PROJECT

MS4610 Introduction to Data Analytics

Loan Default Prediction

N Sowmya Manojna | BE17B007
Department of Biotechnology,
Indian Institute of Technology, Madras

Contents

1 Data Preprocessing

The missing values in the data were visualized using the library `missingno`. All the columns in the dataset were found to have approximately 2.5% of the data missing. The visualization of the missing data in the dataset is as follows:

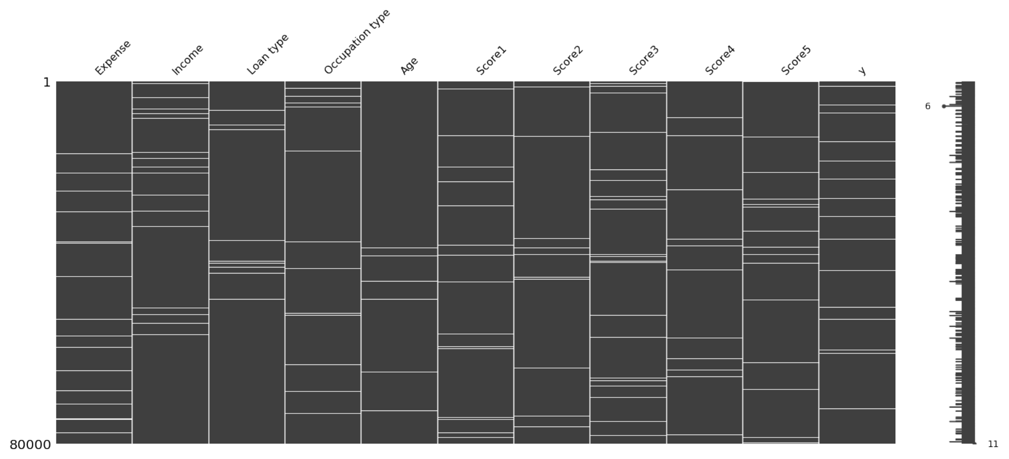

Figure 1: Visualization of missing values in the initial dataset. White horizontal bars indicate missing values.

The spread of the data was analyzed and the dataset was found to have columns with varying scales. Thus, making feature scaling an important pre-processing step.

The codes written for this section - Data Preprocessing, can be accessed in the notebook: `1_data_imputation.ipynb`

2 Data Imputation

Imputation was performed using a Random Forest based method, implemented using the `MissForest` Imputer. The categorical variables were masked and categorical imputation was performed on them. A total of 6 imputation iterations were performed to completely fill all the missing values in the dataset. The visualization of the dataset after imputation is as follows:

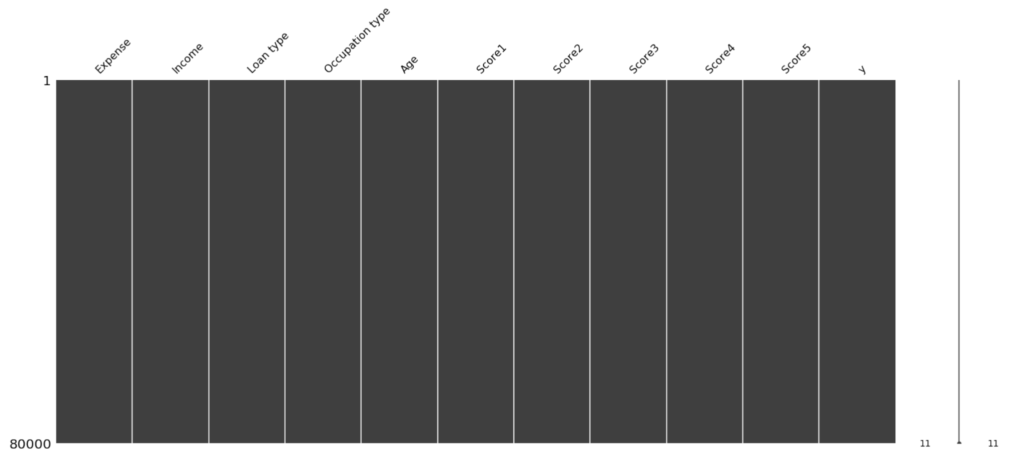

Figure 2: Visualization of missing values in the imputed dataset.

The codes written for this section - Data Imputation, can be accessed in Notebook: `1_data_imputation.ipynb`

3 Feature Selection

One-hot encoding of the features - `Loan type`, `Occupation type` was done after imputing the dataset and the first category was dropped after the encoding, to ensure that the resulting encoded features are linearly independent.

3.1 Correlation

The correlation between the features was calculated and visualized using heatmaps and a cut-off of 0.75 was used to identify all highly correlated features. The heatmap obtained is as follows:

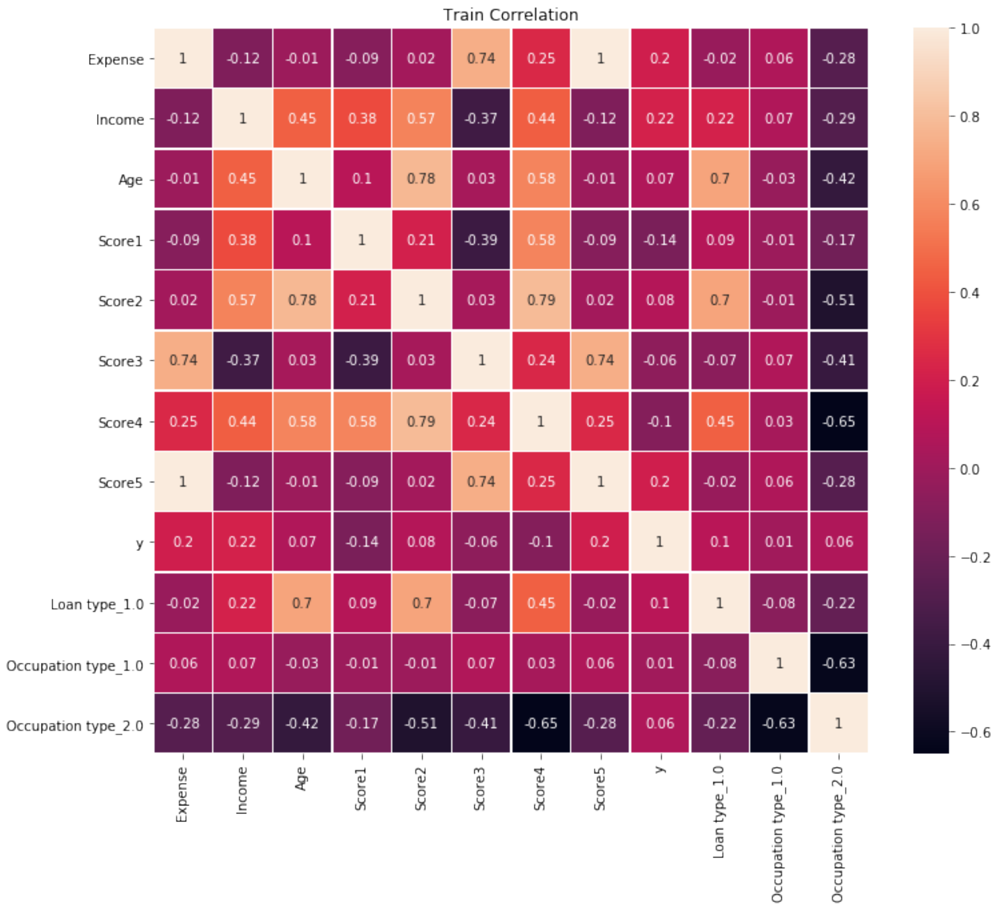

Figure 3: Correlation between the features present in the training dataset.

The highly correlated features in the dataset were:

Feature 1	Feature 2	Correlation
Expense	Score5	1.000000
Score2	Score4	0.786452
Age	Score2	0.780841

Table 1: List of Highly Correlated features in the training dataset.

From Table 1 above, it is seen that the feature - `Expense` & `Score5` are completely correlated and that the features `Score2` & `Score4` and `Age` & `Score2` are highly correlated.

In order to deal with features that have high correlation, the features in the `Feature 1` column in Table 1, were sequentially removed and the resulting dataset was checked for correlated features that have a correlation greater than 0.75. In this process, the features `Expense` and `Score2` were removed. The correlation heatmap after the feature selection is as follows:

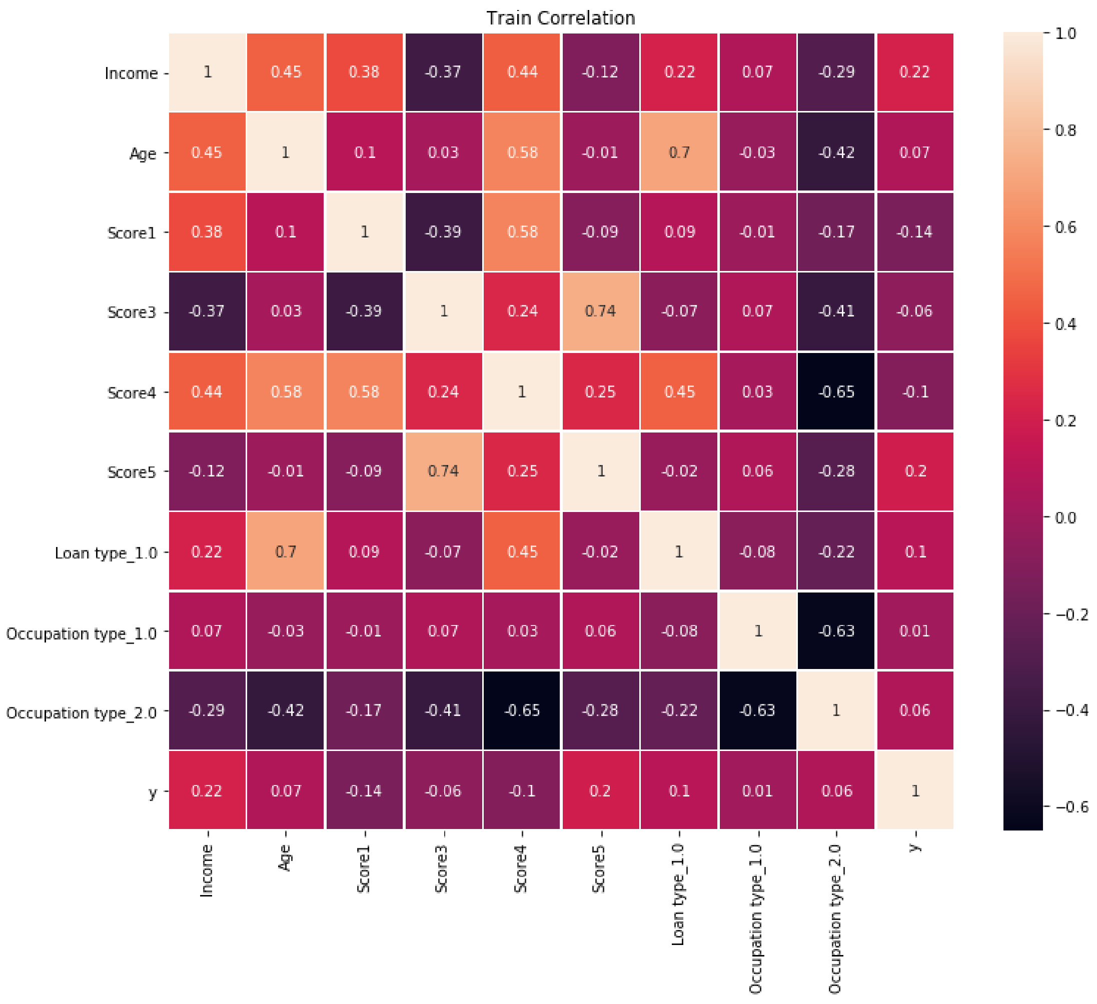

Figure 4: Correlation between the features present in the training dataset, after removing highly correlated features.

3.2 Multicollinearity

Variance Inflation Factor (VIF) was used to calculate the multicollinearity between the features in the dataset. The VIF of the initial dataset is as follows:

Features	VIF Factor	Features	VIF Factor
Score5	9.051581e+07	Score3	1.894692e+03
Score4	5.462307e+07	Occupation type_2.0	1.282768e+01
Expense	5.940019e+06	Occupation type_1.0	7.769822e+00
Score2	3.707871e+04	Loan type_1.0	5.941420e+00
Score1	3.267527e+03	Age	5.783687e+00
Income	2.267588e+03		

Table 2: Variance Inflation Factor of the features in the initial dataset.

From Table 2 it is seen that `Score5`, `Score4` and `Expense` have a high VIF, a trend similar to what was observed in Table 1. The VIF values after the feature selection are as follows:

Features	VIF Factor	Features	VIF Factor
Score5	8969.040865	Occupation type_1.0	7.260461
Score4	6982.889621	Score1	6.515382
Income	569.689564	Age	5.519235
Score3	18.068466	Loan type_1.0	4.289129
Occupation type_2.0	11.882027		

Table 3: Variance Inflation Factor of the features in the final dataset, after feature selection.

The codes written for this section - Feature Selection, can be accessed in the notebook: `1_data_imputation.ipynb`

4 Data Generation

The dataset was highly imbalanced - with 5247 transactions labeled as defaulted out of $80,000$ transactions. As, several Machine Learning methods are sensitive to imbalanced classes, Synthetic Minority Oversampling Technique (`SMOTE`) [1] was used to generate synthetic data for the defaulted transaction class, so that both classes have the same number of samples.

Prior to the `SMOTE` data generation, the dataset was split into the train set and the validation set in the ratio $3 : 1$. The samples for class 1, in the train dataset alone were generated using `SMOTE`. The distribution of samples before and after the `SMOTE` data generation is as follows:

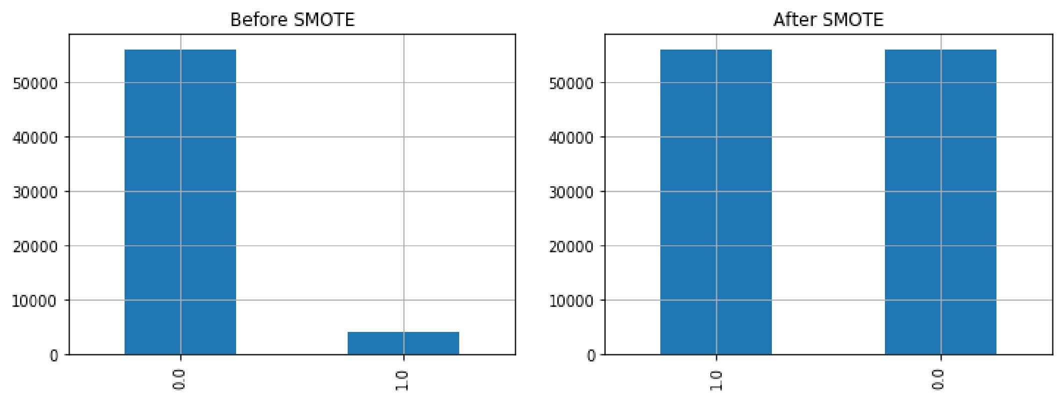

Figure 5: Sample distribution before and after `SMOTE` data generation. Synthetic data was generated for class 1, so that the number of samples in both classes are the same.

The codes written for this section - Data Generation, can be accessed in the notebook: `2_data_generation.ipynb`

5 Parameter Tuning

Parameter tuning was carried out using Grid Search with a 5-fold Cross Validation and `F1 Score` as the scoring function. The parameter tuning was carried out for the following ML models:

- Multi Layered Perceptron (MLP)
- Multi Layered Perceptron (MLP), with Standardized Input Dataset
- Random Forest (RF)
- AdaBoost with Decision Trees as the base classifier.
- Support Vector Machines (SVM)
- Support Vector Machines (SVM), with Standardized Input Dataset
- K-Nearest Neighbors (KNN)
- K-Nearest Neighbors (KNN), with Standardized Input Dataset
- Decision Trees Classifier
- SGD Classifier (linear SVM with Stochastic Gradient Descent)
- Logistic Regression

The parameters that resulted in the best mean `F1` score were saved and used on the validation dataset. The tuned parameters for the models can be accessed in the `parameter_search` folder.

The performance of the models on the training data, with the best parameter set returned is as shown below:

Model	F1 Score	Accuracy	Model	F1 Score	Accuracy
SGD	80.22	81	KNN	94.59	98
MLP	85.16	87	Decision Tree	97.74	100
Logistic	87.42	88	KNN_Scaled	98.31	99
AdaBoost	92.66	93	RF	98.62	100
AdaBoost	92.66	93	MLP_Scaled	98.73	99
SVM	93.34	98	SVM_Scaled	98.73	98

Table 4: Performance of the models using the parameters that returned the best results on the training data.

From Table 4, we can see that the MLP model with Standardized inputs, Random Forest model, KNN model with Standardized inputs, Decision tree model and SVM with Standardized inputs have performed with an `F1 score` greater than 95%.

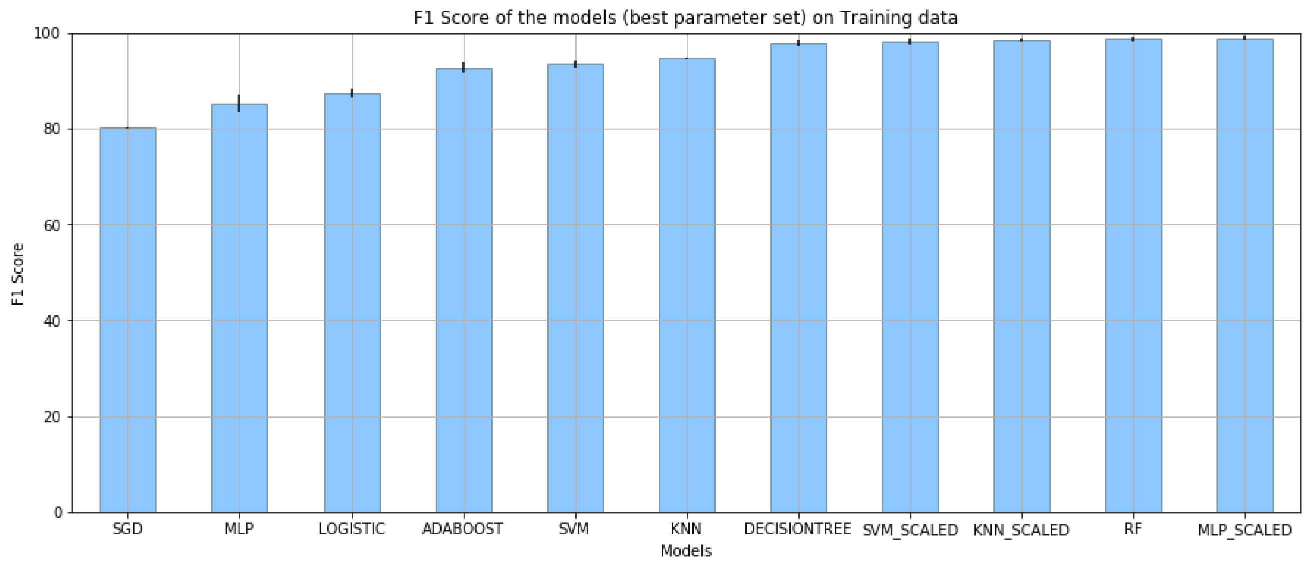

Figure 6: Performance of the models on the training data using the best tuned parameter set returned from GridSearchCV.

The codes written for this section - Parameter Tuning, can be accessed in the notebook: `3_gridsearch.ipynb` and the results of the parameter tuning can be accessed in the folder: `parameter_search/`.

6 Model Selection

All the models above, with the trained parameters and some additional models such as - `GaussianNB`, `BernoulliNB`, `LDA` (Linear Discriminant Analysis) and `ExtraTreesClassifier`, were implemented on the Validation dataset. The performance of the models is as follows:

Model	Precision	Recall	F1-Score	Accuracy
MLP Classifier, with StandardScaler	90.0	91.0	91.0	99.0
SVM, with Standard Scaler	87.0	89.0	88.0	98.0
SVM	88.0	89.0	88.0	98.0
ExtraTreesClassifier	90.0	85.0	87.0	98.0
Random Forest Classifier	87.0	86.0	86.0	98.0
KNN, with StandardScaler	80.0	87.0	83.0	98.0
Decision Tree Classifier	73.0	85.0	79.0	97.0
Gradient Boosting, with StandardScaler	72.0	85.0	78.0	97.0
Gradient Boosting	72.0	85.0	78.0	97.0
AdaBoost, with StandardScaler	50.0	80.0	62.0	93.0
AdaBoost	50.0	80.0	62.0	93.0
GaussianNB	41.0	78.0	54.0	91.0
LDA, with StandardScaler	41.0	71.0	52.0	91.0
LDA	41.0	71.0	52.0	91.0
KNN	40.0	69.0	50.0	91.0

SGDClassifier, with StandardScaler	34.0	78.0	47.0	88.0
MLP Classifier	31.0	80.0	45.0	87.0
SGDClassifier	24.0	79.0	37.0	82.0
BernoulliNB	8.0	57.0	15.0	56.0

Table 5: Performance of the models, with the best parameters returned from `GridSearchCV` on the Validation dataset.

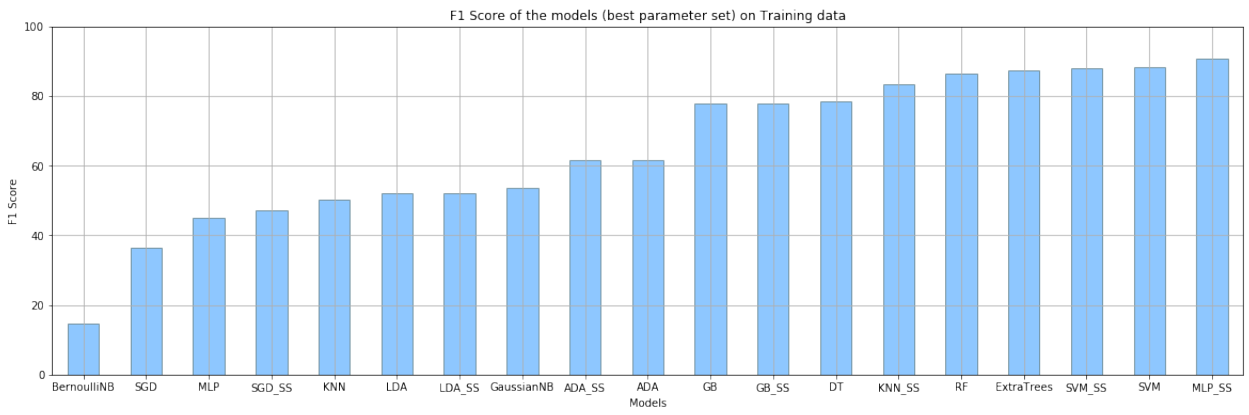

Figure 7: Performance of the models with the best tuned parameters, on the validation data. `ADA` refers to `AdaBoost`, `GB` refers to `GradientBoosting`, `DT` refers to `DecisionTrees` and `_SS` refers to Standardized Inputs.

From Table 5 and Figure 7, it is noted that the `F1-Score` of the models - `KNN_SS`, `RF`, `SVM_SS`, `ExtraTrees` and `MLP_SS` is greater than 85%.

The ROC Curves of the models that have `F1−Score` greater than 85% are as follows:

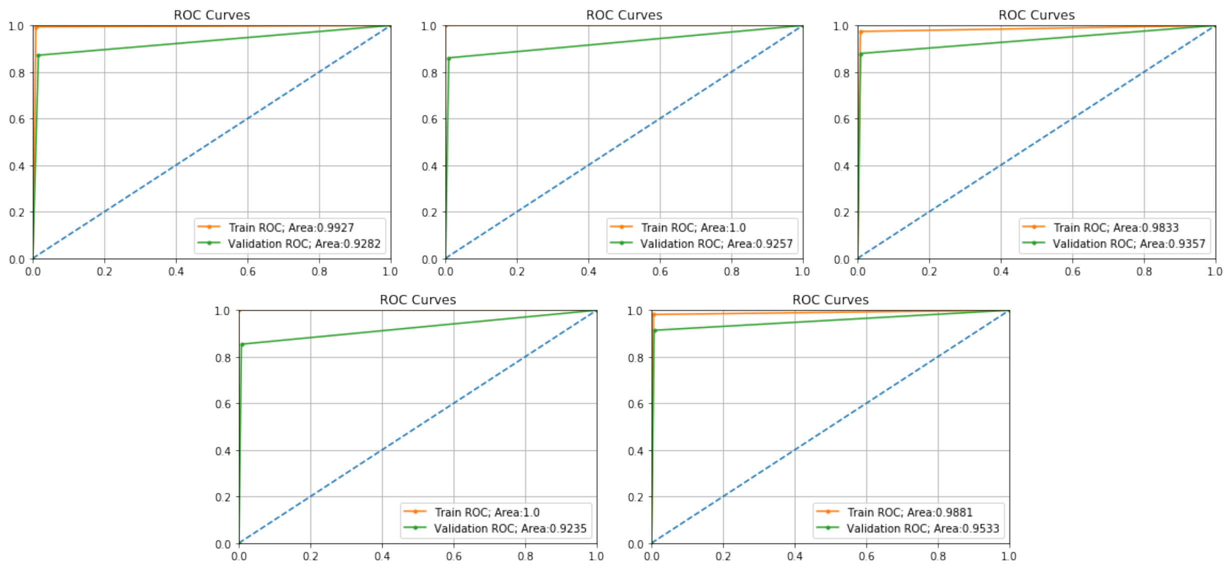

Figure 8: ROC Curves of the models with F1-Score greater than 85%, in the order: KNN with Standardized Input, Random Forest, SVM with Standardized Input, ExtraTrees Classifier and MLP with Standardized Input.

The codes written for this section - Model Selection, can be accessed in the notebook: `4_validate_models.ipynb`

7 Final Model - Stacking Classifier

The models that gave an F1-Score greater than 85% were chosen and ensemble classification was done using these chosen models. The ensemble classification methods used are - `StackingClassifier` with the base model as MLP with Scaled input and `VotingClassifier` with both hard and soft voting. The results obtained are as follows:

Model	Precision	Recall	F1-Score	Accuracy
Stacking Classifier	91.0	90.0	91.0	99.0
Voting Classfier (Soft)	89.0	89.0	89.0	99.0
Voting Classfier (Hard)	89.0	88.0	88.0	98.0

Table 6: Performance of ensemble models, with the models that returned an F1-Score greater than 85% on the Validation dataset.

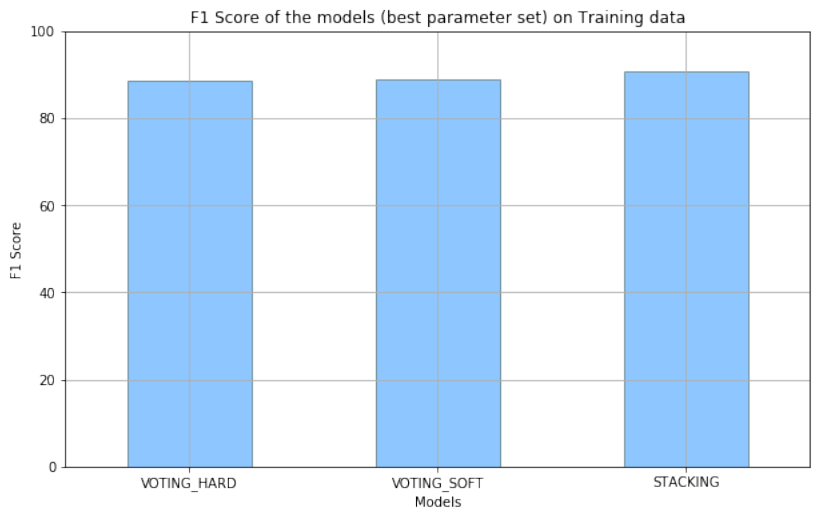

Figure 9: Performance of ensemble models with the models that gave accuracy higher than 85%.

As the `StackingClassifier` has given the best performance, it has been chosen as our final model. The final model has been pickled and saved as `finalized_model.sav`. Pre-processed test dataset was passed to our model and our predicted result has been saved at `datasets/pred_y.csv`.

The ROC Curves of the Ensemble Models are as follows:

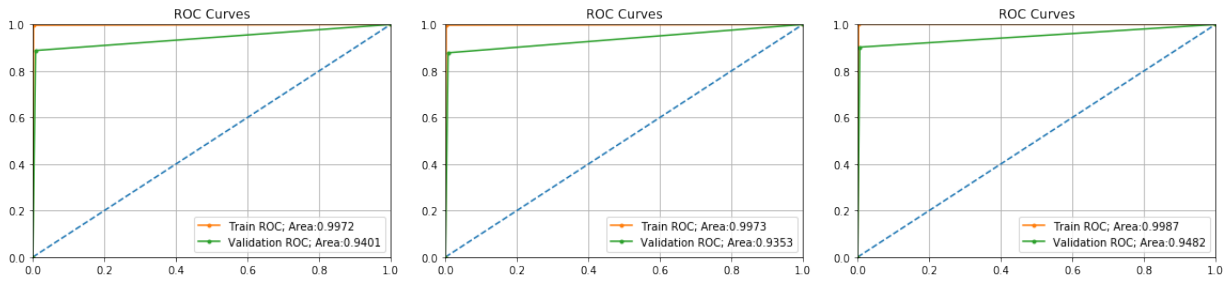

Figure 10: ROC Curves of the ensemble models, in the order: Voting Soft, Voting Hard and Stacking Models.

The codes written for this section - Model Selection, can be accessed in the notebook:
`4_validate_models.ipynb`

8 Computational Resources

Most of the analysis has been powered by an Intel Core i5 CPU, 64-bit computer laptop, with Ubuntu 20.04 Operating System. The computationally intensive SVM GridSearch was carried out using GPU support offered by Google Colab.

8.1 Programming Environment

All the codes were run using Jupyter Lab (Jupyter version: 4.6.3, Jupyter Notebook version: 6.0.3, Jupyter Lab version: 2.2.9, IPython version: 7.19.0).

The version of all the libraries used has been included in the `requirements.txt` file. The libraries can be installed using: `pip3 install -r requirements.txt`.

8.2 Reproducibility

The codes written as a part of this course project can also be accessed in the following GitHub Repository: MS4610 Introduction to Data Analytics Project.

In case of any notebook rendering problem, please upgrade the Jupyter notebook version or view the online rendered version on GitHub or nbviewer (using the GitHub URLs).

References

[1] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.