Project Title: Credit cards approval classification by Tuo Sun, Di Jin, Jiayi Lin

Problem Definition

Seventy-four years has passed since the first Bank-issued charge card was issued in 1946. Nowaday, credit cards have become one of the major payment instruments for people all over the world. About one billion credit cards [1] are issued each year. Among one billion cards, there are lots of holders with the history of past due, overdue, or even bad debts and write-offs. If banks choose to give these customers high credit lines, potential huge money loss will incur toward banks. In order to avoid this situation, our team will use Python as the programming language and multiple machine learning and deep learning models including Decision Tree, Boosting, Random Forest, Support Vector Machines, and Neural Networks to answer the following questions:

1) What are the definitions of good customers and bad customers that will bring huge potential loss?
2) Which models can give us the high accuracy predicting if a customer is a good customer or not to the bank?
3) Which model and the specific factors we should use to predict the customer behavior in a business environment？

To answer this question above, our team will use all the data from customers' credit card application profiles to train models and test models for better accuracy. All the raw data is based on "Credit Card Approval Prediction" [2] was published by *Song, Xiao* from kaggle.com.

Description of Background

In our project, we will use Decision Tree, Boosting, Random Forest, Support Vector Machines, and Neural Networks to build classification models that can help the company decide whether to accept credit card applications from applicants or not. And these classification models will provide great value to the company by cutting down the workloads in the Risk Management Department. According to *Kolakowski* [3], the median annual salary of a risk management employee is $127,990 or $61.53 per hour. Furthermore, a risk management department for a big bank might have hundreds or even thousands of employees. Let's assume that each employee uses 20% of their working time to approve credit applications. That means each bank pays $25,598 on average to its risk management employees. If this task can be finished by the classification models, banks could save millions of dollars each year.

Description of Dataset

Our dataset includes two CSV files. The first csv file "application_record.csv" has 438557 credit card applicant IDs. The second csv file "credit_record.csv" has 1048576 transaction records from 45986 customer IDs.

"application_records.csv" has the following columns:
ID: Unique Id of the row
CODE_GENDER: Gender of the applicant. M is male and F is female.
FLAG_OWN_CAR: Is an applicant with a car. Y is Yes and N is NO.
FLAG_OWN_REALTY: Is an applicant with realty. Y is Yes and N is No.
CNT_CHILDREN: Count of children.
AMT_INCOME_TOTAL: the amount of the income.
NAME_INCOME_TYPE: The type of income (5 types in total).
NAME_EDUCATION_TYPE: The type of education (5 types in total).
NAME_FAMILY_STATUS: The type of family status (6 types in total).
DAYS_BIRTH: The number of the days from birth (Negative values).
DAYS_EMPLOYED: The number of the days from employed (Negative values). This column has error values.
FLAG_MOBIL: Is an applicant with a mobile. 1 is True and 0 is False.
FLAG_WORK_PHONE: Is an applicant with a work phone. 1 is True and 0 is False.
FLAG_PHONE: Is an applicant with a phone. 1 is True and 0 is False.

FLAG_EMAIL: Is an applicant with an email. 1 is True and 0 is False.
OCCUPATION_TYPE: The type of occupation (19 types in total). This column has missing values.
CNT_FAM_MEMBERS: The count of family members.
“credit_records.csv” has the following columns:
ID: Unique Id of the row
MONTHS_BALANCE: This customer did not pay his or her credit card for how many months. 0 is the current month, -1 is the previous month, and so on.
STATUS: The status of credit. 0: 1-29 days past due 1: 30-59 days past due 2: 60-89 days overdue 3: 90-119 days overdue 4: 120-149 days overdue 5: Overdue or bad debts, write-offs for more than 150 days C: paid off that month X: No loan for the month.

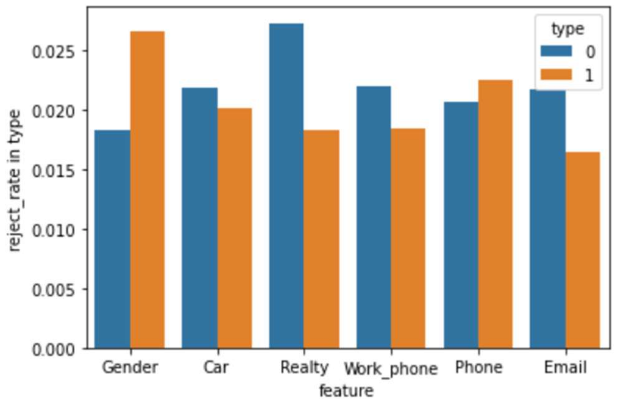
At the first glance, there were some issues in our data which could greatly affect the quality of the final results. First, in the “application_record.csv”, we found that there are some applications submitted by the same person since they have exactly the same profile. Second, “X” and “C” in the STATUS column of “credit_records.csv” have the same meaning. Finally, according to our early manual observation, our datasets have a very imbalanced data distribution which is another issue that needs to be solved. Beyond these three issues, our dataset looks complete and clean. We will illustrate how we solve these three issues by cleaning data and choosing training and testing sets in the next section.

Description of methods used

Since we directly download the dataset from the kaggle. Our data collection process only includes manually downloading. After that, we manually checked the data sets and solved the three issues we mentioned in the last section. We cleaned the dataset by 1) dropping duplicates for IDs with the same features 2) replacing the status “X”, “C” and changing the STATUS column type to integer for further usage. We also found that if status is greater or equal to 2 that means this customer has a bad credit history according to the definition. Thus, the new

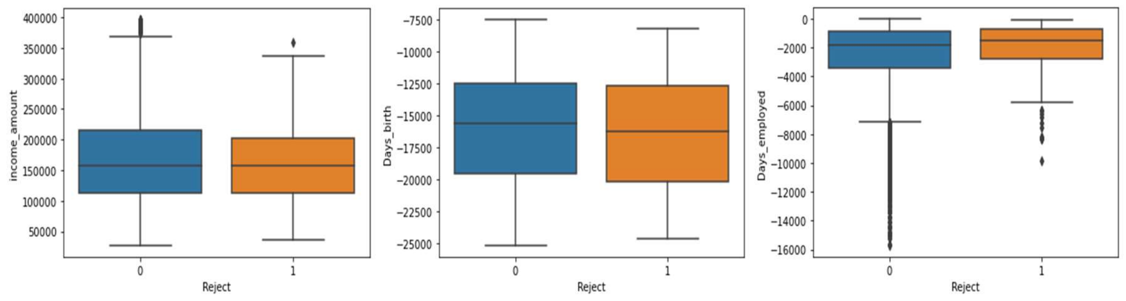
STATUS column only has 2 values available: 0 for good credit history, 1 for bad credit history. After the data cleaning process, we found that there are only 206 rows with STATUS=0 in 9706 rows for all data. For imbalanced classification problems like this, we are going to use the SMOTE function for balancing the data. As most of the articles have selected, we chose 70% vs 30% for the training and testing set respectively. Before modeling the data, we will firstly finish an exploratory data analysis. In order to have a better understanding of features in the dataset, we will explore the data separately. It will be divided into three parts: binary features, digital features, and class features exploratory data analysis.

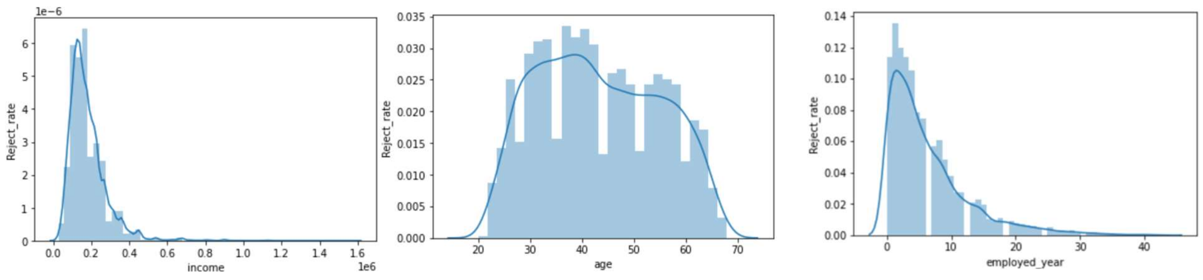
For the binary features, we have 7 binary features in the dataset which are *Gender, Car, Realty, Mobile, Work_phone, Phone* and *Email*. We dropped the column *Mobile* since every applicant has a mobile so that this feature will not bring additional information gain. Then we used these features to generate the features' rejection rate plot (shown below). In this plot, type 0 and type 1 means different features. *Gender*'s type 0 means female applicants, and *Gende*r's type 1 means male applicants. For other attributes, type 1 means the applicants have such assets or tools, and type 0 means the applicants do not have such assets or tools. As you can see from the plot, *Gender* and *Realty* have huge gaps (difference > 0.8% in rejection rate) between type 0 and type 1. On the other hand, *Phone* and *Car* (difference < 0.2% in rejection rate) do not have huge gaps.



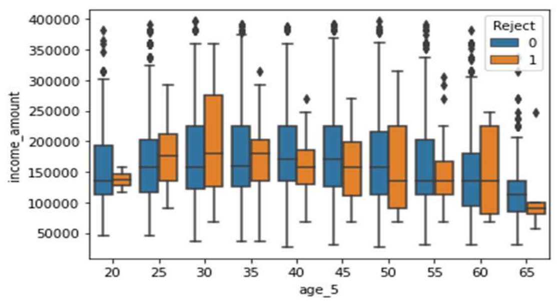
For the digital features, there are 5 digital features in the dataset *Children_count*, *ount_family_members,* *Income_amount*, *Days_birth*, and

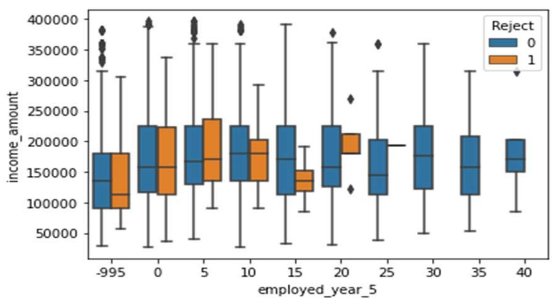
Days_employed. For the attribute *Children_count*, we found that applicants with 0-2 children have almost the same rejection rate (range: 2.02% - 2.34%). Applicants with 3 or more children have a higher rejection rate (3.95%), but it could due to small data size (only 152 instances). Thus, we do not think *Children_count* can be a useful factor for our classification models. Rejection rate of applicants with 1, 2 or 3 family members are similar. More children usually mean more family members, so the conclusion seems similar. Then, we examined the relationship between rejection rate and Income_amount, Days_birth, Days_employed generated boxplots and histograms (Box plots :1 means rejection, 0 means approval. Histograms: y-axis is the rejection rate) below.





As you can see from histograms, when applicants with low income and employed years, they will more likely be rejected compared with others. We can see the rejection rate is pretty high when the applicants only earn less than 40,000. And we can also see a clear path of rejection rate failing when years of employment increase. For the age column, we can see a clear boundary between high rejection rate and low rejection rate which is age 40-45. However, years of employment and income amount have higher correlation with rejection rate. To demonstrate our points, we also generated the boxplot combining income amount, age, rejection rate and income amoun, employed year, rejection rate (boxplots shown below). After examining these two boxplots, we were convinced that *Income_amount* and *Days_employed* will be great factors for constructing classification models.



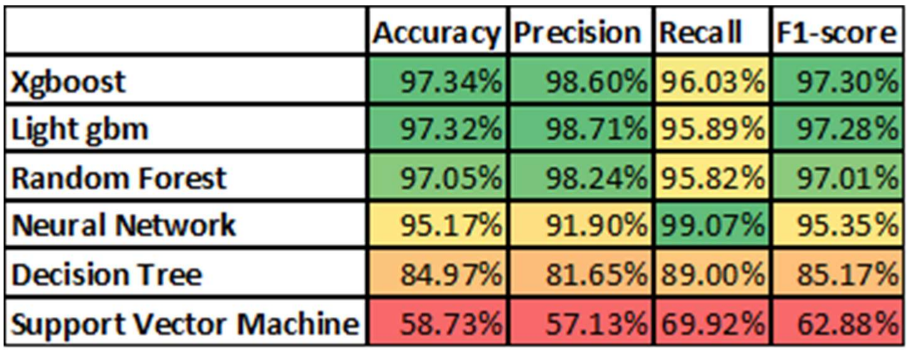


Third, we have class features including *Income_type*, *Education_type*, *Family_status*, *Housing_type*, *Occupation_type*. After we analyzed these features we found the following anomalies: **1)** if an applicant's primary income source is pension fund, its rejection rate is 3.91% which is way higher than other groups (*Income_type* rejection rate range except pensioner: 1.39%-1.83%). **2)** if an applicant only has a lower secondary certificate, he or she will have a 4.39% rejection rate (*Education_type* rejection rate range except lower secondary: 1.89%-2.16%). **3)** if an applicant's current marriage status is not married or single, he or she will have 3.75% rejection rate (*Family_status* rejection rate range except single or not married: 1.44%-2.93%). **4)** if an applicant is living in a municipal apartment he or she will have a 4.33% rejection rate (*Housing_type* rejection rate range except municipal apartment: 1.32%-2.08%). **5)** the occupation type of applicants also has a great influence on the rejection rate. We found that secretaries, realty agents, HR staffs and IT staffs have 0 rejection rates. And medical staffs only have a 0.69% rejection rate. On the other hand, low-skill laborers have the highest and abnormal rejection rate which is 5.66%. From all the observations from class features, we will use the five abnormalities mentioned above to construct our models.

Finally, we started to use Python construct models we listed above. To achieve this, we import packages including pandas, numpy, Scikit-learn, imblearn, keras, xgboost, lightgbm, matplotlib, and seaborn. We used pandas and numpy to process the data. Using Scikit-learn for building Decision Tree, Random Forest models. Using keras for the Neural Networks model, xgboost and lightgbm for the Boost model. And finally, we use matplotlib and seaborn to generate our final visualizations.

Experiment: analysis results

After we ran all the models mentioned above we generated the following results:



	Accuracy	Precision	Recall	F1-score
Xgboost	97.34%	98.60%	96.03%	97.30%
Light gbm	97.32%	98.71%	95.89%	97.28%
Random Forest	97.05%	98.24%	95.82%	97.01%
Neural Network	95.17%	91.90%	99.07%	95.35%
Decision Tree	84.97%	81.65%	89.00%	85.17%
Support Vector Machine	58.73%	57.13%	69.92%	62.88%

As you can see from the table, Xgboost has the highest Accuracy and F1-score among all the models. Light gbm has the highest Precision value.
Neural Network has the highest Recall value even its Accuracy, Precision, F1-score values did not seem pretty. On the other hand, the Decision Tree model and Support Vector Machine model did not give us acceptable results.

Observation and Conclusion

After careful consideration, our team decided to use Light gbm as the model for applicants classfication. There are three reasons for that. First, our models need to have high accuracy. Thus, only Light bgm and Xgboost can qualify for this (they have almost the same accuracy both > 97.3%). Second, Light gbm has a great F1-score of 97.28% which implies its great quality. Last but not least, Light gbm is 6 times faster than Xgboost. Therefore, even Xgboost has tiny advantage over Light gbm on accuracy, recall, and F1-score, the speed of Light gbm can compensate for this tiny performance. Since we are living in a real world not laboratory, the speed of the model means the money and time we can save for the companies.

Beyond our conclusion, we have other findings. For decision tree algorithm, the influence of each variable on credit is calculated by the information value. By calculating information value, we are able to understand which variables have great impact and which variables have little impact. And we found the top 3 influential are family status / age group / realty. For other tree-based models like Random forest, xgboost and lightgbm. Random forest is a collection of decision trees. With randomly created decision trees with randomly selected features subsets, we found Random forest can reduce overfitting, especially in a dataset with low noise. All the other models that we used like Xgboost and lightgbm as members of the Boosting family have good performances in this dataset. For the deep learning method, after one-hot encoding and scaling features, we are able to apply a dataset to a Back Propagation Neural Network. After a 300 epochs train for 128 batch size. We are able to get a high accuracy for this problem as well. However, Support Vector Machines as a binary linear classifier have a poor performance in this problem. We found this may due to features in two STATUS does not have a clear boundary.

References

[1] Peter, Bianca. “Number of Credit Cards and Credit Card Holders.” WalletHub, 15 July 2020, wallethub.com/edu/cc/number-of-credit-cards/25532.
[2] Song, Xiao. “Credit Card Approval Prediction.” Kaggle, 24 Mar. 2020, www.kaggle.com/rikdifos/credit-card-approval-prediction.
[3] Kolakowski, Mark. “Risk Management Employee Job Description: Salary, Skills, & More.” The Balance Careers, www.thebalancecareers.com/risk-management-career-profile-1286898.
[4] Tuo, Sun. “Credit Card Approval Prediction: EDA” Kaggle, 27 Nov. 2020, https://www.kaggle.com/tuosun493/credit-card-approval-prediction-eda