Lending Club
Loan Default Classification

Aidyn Kadyr. ROBT 407 Final Project

***Abstract*—This report introduces machine learning models that aim to predict whether a customer of a bank will be able to repay the loan. The classification problem contains customer's information as features and 0/1 target variable. Machine learning algorithms, such as Linear and Quadratic Discriminant Analysis, Logistic Regression and AdaBoost are implemented. As a result of feature selection, hyperparameter tuning and cross validation, the best performance was achieved by the AdaBoost model of 800 estimators and learning rate of 1 with the ROC AUC score of 0.84418.**

I. Introduction

Classification of a credit worthiness is a significant part of any banking system. It allows a bank to evaluate whether to provide a customer a loan, if yes, what is the maximum amount and how much interest rate to charge. The All Lending Club Loan data (link) contains information for customers e.g. their age, credit amount, duration and job. The purpose of this project is to classify the customers into two groups: those who will repay and those who will fail to repay the loan. The data set contains 2 260 701 observations and 151 columns.

II. Methods

A. Data Preprocessing

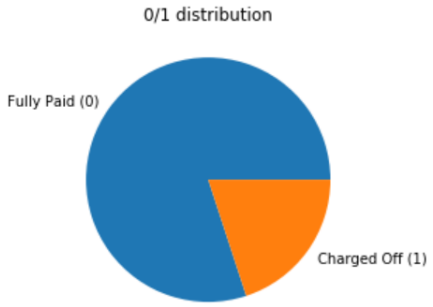
In practice, raw data is almost never perfect. It may contain missing values, mixed data types or values with a large range compared to other variables. In this step data is cleaned and organized in an understandable way, such that it will result in a better performance and better interpretability of the model.

1) Target Variable: After importing the libraries and reading the data, the target variable 'loan status' which contains 9 different categories with different distribution, we only consider the loan being 'Fully Paid' or 'Charged Off', which means a customer will be able to repay the loan or not.

```
df['loan_status'].unique() # Values that the target variable takes
```

```
array(['Fully Paid', 'Current', 'Charged Off', 'In Grace Period',
       'Late (31-120 days)', 'Late (16-30 days)', 'Default', nan,
       'Does not meet the credit policy. Status:Fully Paid',
       'Does not meet the credit policy. Status:Charged Off'],
      dtype=object)
```

This way, the classification problem becomes binary classification. However, the pie chart shows an uneven distribution of labels, which makes the problem unbalanced. This issue is solved later by removing some data points for 'Fully Paid'



2) Data cleaning: The null or missing values are filled with corresponding columns' mean or mode value. Values such as '10 years' and 'less than 1 year' are converted to float numbers by replacing with '10' and '0' respectively.

3) Categorical Variables: Values for categorical variables, 'sub grade','home ownership','verification status' and 'purpose', are replaced with integers by using LabelEncoder from sklearn.preprocessing library. It assigns values from 0 to k-1 to each category, where k is the number of unique values in that column.
Some categorical variables dropped based on intuitive explanation. For example, 'zip code', 'id' and 'address' are irrelevant for prediction and contain too many values. Discarding those does loose information.

B. Exploratory Data Analysis

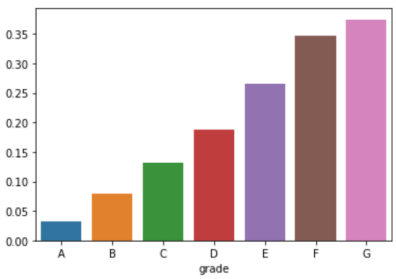
1) Correlation Analysis: Features which are highly correlated with each other are dropped from prediction. For variables correlated with the target variable, two least correlated features are dropped:

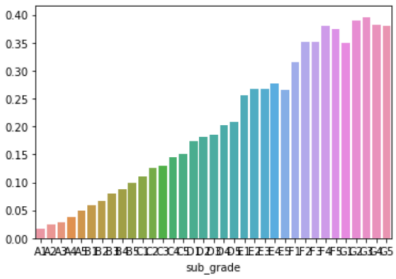
```
todrop = df2.corr()['Charged Off'].abs().sort_values(ascending = True)[0:2].index
todrop
```

```
Index(['total_acc', 'open_acc'], dtype='object')
```

```
df2 = df2.drop(todrop, axis = 1) # drop the 2 least correlated features.
```

The effect of loan's grade and sub grade on the target is the same from the bar charts below:





As a result the 'grade' column was dropped as it contains more information.

The correlation between the following two variables is 1, hence one can be dropped:

```
df2[['fico_range_high', 'fico_range_low']].corr()
```

	fico_range_high	fico_range_low
fico_range_high	1.0	1.0
fico_range_low	1.0	1.0

Data Preprocessing and Exploratory Data Analysis facilitated feature selection and data cleaning. Starting with 151 columns, the final number of features is 27. Initially there were 1 076 751 Fully Paid and 268 559 Charged Off loans. To handle the unbalanced data, the number of Fully Paid loans was taken the same as the number of Charged Off loans, since there is enough data available. As a result, there are 537 118 observations for 27 features. Functions, such as .info() and .isnull() were called after every step of data preprocessing to make sure there are no null values and features take numerical values, i.e. float and integer datatypes.

C. Evaluation metrics

Common evaluation metric for binary classification problems is *accuracy*. It is simply the proportion of misclassified examples. However, one should note that there are two types of misclassifications, i.e. two types of errors:
Type I Error: False Positive (FP). Predicted output is 1, while true value is 0.
Type II Error: False Negative (FN). Predicted output is 0, while true value is 1.
Hence, to account for these kind of erros, we introduce concepts of **Precision, Recall, F1 Score and ROC AUC score**.

$$\textbf{Recall} = \frac{TP}{TP+FN}, \textbf{Precision} = \frac{TP}{TP+FP}$$

F1 Score is the harmonic mean of Recall and Precision:

$$\textbf{F1} = 2\frac{Precision * Recall}{Precision + Recall}$$

AUC stands for the Area Under The Curve and **ROC** for Receiver Operating Characteristics. The ROC AUC metric measures how well the model separates the two classes based on changing the threshold value. The range of this metric is between 0 and 1. The higher the value the better.
We will mainly focus on the ROC AUC score and then F1 score. Relevant literature on ROC AUC is provided in the reference.

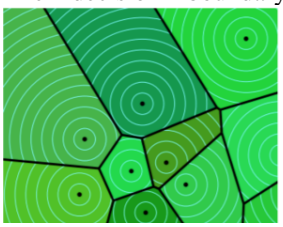
D. Model Selection

The data is split into Training and Testing sets using sklearn.modelselection library. The target variable y is a binary 0/1 'Charged Off' feature, while X is a dataframe of integer and float features. The test size taken as 0.2 and random state 66.
Data is fitted into StandardScaler from sklearn.preprocessing library. The Standard Scaler normalizes the values such that each column has a mean of 0 and unit variance. This is done to set features of different units to a similar scale.

1) Linear Discriminant Analysis (LDA): LDA is a type of generative classifiers which models the joint distribution of target and features and tries to maximize its probability. It assumes independence between classes but equal covariance matrices.

$$X \sim \mathcal{N}(\mu_k, \Sigma)\,.$$

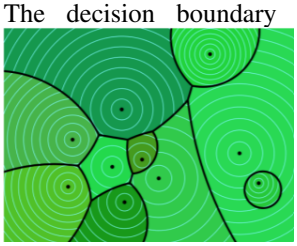
The decision boundary in case of LDA is linear.



2) Quadratic Discriminant Analysis (QDA): QDA is another type of generative classifiers. In contrast to the LDA, QDA assumes different covariance matrices for different classes. Hence the conditional distribution is given by

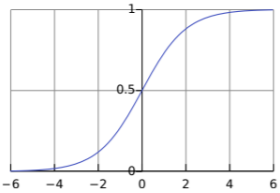
$$X \sim \mathcal{N}(\mu_k, \Sigma_k)\,.$$

The decision boundary in case of QDA is quadratic.



3) Logistic Regression: Logistic Regression for Binary classification produces a linear combination of weights and observations and squishes it into a range from 0 to 1 using the sigmoid function:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



The output is then interpreted as the probability of belonging to class 1. Setting threshold to 0.5, class 1 is predicted if $\sigma(w^T x) \geq 0.5$ and class 0 is predicted if $\sigma(w^T x) < 0.5$ The model is trained

by minimizing the cross-entropy loss, motivated by the Maximum Likelihood Estimator (MLE)

4) AdaBoost: Adaptive boosting is a boosting technique that combines several weak models in a more sophisticated way than in ensemble methods, where each weak model has the same weight. Adaboost minimizes the exponential loss $e^{-y\hat{y}}$ assigning much higher penalty (exponential) for incorrect classification.

III. Results

1) Model Training and Model Selection: First, the 4 models are trained without hyperparameter tuning using 5-fold cross validation and 5 scoring metrics in the following code snippet:

```
from sklearn.model_selection import cross_validate
scoring = ['roc_auc', 'accuracy', 'precision', 'recall', 'f1']
dfs = []
models = [logistic, lda,qda,adaboost]
for model in models:
    cv = cross_validate(model, X, y, cv=5, scoring = scoring)
    df = pd.DataFrame(pd.DataFrame(cv).mean())
    dfs.append(df)
pd.concat(dfs, axis = 1)
```

The results are then visualized using pandas dataframe, where columns are the model names and rows are scoring metrics:

	logistic	lda	qda	adaboost
fit_time	1516.538798	2.151198	0.854514	88.629283
score_time	0.199683	0.176133	0.343900	4.204899
test_roc_auc	0.758734	0.797143	0.803511	0.824691
test_accuracy	0.703333	0.717507	0.704167	0.737726
test_precision	0.642314	0.704296	0.639817	0.713004
test_recall	0.917716	0.747601	0.934774	0.795665
test_f1	0.755694	0.724082	0.759596	0.751840

Using this results it is clear that ROC AUC score of 0.824691 is highest for the Adaboost classifier. Also it has the highest Accuracy of 0.737726 and Precision of 0.713004. However, F1 score of 0.751840 is comparable to other models' performance as well.

2) Hyperparameter tuning: Based on the abovementioned results, Adaboost was chosen for hyperparameter tuning using GridSearchCV provided by sklearn model selection package. The GridSearchCV iterates through all combinations of provided parameter values. For Adaboost number of estimators defines the number of weak learners that the model combines. The learning rate parameter specifies the weight of each of these weak learners. Finally, all of such combinations are passed through the cross validation. Cross validation splits the data into k folds, where each fold serves as a validation set and the rest used for training. This way, average score across cross validation splits returns more accurate results.

```
from sklearn.model_selection import GridSearchCV
ada=AdaBoostClassifier()
search_grid={'n_estimators':[100, 500, 800, 1200],
             'learning_rate':[1, 0.1, 0.01, 0.001]}

search=GridSearchCV(estimator=ada,
                    param_grid=search_grid,
                    scoring='roc_auc',n_jobs=1, cv = 5)
```

Finally, based on the data preprocessing, model selection and hyperparameter tuning, the Adaboost model with 800 estimators and learning rate of 1 provided ROC AUC score of 0.84418.

IV. Conclusion

Good data is half the work. The significant amount of time was devoted into cleaning the data and selecting relevant features, which added another level of difficulty since some banking terms needed to be researched. Correlation analysis and data visualization played a vital role in giving the final data. Finally, 4 classification models were used with standard parameters and evaluated based on the ROC AUC score and the one with the highest score was used for hyperparameter tuning and cross validation.

A. Limitations

The main limitation for this project was the large amount of data that was hard to read and process. This lead to a significant increase in the amount of time it took to train the model, including the Grid Search and Cross Validation. To overcome this problem, random selection of the portion of the data was done, but resulted in the loss of information. The data dimensionality reduction technique such as Principal Component Analysis could be applied, howeve the PCA produces linear combinations of the original features, which result in non-intepretability of the principal components. Further study should be done to decrease the number of features, but more importantly the number of datapoints not decreasing model performance.

Also, the amount of litrature on this dataset is limited. Although it was taken from the Kaggle platform, there was no competition held such that it would be possible to compare results with the participants.

B. Result Comparison

The ROCAUC score of 0.84418 is not the best performance on this dataset but competable with the existing literature. The folliwing results are achieved by Fares Sayah, which is available on Kaggle:

```
RANDOM FOREST                roc_auc_score: 0.724
XGBOOST                      roc_auc_score: 0.734
ANNS                         roc_auc_score: 0.906
```

and also by the Author's of the Machine Learning and Data Science Blueprints for Finance book:

```
Best: 0.955684 using {'max_depth': 5, 'n_estimators': 180}
#4 0.941023 (0.006713) with: {'max_depth': 3, 'n_estimators': 20}
#2 0.955217 (0.006234) with: {'max_depth': 3, 'n_estimators': 180}
#3 0.948795 (0.006683) with: {'max_depth': 5, 'n_estimators': 20}
#1 0.955684 (0.007695) with: {'max_depth': 5, 'n_estimators': 180}
```

REFERENCES

[1] Nasiriany, S., Thomas, G., Wang, W., Yang, A., Listgarten, J. and Sahai, A., 2019. *A Comprehensive Guide to Machine Learning.* Department of Electrical Engineering and Computer Sciences University of California, Berkeley.

[2] Tatsat, H., Puri, S. and Lookabaugh, B., 2021. *Machine learning and data science blueprints for finance.* O'Reilly Media, Inc.

[3] Sayah, F., 2021. *Lending Club Loan Defaulters* Prediction. [online] Kaggle.com.

[4] Openintro.org. 2021. *Loan data from Lending Club.* [online]

[5] Mandrekar, J., 2010. Receiver Operating Characteristic Curve in Diagnostic Test Assessment. Journal of Thoracic Oncology, 5(9), pp.1315-1316.

[6] Narkhede, S., 2021. Understanding AUC - ROC Curve. [online] Medium.