 PROJECT REPORT

Project Name: German Credit

Project Members:

Doğukan Iğdeci 27993

Uğurcan Uğur 26448

Ahmet Burak Ekmekcioğlu 29143

Project Collaboratory link: https://colab.research.google.com/drive/16Ve4WGRa-MoGFUL3bOfOlZllvT9ugUS-#scrollTo=OnfAW4cFZr7n

In this project, we analyzed four different machine learning algorithms for the task of classifying a given client information to a good or a bad credit applicant on the German Credit Dataset with 1000 data points and 10 attributes. Our models include KNeighborsClassifier, Logistic Regression, Decision Trees and Support Vector Machines. As to hyperparameter tuning, we tested the mean validation accuracy for every 5-Fold. Hyperparameters tested for KNN, DT, SVM and LR include number of neighbors and distance metric; maximum leaf nodes and maximum depth; cut-off, kernel and gamma; and penalty respectively.

Their performances revolve around the baseline accuracy, accuracy when classifying every datapoint as good, 0.7. We obtained the best accuracy through the experiment step which involved selection of 9 features, and got 73% accuracy with SVM classification.

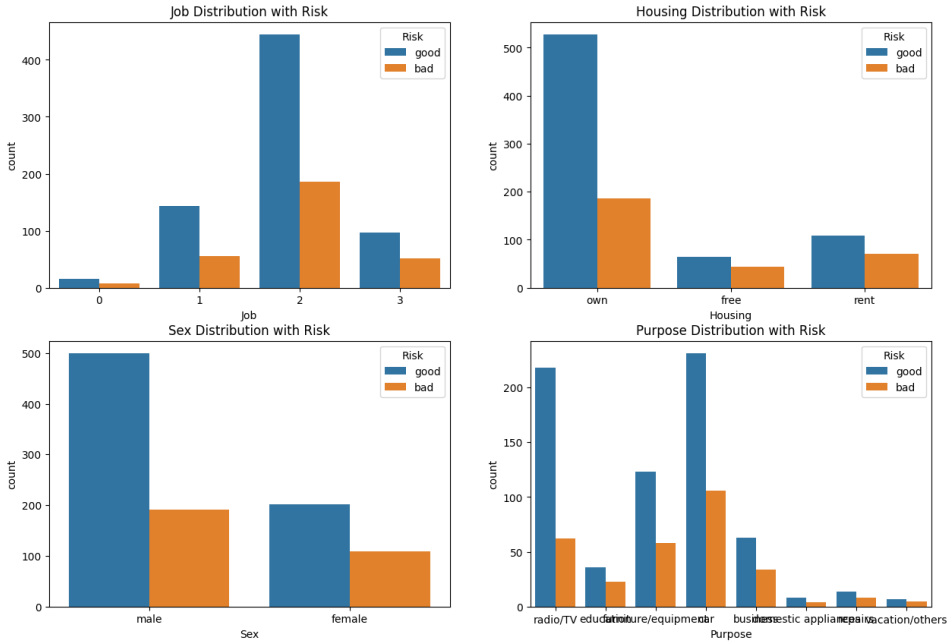
In order to make our machine learning model we need to make classification by changing our string values to binary values. Our binary variables are housing ,sex ,purpose and risk. Classification variables are for ‘Checking account’: little=1, moderate= 2, rich= 3 and for ‘saving accounts’: little=1, moderat= 2, rich= 3, quite rich= 4.

Our dataset includes 1000 data points, all with 10 attributes, namely Age, Sex, Job, Saving accounts, Checking account, Duration, Credit Amount, Housing and Purpose. A large portion of Saving accounts and Checking account columns were empty, 39.4% for Checking account and 18.3% for Saving accounts, while all other columns were intact. All labels except Age, Credit amount, Duration and Job were in character strings. Therefore, we one-hot encoded Sex, Housing and Risk; and ordinally encoded Checking accounts and saving account columns, since they involved an ordinal relationship between variables (little < moderate < rich < quite rich).

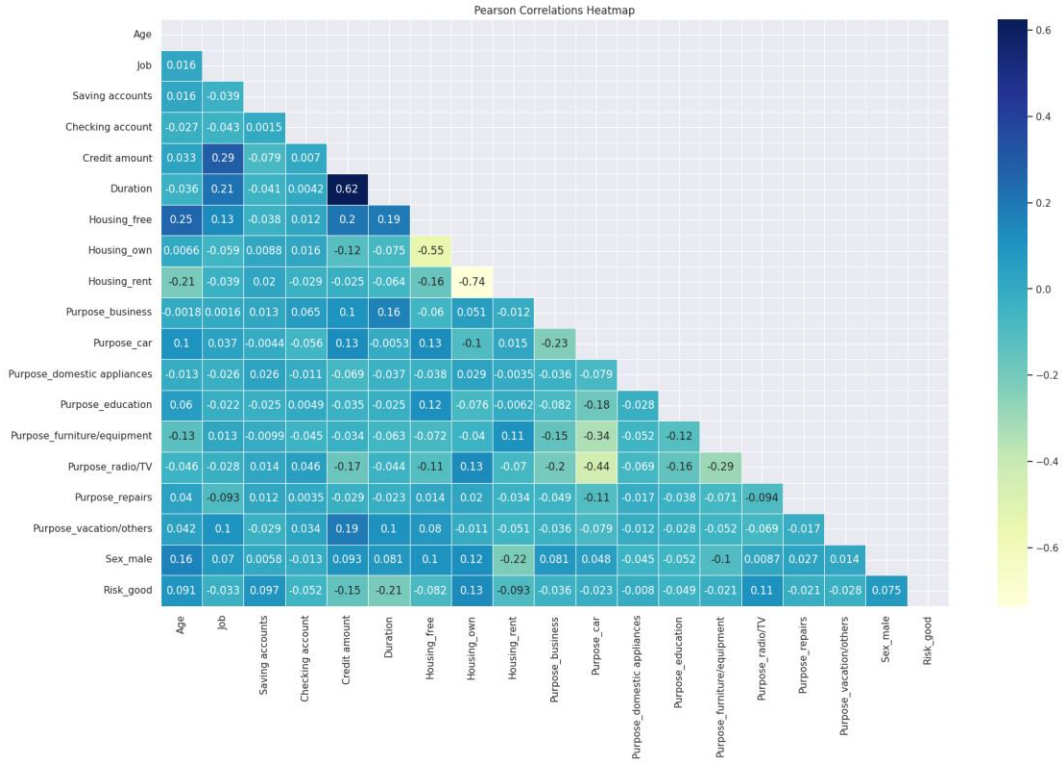
Below are some of the densities of some columns, and bar charts with respect to their label.

Distributions of columns Age, Credit amount and Duration

Counts with respect to Job, Housing, Sex and Purpose



After preprocessing, we plot a heatmap showing correlation between variables.



As to how we utilized our datasets to fit into our models, we decided to allocate 20% of samples for test data and 80% for train-validation data. Moreover, we used a 5-fold cross validation technique, so that we can test our hyperparameters more efficiently. In every fold, we had 640 training samples and 160 validation samples. Below are two examples of good and bad applicants:

42	female	2	rent	rich	rich	409	12	radio/TV	good
63	male	2	own	little	little	6836	60	business	bad

Our methodology can be summarized as encoding, normalizing and imputing. Since many columns were strings, encoding was necessary. Normalization is done before imputing, since we did not have enough data to say that the distribution of missing values would represent the true population of their respective fields. As to hyperparameter tuning,

For SVM model, we tested for below parameters and values:

Cs = [0.0001, 0.001, 0.01, 0.1, 1, 10]

kernels = ['linear', 'rbf', 'sigmoid']

gammas = [0.0001, 0.001, 0.01, 0.1, 1, 10]

We also tried a polynomial kernel; however, it took much more time relatively than the other kernels and produced no or less accuracy. Thus, we did not include it in the final evaluation.

For KNN model:

distance_metric = ['euclidean', 'manhattan']

N_neighbors = from 1 to 102 by 2 steps, so always odd.

Distance metric ‘cosine’ was also used, despite not providing significant improvement.

For DT model:

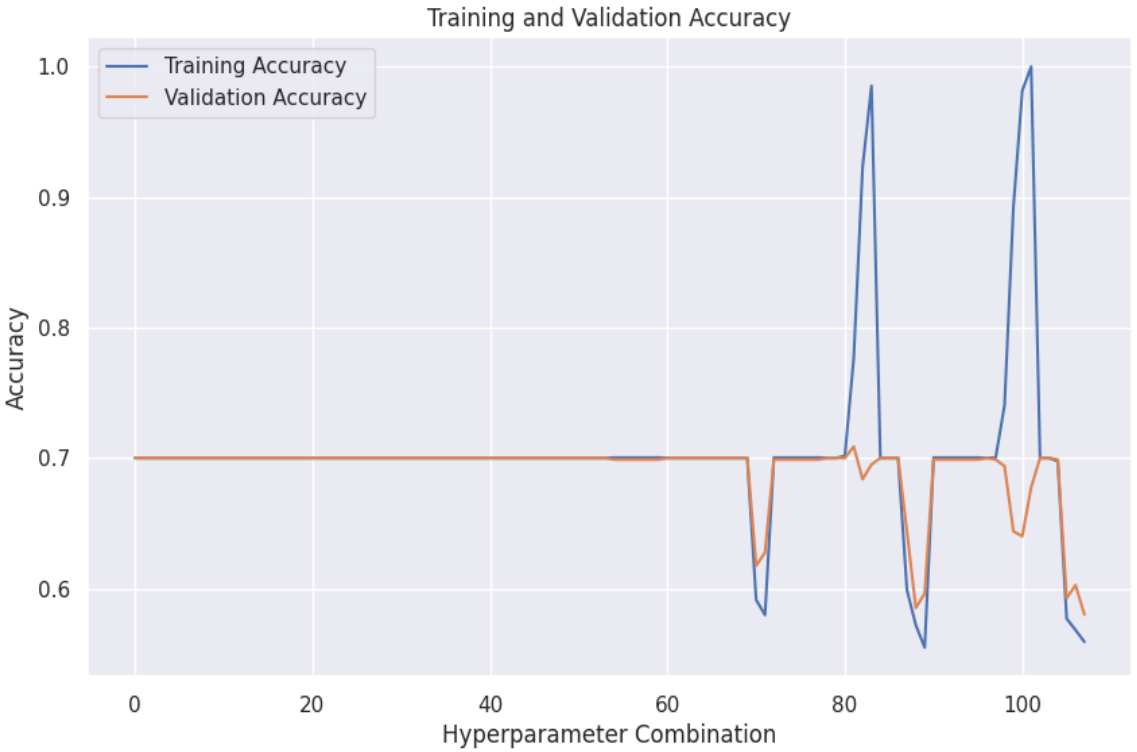
max_depth = from 1 to 31

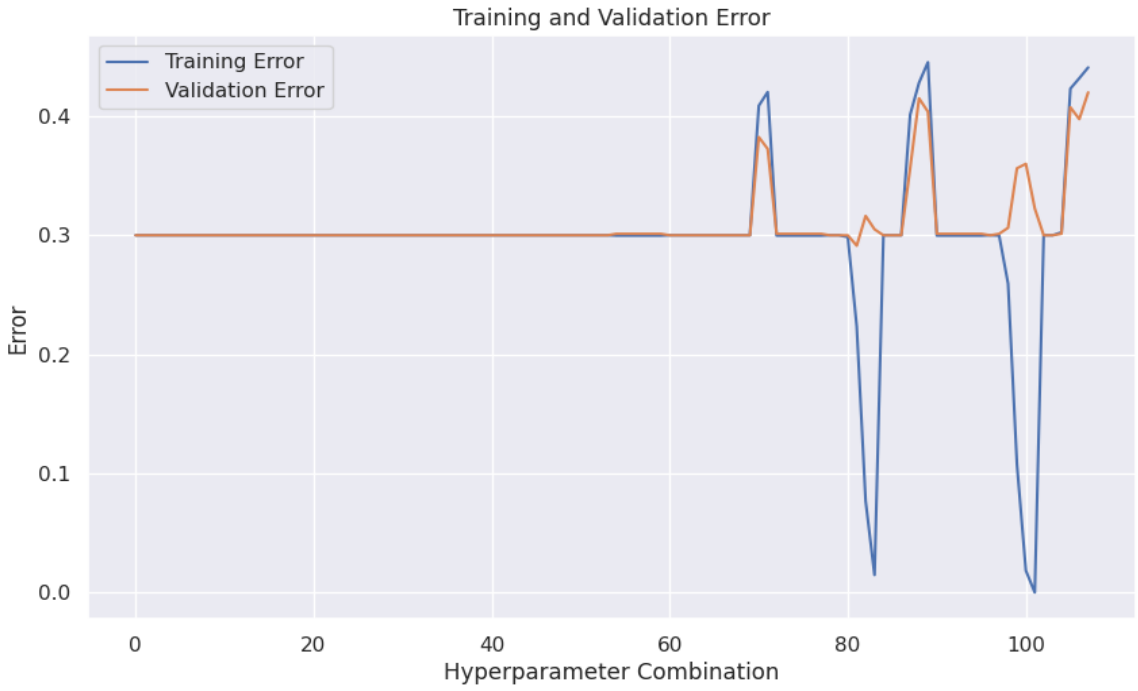
Max_leaf_nodes = from 2 to 15

For LR model:

Penalty = [‘l2’, None]

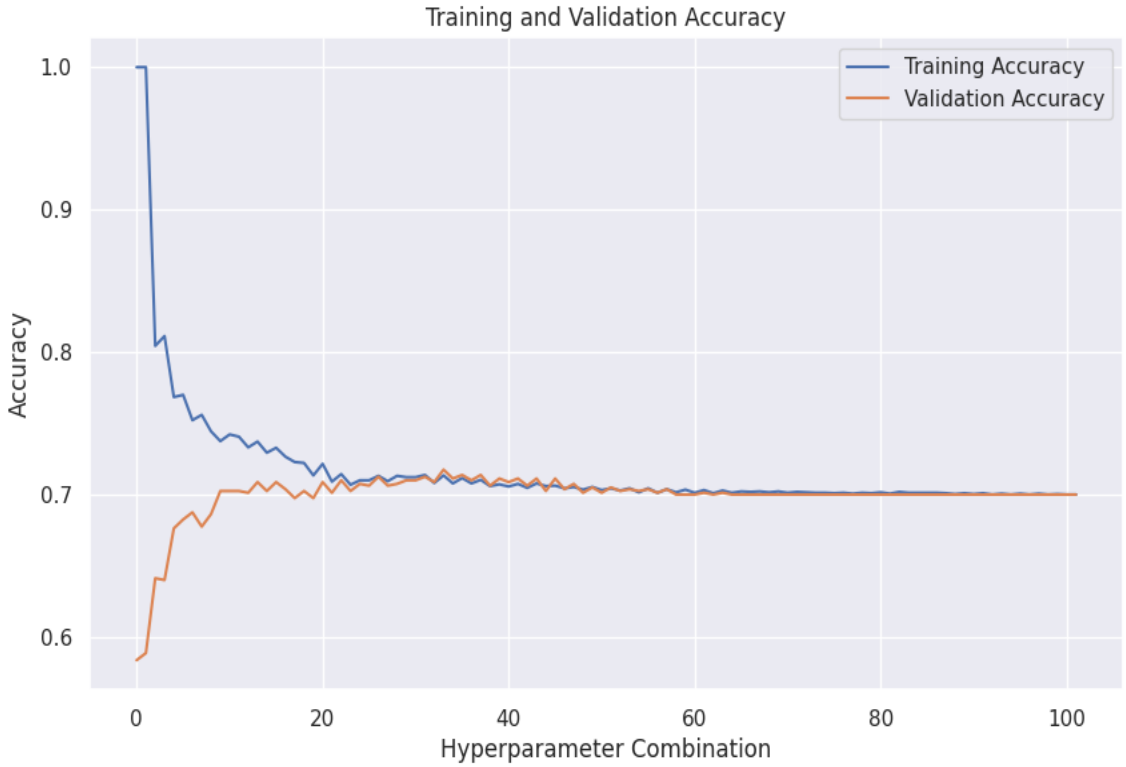
For SVM

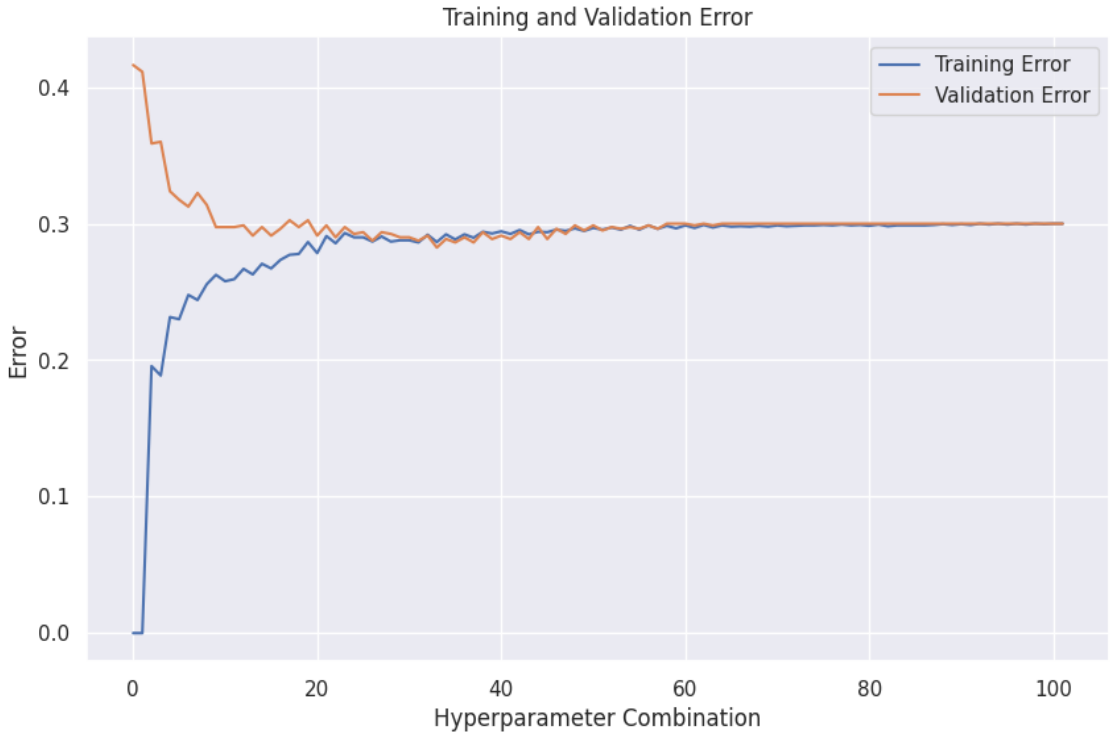




Best accuracy is 0.70875 best c value : 1 and the best kernel rbf gama : 0.1

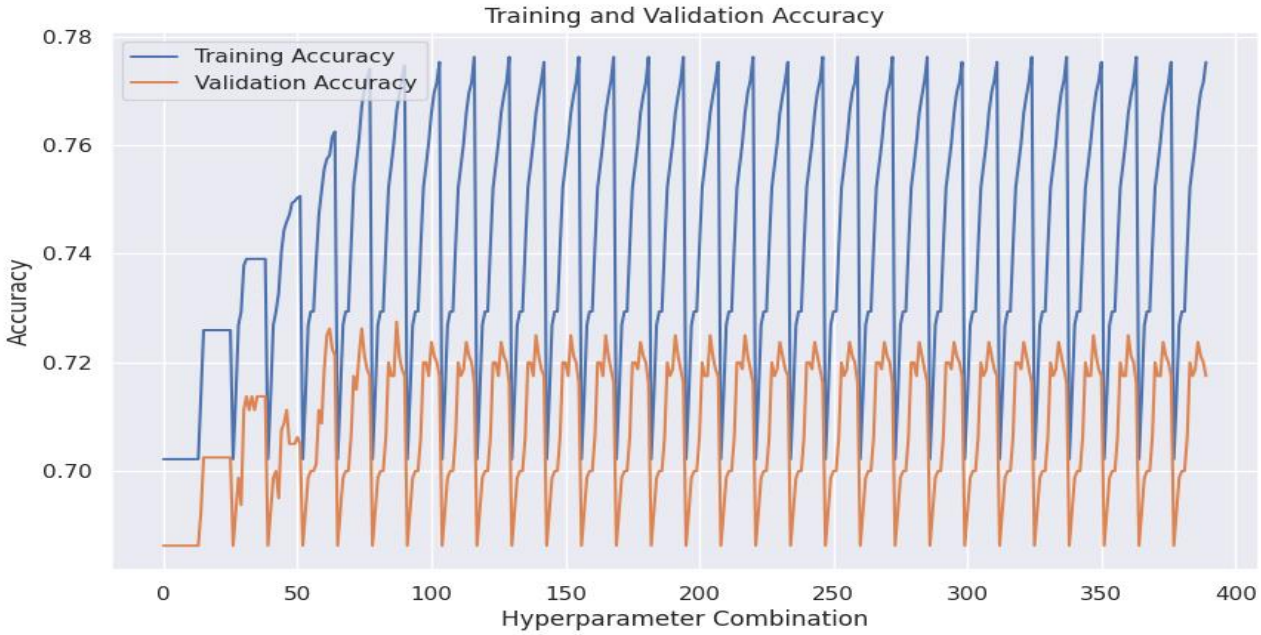
For KNN

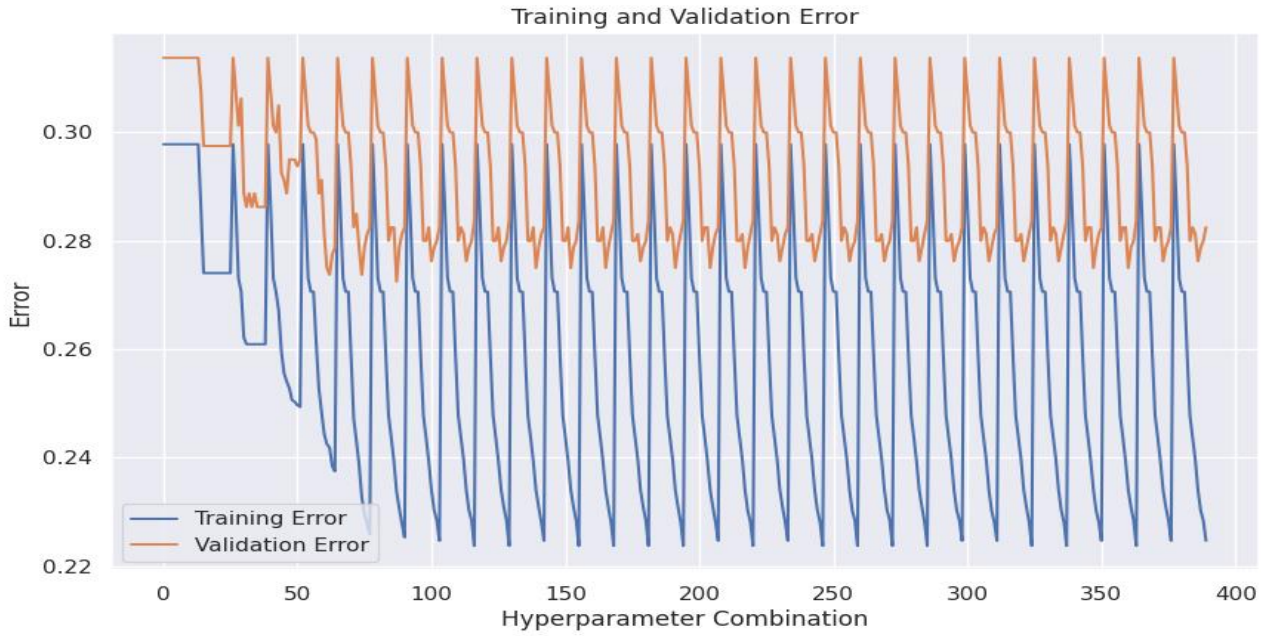




Best k is 33 , best metric is Manhattan , best accuracy is 0.7174

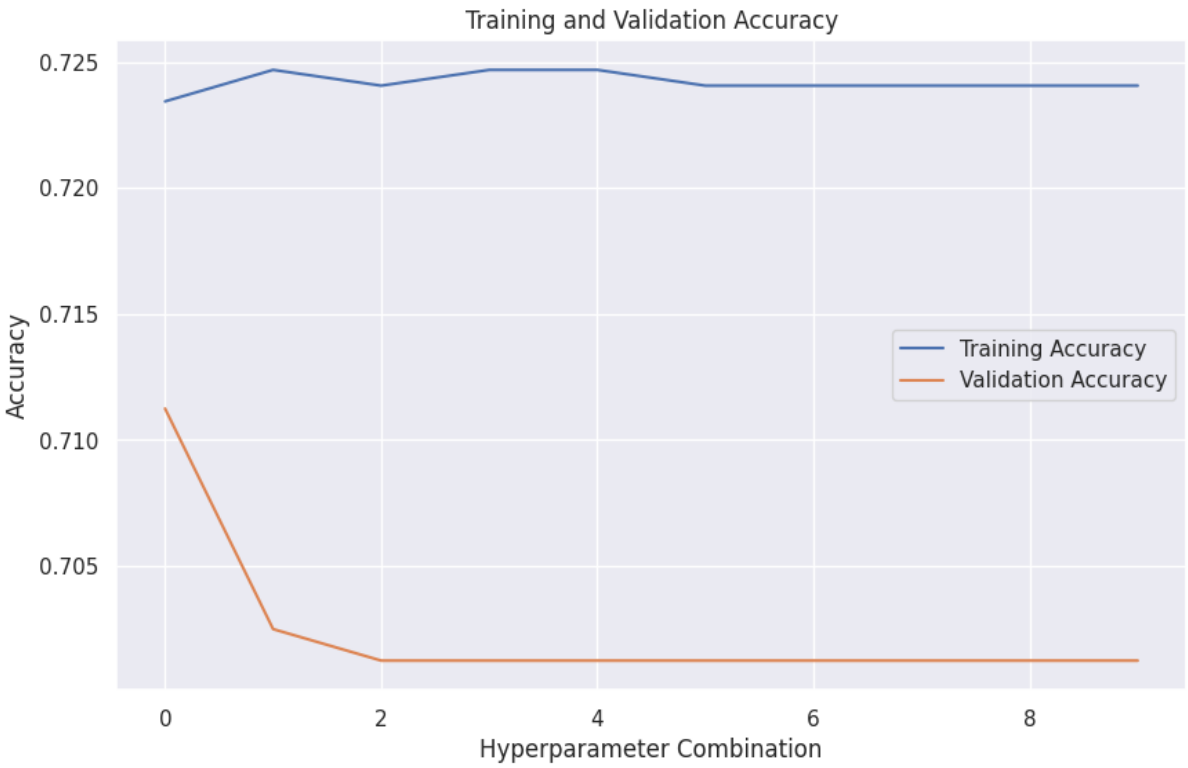
Decision tree classifier

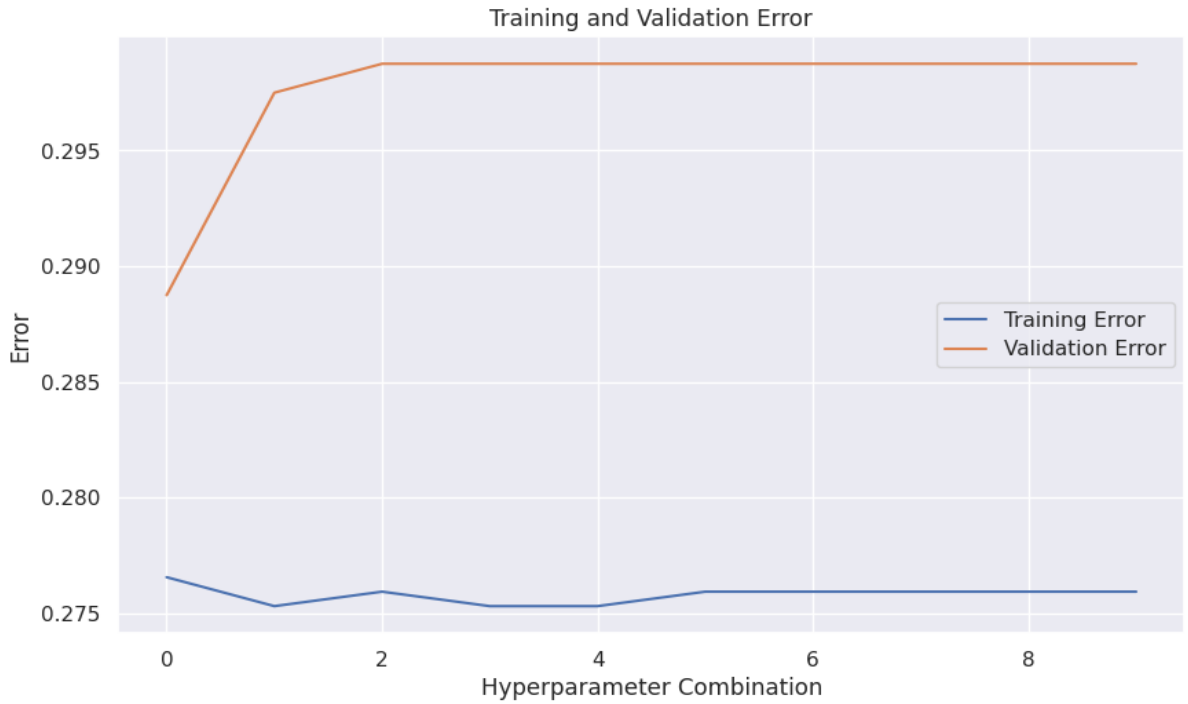




Best score is 0.7275 while best max depth is 7 and best max leaf nodes is 11

logistic regression



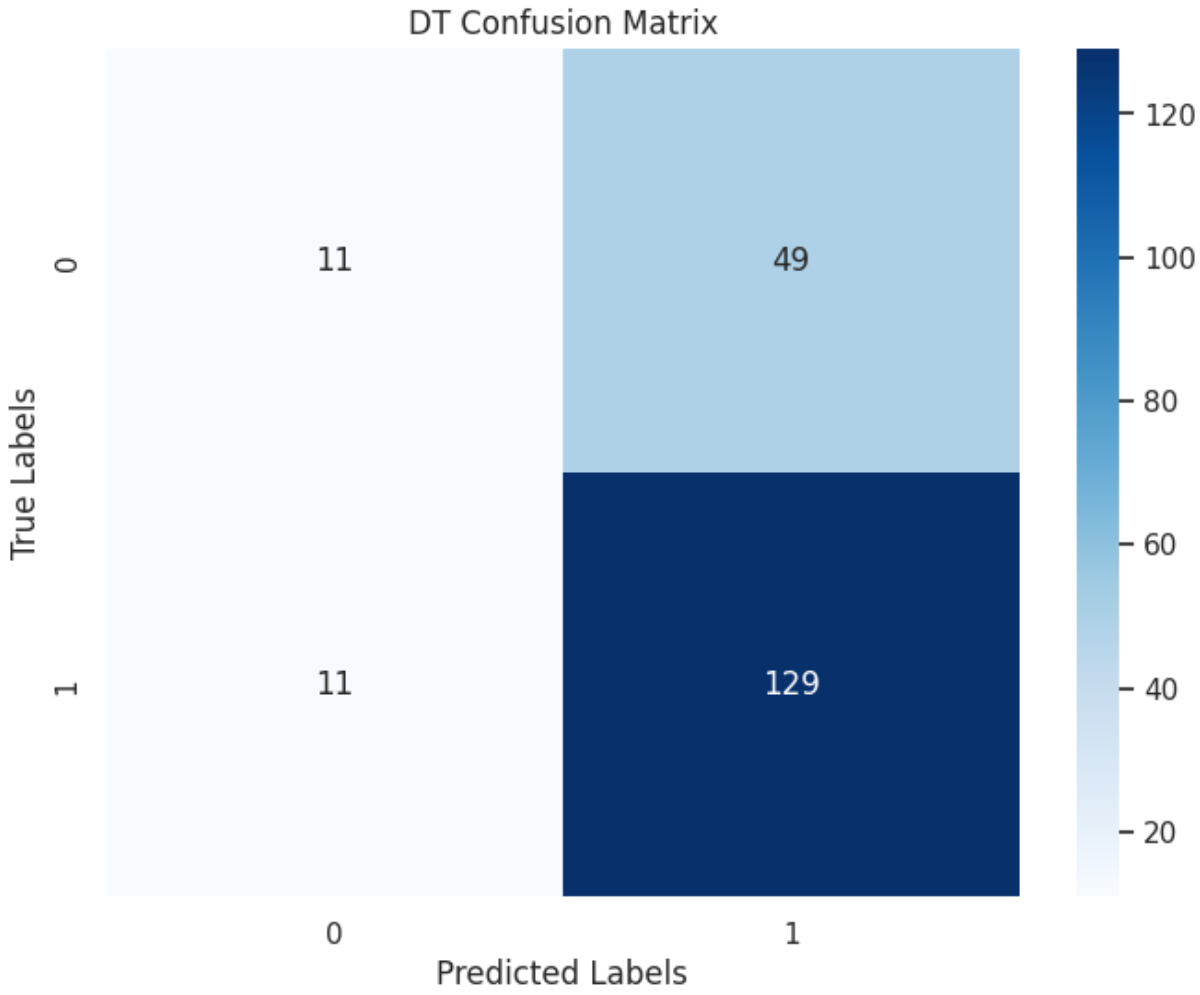


Best score is 0.71125 while best cutoff value is 0.1 and the best penalty is l2

All the models are trained and tested with details mentioned in the report.

The best algorithm for this project was the Decision Tree classification with the accuracy score of 0.7275.

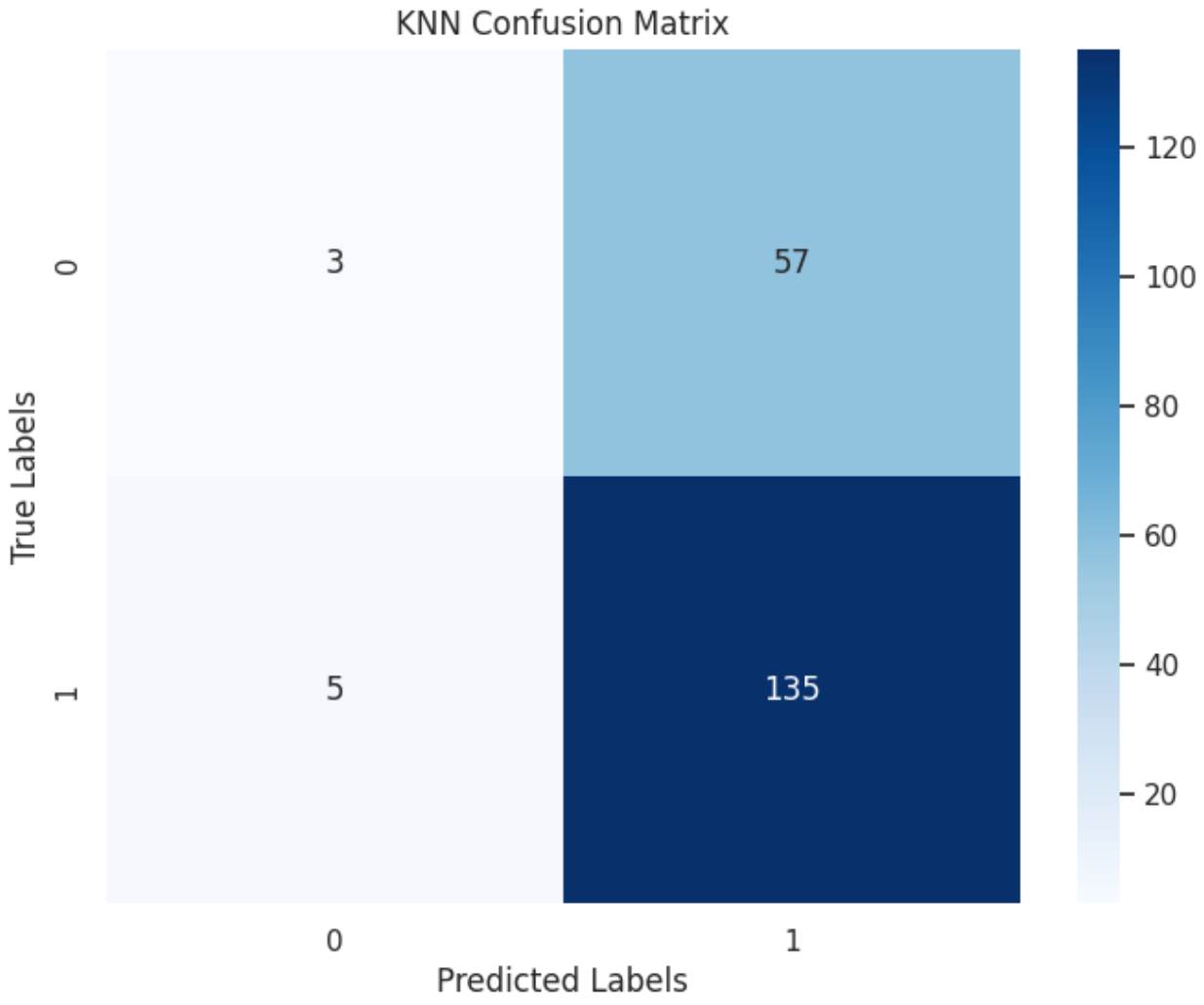
For Decision tree:



In decision tree best test score is 0.7275 while best_max_depth is 7 and best_max_leaf_nodes is 11

In this confusion matrix, there are 11 false positives and 49 false negatives. The error rate is 60/200 = 0.30

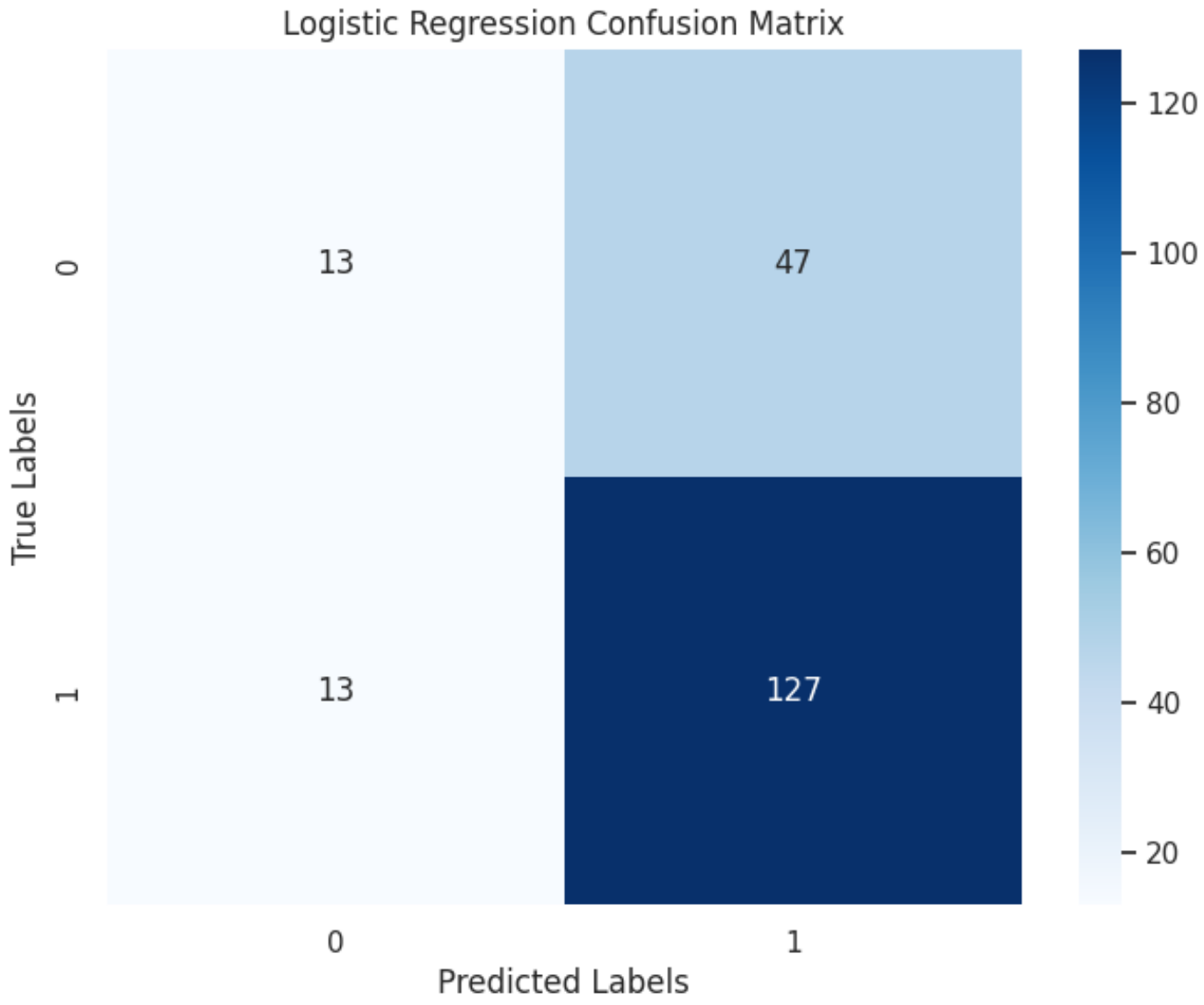
For KNN:



Using the values found during tuning in KNN, the test accuracy is 0.69

In this confusion matrix, there are 5 false positives and 57 false negatives. The error rate is 62/200 = 0.31

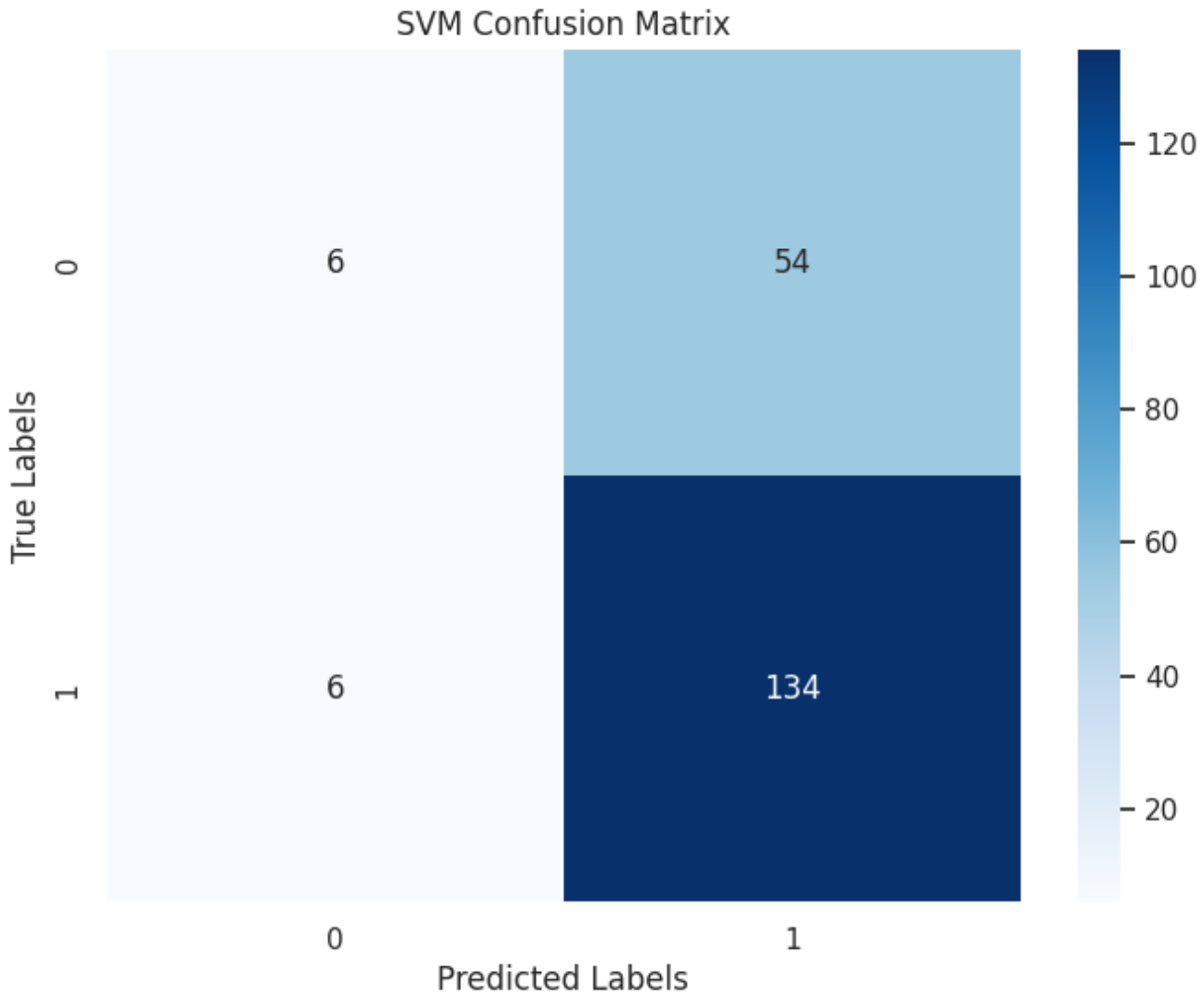
For Linear Regression:



Test accuracy score for a Logistic Regressor machine, built with the found parameters is 0.7.

In this confusion matrix, there are 13 false positives and 47 false negatives. The error rate is 60/200 = 0.30

For SVM:



Test accuracy score for a SVM machine, built with the found parameters is 0.7

In this confusion matrix, there are 6 false positives and 54 false negatives.The error rate is 60/200 = 0.30

As a bonus, SelectKBest and f_classif from sklearn.feature_selection was used in order to do feature selection.

The code in the last code block used feature_selector as SelectKBest and selected feature names according to the indices. Unnecessary columns were dropped from the newly selected X train as well as X test.

After this step, all four models that are described in detail in this report are trained and tested with the new X train and X test datasets extracted with feature selection.

As a result, the SVM classifier's accuracy is increased as opposed to the normal X train and X test datasets.

Ultimately, we have reached a consensus that the results we obtained are though naïve but desirable. Despite our predictions being very close to the base-line technique, they were able to identify good and bad applicants as precise and even more as real-world banking experts. It was surprising to see how sex, job and housing influenced how banks would classify their application. Proportionally, female people were less credible than male people; and owning a house left the bank with a good impression. Duration and Credit amount were expected to be correlated, and they were; and while you did have more expertise, you were credible. Overall, it was more than interesting to see how hard it was to make our models classify the applicant according to their information, which has always been the exact case for our hard-working banking officers.