30412 Machine Learning Final Project

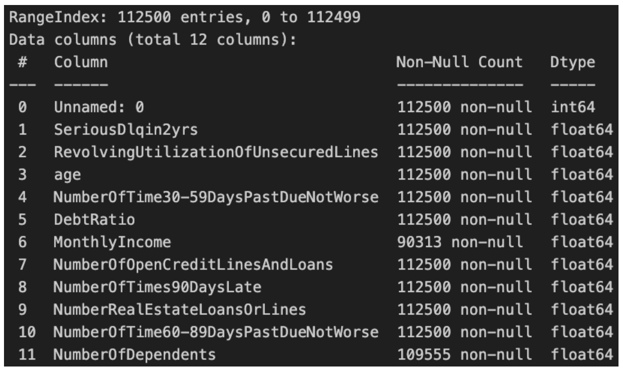
Marco Maluf (3296117), Konstantino Haitas (3112866)[1]

[1]*Machine Learning, 30412; Final Project*

The goal of the project was to create the simplest model with the highest AUC score to predict whether someone experienced 90 days past due delinquency or worse or not. We wanted to use the techniques mentioned in lecture like logistic regression, bagging classifiers, and random forests as we thought those would perform the best and be the most interpretable models. We used certain data cleaning and preprocessing methods from select academic papers.

I. DATA CLEANING AND PREPROCESSING

After downloading the data.zip from blackboard, we loaded in the datasets in python using Pandas and started by looking at the info of the imported data.



```
RangeIndex: 112500 entries, 0 to 112499
Data columns (total 12 columns):
 #   Column                                Non-Null Count   Dtype
---  ------                                --------------   -----
 0   Unnamed: 0                            112500 non-null  int64
 1   SeriousDlqin2yrs                      112500 non-null  float64
 2   RevolvingUtilizationOfUnsecuredLines  112500 non-null  float64
 3   age                                   112500 non-null  float64
 4   NumberOfTime30-59DaysPastDueNotWorse  112500 non-null  float64
 5   DebtRatio                             112500 non-null  float64
 6   MonthlyIncome                         90313 non-null   float64
 7   NumberOfOpenCreditLinesAndLoans       112500 non-null  float64
 8   NumberOfTimes90DaysLate               112500 non-null  float64
 9   NumberRealEstateLoansOrLines          112500 non-null  float64
 10  NumberOfTime60-89DaysPastDueNotWorse  112500 non-null  float64
 11  NumberOfDependents                    109555 non-null  float64
```

FIG. 1: Summary of data information, number of non-null entries for each feature.

As far as cleaning the data, we decided to drop the 'Unnamed: 0' column and made all the column names lowercase for ease of programming later down the line.

Noticing the large number of null entries in the 'monthlyincome' and 'numberofdependents' features, we realized that we might need to impute data to remedy the missing entries of the data set. There was a clear difference between the mean and standard deviation of the variable 'debtratio' between observations with NaN 'monthlyincome' entries and those without. Becuase of this, the missingness could not be classified as MAR (missing at random) or MCAR (missing completely at random), and therefore the best way to impute missing data was with Single Value Imputation.

Single Value Imputation (SVP) involves imputing a single value for the missing data points based on a robust statistic such as the mean or median. Due to the apparent correlation between 'debtratio' and the missingness present in 'monthlyincome', I will do mean imputation based on the following logic:

$$X_{impute} = \mathbb{E}_{NA}[debtratio] * 246.2368,$$

where 246.2368 is derived by the following ratio:

$$\frac{\mathbb{E}_{notNA}[monthlyincome]}{\mathbb{E}_{notNA}[debtratio]} = 246.2368$$

There are also missing values for the variable 'numberofdependents', which will be imputed with 0 since if someone did not provide the number, it is safe to assume it is 0.

We also took a second approach to imputation, using Iterative Model-Based Imputation (IMBI). We used the IterativeImputer class from the scikit-learn library, which allows one to impute missing values using a sequence of estimators (inspiration for the technique comes from 2 papers which are cited at the end of the document).

II. MODEL BUILDING AND SELECTION

Logistic Regression

We first decided to fit two Logistic Regressions, one on the SVP data set and the other on the IMBI data set. They both performed similarly well, with the one trained on the IMBI data set performing better. We decided to use the IMBI data set for the rest of the models.

Decision Tree, Bagging Classifier

After fitting two Logistic Regressions, we decided to try using a Decision Tree Classifier. We started with a single tree and we saw that the performance was worse than both the Logistic Regression models, but we know that performance of a single tree can vary tremendously. Using that knowledge and being inspired from the lecture on Ensemble Methods, we decided to use a Bagging Classifier. We ran code to test performance on a segment of the test set using a different number of base estimators, and we found that 400 estimators yielded the best performance.

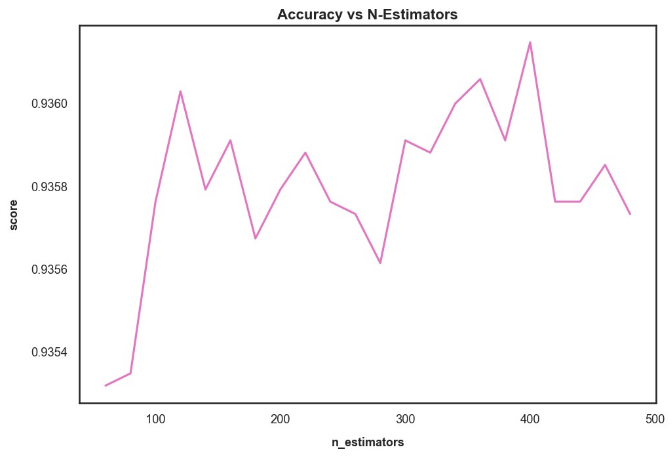


FIG. 2: Accuracy score of the Bagging Classifier with different numbers of base estimators. n_estimators = 400 was the optimal number of estimators.

The Bagging Classifier with 400 base estimators significantly outperformed the previous models.

Random Forest Classifier

After fitting the Bagging Classifier with 400 base estimators, we decided to use a Random Forest Classifier with the same number of base estimators to see if a reduced effect of correlation within the base estimators would yield better results. After fitting the model, we saw a marginal increase over the Bagging Classifier. We decided to use it as our final model and based our submission off the predictions from this model.

III. RESULTS

Accuracy Scores and AUC Scores

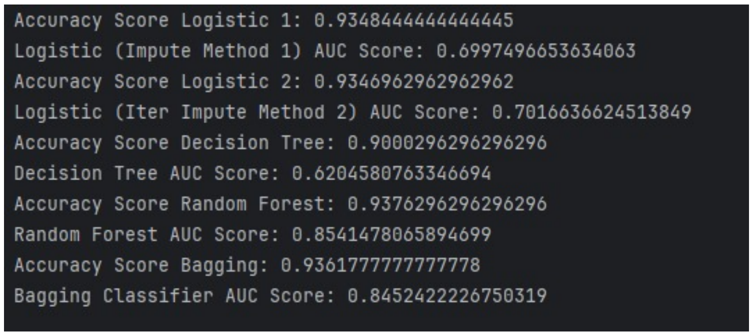
The model that showed the worst performance was the single Decision Tree, which when trained on the IMBI data set had an accuracy score of 0.9000 and an AUC of 0.6205.

Following the single Decision Tree was the first Logistic Regression on the SVP data set, with an accuracy score of 0.9348 and an AUC of 0.6997.

The next best model was the second Logistic Regression on the IMBI data set, with an accuracy score of 0.9347 and an AUC of 0.7017.

The second best model was the Bagging Classifier with 400 base estimators, which when trained on the IMBI data set had an accuracy score of 0.9362 and an AUC of 0.8452.

The final and best model was the Random Forest Classifier with 400 base estimators, which when trained on the IMBI data set had an accuracy score of 0.9376 and an AUC of 0.8541.



```
Accuracy Score Logistic 1: 0.9348444444444445
Logistic (Impute Method 1) AUC Score: 0.6997496653634063
Accuracy Score Logistic 2: 0.9346962962962962
Logistic (Iter Impute Method 2) AUC Score: 0.7016636624513849
Accuracy Score Decision Tree: 0.9000296296296296
Decision Tree AUC Score: 0.6204580763346694
Accuracy Score Random Forest: 0.9376296296296296
Random Forest AUC Score: 0.8541478065894699
Accuracy Score Bagging: 0.9361777777777778
Bagging Classifier AUC Score: 0.8452422226750319
```

FIG. 3: Accuracy scores and AUC scores of every model trained.

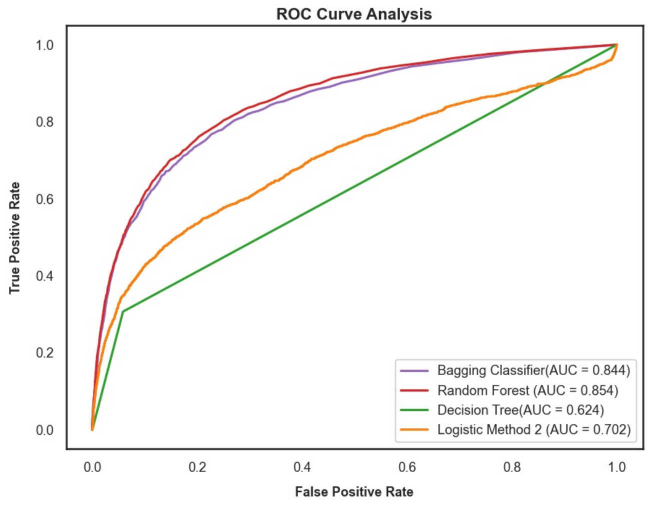


FIG. 4: ROC Curve of the Bagging Classifier, Random Forest Classifier, the single Decision Tree, and the second Logistic Regression with reported AUC scores.

Final Insights

Starting with simpler models and building our way up to more complex ones proved to be a robust and effective way to build a very accurate predictor without sacrificing interpretability. With more time, we would have liked to have been able to train increasingly more complex models, or try different approaches such as boosting or implementing soft and hard margin SVMs.

IV. BIBLIOGRAPHY

Where we got the Iterative Model-Based Imputation idea from:
`https://link.springer.com/chapter/10.1007/978-3-319-55723-6_1#Sec15`
`https://arxiv.org/abs/2110.11951`