Foundation of Machine Learning – Assignment 2

Multi-Class Credit Score Classification Kaggle Challenge - Team name - ‘**DISTRICT 12’**

Irene SUNNY
Master in Data Science and Business Analytics
CentraleSupélec
irene.sunny@student- cs.fr

Shuqi DENG
Master in Data Science and Business Analytics
CentraleSupélec
shuqi.deng@student-cs.fr

1. INTRODUCTION

In an era where financial transactions are increasingly digitized and data-driven, the ability to accurately predict creditworthiness has far-reaching implications. This project leverages machine learning to develop an advanced automated model that categorize clients into distinct credit score brackets.
The dataset, encompassing key features included demographic information, banking history, credit utilization, and payment behavior, providing a holistic view of each client's financial standing. The seven machine learning methods we evaluated are **Random Forest, XGBoost, Decision Tree, KNN, Gaussian NB, Light GBM, Neural Network**. We implemented these models in python (colab) and analyzed their performance on the test dataset with 30,000 observations, shared by the Kaggle Competition.

2. DATASET DESCRIPTION

The dataset represents the credit score, response variable (Poor, Standard and Good) for customers at every month from January till August. Hence, Month and Customer_ID/SSN serves as the unique row identifier. There were 12,498 unique customers in the train dataset. Since Credit_History_Age for a specific customer remains constant across different months, it's safe to assume that these months belong to the same year.

3. EXPLORATORY DATA ANALYSIS

What facilitated smooth pre-processing was the cleanliness of the dataset, to begin with

- The dataset contained no null, missing, strange or incorrect values - confirmed through info and describe functions.
- Numerical columns with outliers were identified through box plots and Z-scores. The choice of applying RobustScaler and log transformation (though outlier removal didn’t favour) that will compress extreme values, on these columns were determined after viewing the distribution of these columns. While some (‘Monthly_Inhand_Salary’,‘Annual_Income’,‘Outstanding_Debt’ etc) are right skewed, columns like ‘Credit Utilization’ have a symmetrical distribution.
- Correlation Matrix: After being encoded to numerical values, credit_score was tested for correlation between all numerical features. Lack of strong correlations (>0.5) between credit score and any variable indicated the complex or even non-linear and multifaceted nature of credit scoring.

4. FEATURE ENGINEERING

4.1 Feature Analysis

Columns were allotted to type numerical or categorical.
‘Type_Of_Loan’ column, which contained the textual description of all the loans taken by the customer, was transformed into a numerical column. A dummy column for every loan type was created with a binary value against it.

4.2 New Features Creation

Around 10 new features were derived from classification of the data based on the distributions we observed and domain knowledge of the team mates. Example, binning the ‘Age’ column into different categories allowed for a more structured analysis of credit behavior across different life stages, recognizing that financial stability and needs often vary with age. 'Credit_History_Length' was created to dig out insights into monthly spending patterns and the length of the customer's credit history.

Polynomial features were introduced to capture nonlinear relationships but they failed to have a positive contribution. We iteratively tested these new features against model performance, i.e ‘Num_Credit_Product’, ‘Num_Transanctions_Per_Month’ and ‘Credit_History_Age’ notably improved the accuracy score while ‘Ration_Delayed_Payments’ and ‘Avg_Spending_Per_Transaction did not.

The development of new features significantly enhanced the model's accuracy from Stage 1, likely because these additions captured valuable information not evident in the original features alone. At this phase, our model incorporated a total of 42 features.

4.3 Scaling and Encoding

RobustScaler was applied to the numerical columns, given it is designed to be robust to outliers in the data. Unlike scaling methods, such as StandardScaler, which uses the mean and standard deviation, RobustScaler uses the median and the interquartile range (IQR), making it less sensitive to extreme values.

While at the beginning, all categorical columns were one hot encoded, we iteratively tested how well each encoding method affected the model and decided to do ordinal encoding for the following:'Credit_Mix','Payment_of_Min_Amount','Payment_Behaviour', 'Occupation', 'Age_Category and ’Month’. This decision was

underpinned by the inherent order and hierarchical nature of these variables.
For instance, 'Credit_Mix' and 'Payment_of_Min_Amount' exhibited a clear gradation in terms of creditworthiness and payment reliability, respectively. Similarly, 'Occupation' and 'Age_Category' were considered to have a latent order correlating with financial stability, i.e a customer earning high income for several years will have better ability to manage credit. At this stage, we had a total of 86 features.

4.4 Feature Selection

In the initial stage of our project, a comprehensive literature review and the inherent advantages of ensemble methods and gradient boosting framework led us to hypothesize that RF and XGB models would be particularly effective for our multi-class credit score classification problem. Using these two models, we extracted the top 20 features through feature importance score, which reflected the average reduction of the classifier's error when a feature is used.

We had these sets of features tested and treated for multicollinearity through the VIF Test. All the models performed better when all features were given as regressors.

Random Forest was selected as best performer and three feature selection methods (SelectFromModel, Top-N features, RFE) were tested on it. We chose SelectFromModel for feature selection, leveraging its efficiency and ability to improve model performance through feature importance. Through testing, we determined that the optimal threshold for SelectFromModel was the 'Median'.

Since Random Forest Model handles high dimensionality and extract non-linear relationships effectively, dimensionality reduction methods like PCA did not improve performance.

Irrelevant columns - Customer_ID, Name and SSN were dropped from the dataset for modeling.

5. MODEL BUILDING

5.1 Class Imbalance Treatment and Dataset Split

Training dataset exhibited a class imbalance with a predominance of the 'Standard' class.

The original train data was split into two parts and the part which was used for training is referred to as the training set. The other part that was set aside for evaluation of the models is referred to as the test set. The Training and Test split choice takes into consideration the amount of data we had.

We approached this process in three different ways- 1) Used train_test _split function and then applied RandomOverSampler to only the training set to address class imbalance by oversampling the minority class. 2) StratifiedShuffleSplit that splits the data in a way that maintains the same proportion of classes in both the training and test sets as in the original dataset. it ensures that both splits represent the class distribution of the original dataset. 3) SMOTE that generates new synthetic samples that are similar to the existing minority class instances by randomly picking a point between a minority class sample and its neighbors. This can lead to a more robust and fairer model, as the minority class is better represented. 1) and 3) were performed after the data is split into training and validation and 2) was itself a splitting function.
RandomOversampler gave the best results.

5.2 Model selection and tuning

To begin with, we tried 5 different machine learning algorithms, namely - RF, XGB, Decision Tree, KNN, LightGBM to make predictions about our multi-class classification problem, with just the default parameters and using features selected from RF and XGB, separately. RF (picked and tuned further) performed the best followed by XGB.

5.3 Hyper Parameter Tuning and Cross Validation

In optimizing our Random Forest model, we employed a two-phase hyperparameter tuning approach. Initially, we used Grid Search CV, identifying a set of promising parameters. Building on this initial insight, we conducted a second, narrowed iteration of Grid Search CV, targeting specific ranges and values such as.

This two-step, narrowing-down strategy allowed us to effectively hone in on the best combination of parameters, balancing model complexity and performance.

Having narrowed down the hyperparameter space with Grid Search, we used Bayesian optimization to further refine the hyperparameters. Unlike Grid Search that exhaustively tests all possible combinations within the specified grid, Bayesian optimization focuses on areas of the parameter space most likely to yield improvements. This took our accuracy from 0.813 to 0.819.

Cross-validation is inherently part of Grid Search CV and Bayesian optimization. The dataset was split into '3' folds, and for each combination of parameters in the grid, the model was trained and validated '3' times. In the second iteration, we increased the folds to ‘5’ to train and validate model on more diverse combinations of data.

5.4 Evaluation

1. Accuracy : The metric which was indicated to be the evaluation criteria for this kaggle competition, quantifies how often the model correctly classifies a customer's credit score. RF model topped in terms of accuracy at the first stage. XGB was the second best.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

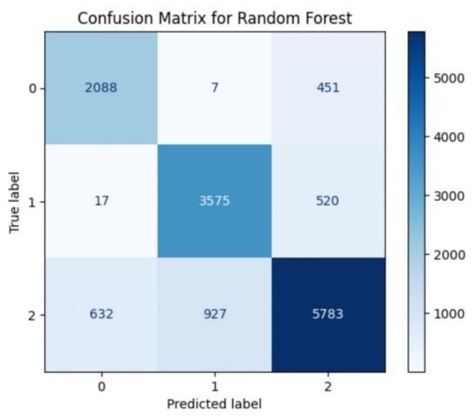
2. Confusion Matrix: It provides visualization of performance of a classifier algorithm. More importantly it displays label noise, that is, examples from one class that are labeled as belonging to another class. This matrix provides a quick diagnostic of the performance of a classifier.

3. Additionally, we plotted the Learning Curve (Refer Appendix 10.2) to understand if the mode is fitting the data well and at various stages, the accuracy score on train data varied significantly compared to test data on kaggle. We applied early stopping to prevent overfitting and to halt training when the model's performance on the validation set stops improving, and to find the best number of trees.

6. OVERALL EXECUTION AND RESULT

The execution of this project involved trying different feature combinations, testing multiple models and applying various preprocessing techniques. Each attempt provided insights that guided our subsequent strategies and decisions. Hence, this project's narrative is enriched by an iterative journey. At the appendix, find the summary of the modeling attempts covering major milestones - starting with trying 5 models with basic parameters, selecting RF as the best performing model, trying altearnative models, improving RF performance through feature engineering and measuring process and finally tuning the hyper parameters through different methods available.

Below is the confusion matrix of the final, best performing RF model. The final model ended with an accuracy score of 0.814857 on the validation set and 0.8227 on the test set.



```
              precision    recall  f1-score   support

           0       0.76      0.82      0.79      2546
           1       0.79      0.87      0.83      4112
           2       0.86      0.79      0.82      7342

    accuracy                           0.82     14000
   macro avg       0.80      0.83      0.81     14000
weighted avg       0.82      0.82      0.82     14000
```

7. ALTERNATIVE MODELS TESTED

The two other models we employed and enhanced were

7.1 Neural Networks

The choice is due to the fact that it is capable of learning intricate, non-linear relationships within the data. Unlike traditional machine learning models that require manual feature engineering, neural networks can automatically learn and derive features during the training process. This ability to perform feature extraction and selection implicitly can be particularly advantageous in high-dimensional spaces. However, despite the effort in feature engineering, preprocessing and hyperparameter tuning, the Neural Network model did not achieve the desired accuracy given a much lower baseline score compared to Random Forest.

7.2 Ensemble Learning

Stacking - We created a stacking classifier with Random Forest and Gradient Boosting as base models and then fitted the stacking model on the training data, ending in a lower score(0.7833)
Boosting - We tried AdaBoost Model and XGBoost Model, both ended with a lower score, 0.80714 and 0.79793.

8. DISCUSSION

Another attempt we could not pursue was trying to capture the temporal trend in the dataset. At a Customer_ID and Month level, we created new feature called 'Rolling_Avg_Score. However, it cannot be applied to the test data, since we don't have any credit score in test data. Secondly, whether to keep the identifiers as features could not be decided. Customer_ID, would be irrelevant and categorical columns will be too many. We created a column 'Rolling_Avg_Total_EMI' to capture temporal trend which was a proxy for Credit_Score, which improved the performance on validation set to beyond 0.82, however only 0.8171 on kaggle. We believe, Deep learning models, particularly recurrent neural networks (RNNs) or long short-term memory networks (LSTMs) could excel in capturing intricate temporal dependencies and patterns within sequential data.

9. REFERENCES

- Ethem Alpaydin. Introduction to Machine Learning (Adaptive Computation and Machine Learning). The MIT Press, 2004
- R.L. Liu. Machine learning approaches to predict default of credit card clients. Modern Economy, 9:1828–1838, 2018. URL: https://doi.org/10 .4236/me.2018.911115.
- Rashmi Malhotra and D. K. Malhotra. Evaluating consumer loans using neural networks. Omega, 31(2):83–96, 2003.
- Lyn Thomas, David Edelman, and Jonathan Crook. Credit Scoring and its Applications. 01 2002.
- Lien C. H. Yeh, I. C. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. Expert Systems with Applications, 36(2):2473–2480, 2009.

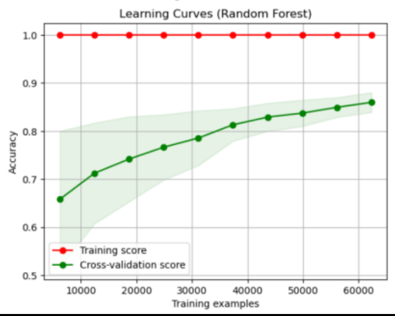
10. APPENDIX

10.1 Summary of the modelling milestones

Stage 1 – Testing 6 Models	Stage 2 – Enhancing RF Model	Alternative Models Tested

Algorithm	Description	Parameters	Test Accuracy
Random Forest	Default parameters, Top features selected from RF	random_state=42	0.80
XGBooster	Default parameters, Top features selected from XGB	random_state=42	0.77
Decision Tree	Default parameters, Top features selected from RF	random_state=42	0.73
Light GBM	Default parameters, Top features selected from RF	random_state=42	0.75
KNN	Default parameters, Top features selected from RF	-	0.74
Gaussian NB	Default parameters, Top features selected from RF	-	0.66
Random Forest	Default parameters, revised feature engineering, newly derived features used	n_estimators=100, random_state=42	0.808
Random Forest	Switching from hot encoding to ordinal for few columns, Best parameters from Grid Search CV	'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 300	0.814
Random Forest	Best parameters from Bayesian Search after Grid Search CV	OrderedDict([('max_depth', 30), ('min_samples_leaf', 1), ('min_samples_split', 2), ('n_estimators', 300)])	0.822
Neural Network	Default parameters, all features except irrelevant columns	optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'],epochs=50, batch_size=32, validation_split=0.2, callbacks=[early_stopping]	0.734
Ensemble Stacking	Default parameters, all features except irrelevant columns	rf_model = RandomForestClassifier(n_estimators=100, random_state=42), gb_model = GradientBoostingClassifier(n_estimators=100, random_state=42)	0.783
Ensemble AdaBoost	Default parameters, all features except irrelevant columns	rf_model = RandomForestClassifier(n_estimators=100, random_state=42), adaboost_model = AdaBoostClassifier(base_estimator=rf_model, n_estimators=50, random_state=42)	0.807
Ensemble XGBoost	Parameters based on Bayesian Search, features selected by SelectFromModel	best_xgb_model = XGBClassifier(n_estimators=300, max_depth=30,learning_rate=0.1, subsample=1.0, colsample_bytree=1.0,random_state=42)	0.797

10.2 Learning Curve


Learning Curves (Random Forest)
Accuracy
Training examples
Training score
Cross-validation score