Mario Bouzakhm

ID: 260954086

Deliverable 3 – MAIS 202 Project

Problem statement: In this project, I will be using data set from Kaggle to design a machine learning model that predicts whether a financial institution will accept or not a loan request of a client. The model used will be a binary logistic regression model.

Model Evolution:

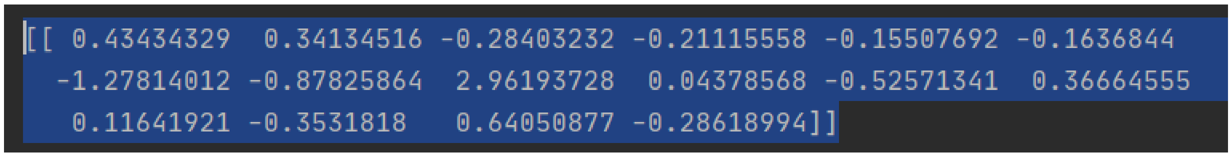
I noted the following ideas that could be implemented in this deliveable in the previous report to try and get new results:

- Try to add the median/mean value, and compare results for missing rows instead of deleting where applicable.
- Try to perform more data preprocessing, specifcally change min/max scaling to z score for Income since one of the values is really high and is offsetting the rest.
- Testing the correlation of the inputs to see if any of them can be removed/is irrelevant for our model in order to save testing time and reducing the number of features.

Adding a median/mean values to the appropriate columns where data was missing: Applicant Income, Co-Applicant Income did not make any difference. In fact, I tested this change using the same metrics I used to present my model in the report for deliverable 2: The Classification Report. The result were exactly the same. So I decided to keep the model as it was for deliverable 2.

Changing the data scaler from MinMaxScaler (provided by sklearn) to StandardScaler (provided by sklearn) gave me inferior results compared to the model presented in deliverable 2 so I abandoned this change.

The following weights were obtained from the Logistic Regerssion to study the possibility of removing some data columns in the mode:



```
[[ 0.43434329  0.34134516 -0.28403232 -0.21115558 -0.15507692 -0.1636844
  -1.27814012 -0.87825864  2.96193728  0.04378568 -0.52571341  0.36664555
   0.11641921 -0.3531818   0.64050877 -0.28618994]]
```

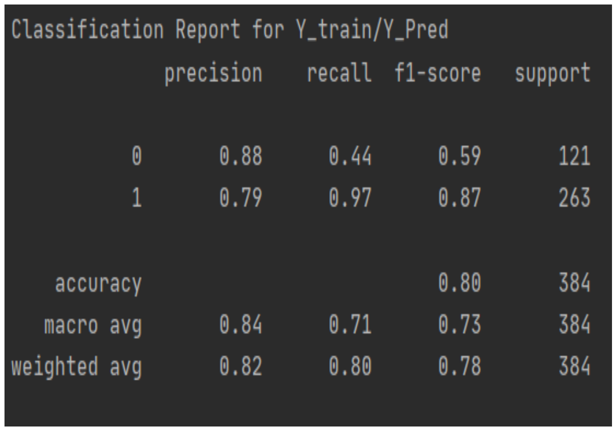
I did not find that any column had a way that was considerably smaller than the others to remove it so I kept the model as is.

However, while doing this analysis I discovered that my data processing was wrong. The OneHotEncoding for Property Area and Dependants were not correctly mapped and hence some of the classes in were not being added to my data. I have since fixed the error which was due to index mismatching when joining pandas DataFrames. This affected the performance of the model and this change will be portrayed by the content of the next section.

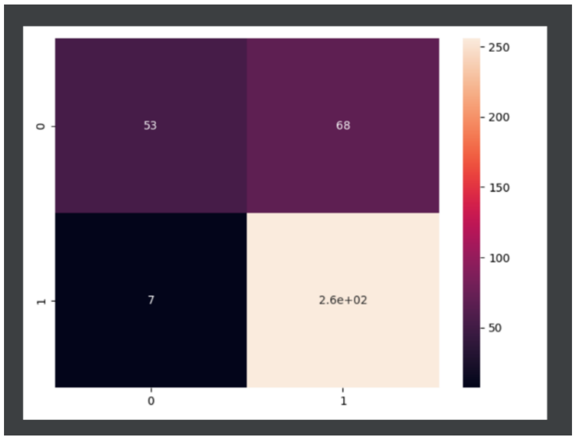
Model Final Results:

After discovering and fixing the error that was affected the data processing of the dataset, I retrained the model and obtained the following final result: (This next section will contain approximately the same content as last report since the metrics of testing were not changed since the data itself was not changed).

I tested my model on the trained data. I used the classification report provided by the sklearn module to evaluate the model. It provides me with various metrics that are relevant to classification problems. Weights were saved to be added to the application implementation of the project.



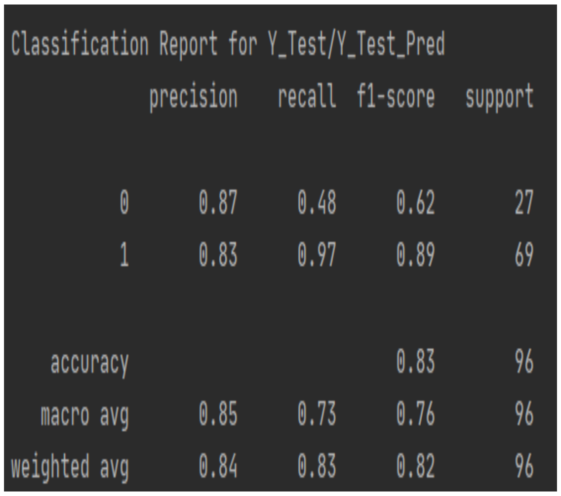
```
Classification Report for Y_train/Y_Pred
              precision    recall  f1-score   support

           0       0.88      0.44      0.59       121
           1       0.79      0.97      0.87       263

    accuracy                           0.80       384
   macro avg       0.84      0.71      0.73       384
weighted avg       0.82      0.80      0.78       384
```



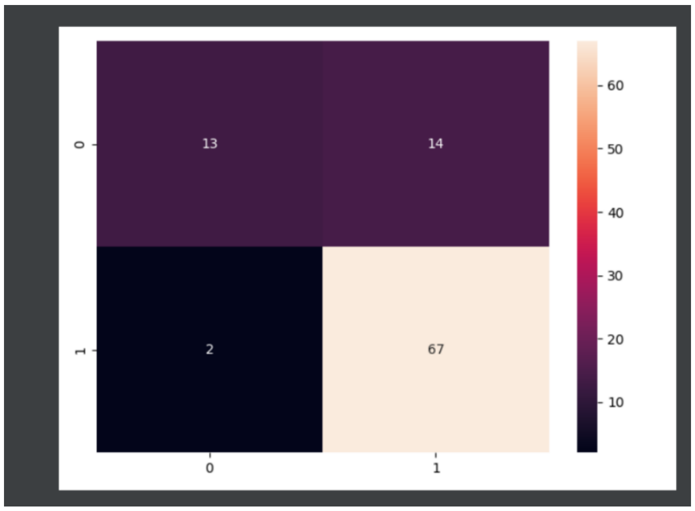
This indicates high accuracy meaning the model is accurately predicting most of its inputs. The precision/recall values especially the recall value for 0 indicates that we can have some extra tuning of the model.

I proceeded to test the model on the test data.

I found the results surprising since I obtained even better results than with the training dataset. The model is not overfitting since we are seeing similar results on both the train and test data sets. From these preliminary results, I can say that the model is feasible/effective on the problem statement. I observed a 15% error result which can be worked on to decrease during the next phase of the project.



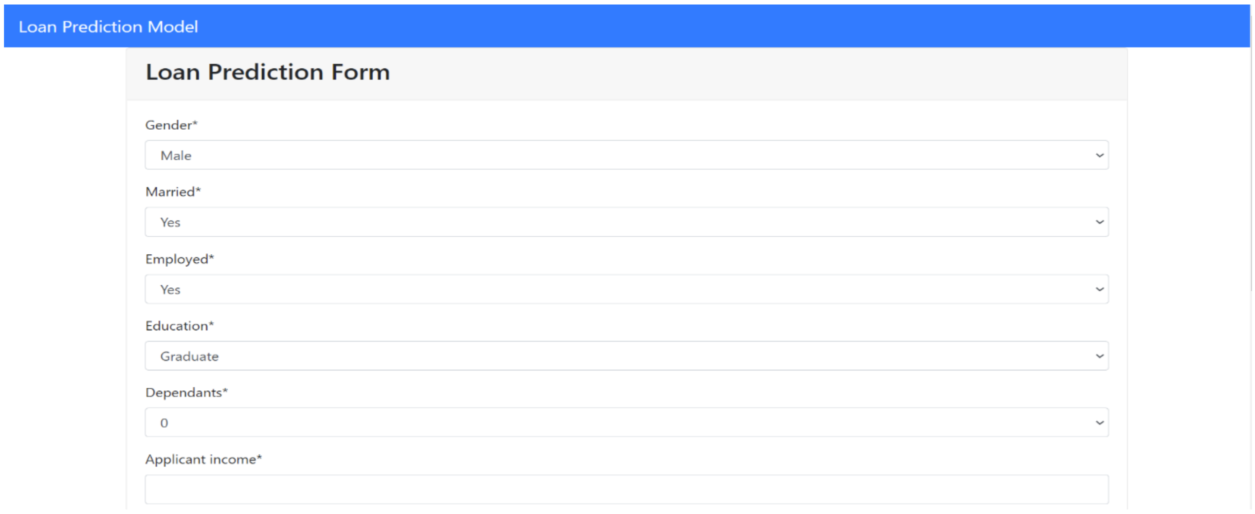
```
Classification Report for Y_Test/Y_Test_Pred
              precision    recall  f1-score   support

           0       0.87      0.48      0.62        27
           1       0.83      0.97      0.89        69

    accuracy                           0.83        96
   macro avg       0.85      0.73      0.76        96
weighted avg       0.84      0.83      0.82        96
```



Comparing with the previous report, I noticed that there was a drop in the performance of the model when the error in the data processing was fixed. However, the change was necessary to preserve the validity of the model on real-world data.

<u>Final Application:</u>

To showcase my model, I decided to create a django website and integrate the model into the website. The landing page contains a form where the user can input their data and then click on the button in the bottom of the page to get the model's prediction. Following you can find images of the website:



Loan Prediction Model

Loan Prediction Form

Gender*: Male

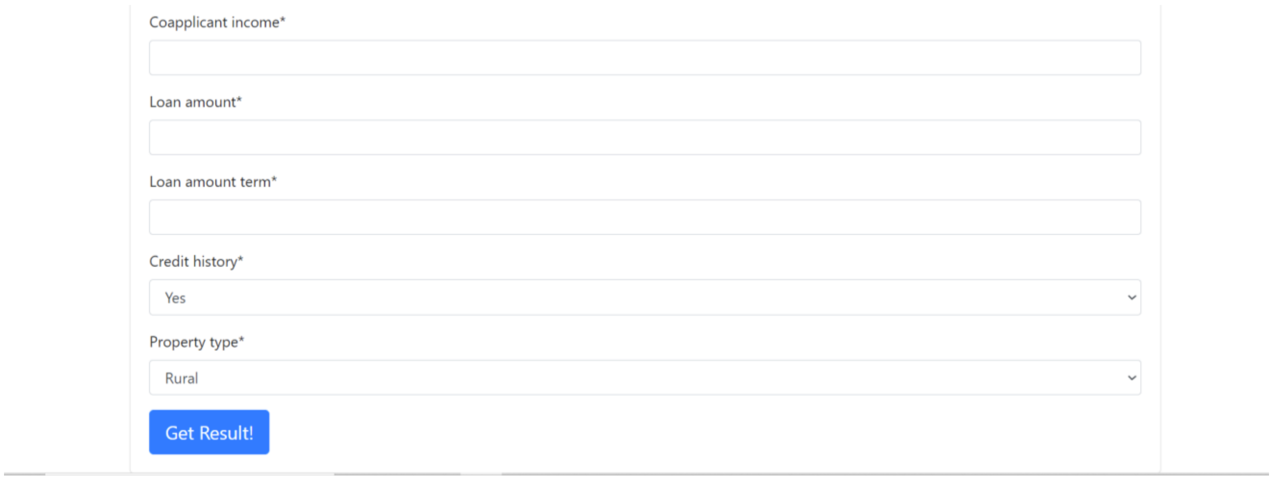
Married*: Yes

Employed*: Yes

Education*: Graduate

Dependants*: 0

Applicant income*:



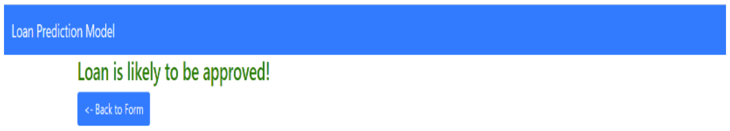
Coapplicant income*:

Loan amount*:

Loan amount term*:

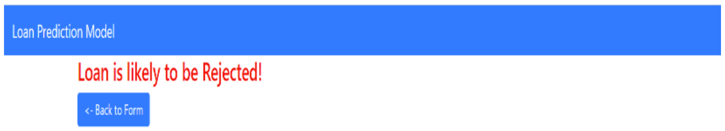
Credit history*: Yes

Property type*: Rural



Loan Prediction Model

Loan is likely to be approved!



Loan Prediction Model

Loan is likely to be Rejected!