**Credit Card and Loan Approval Machine Learning Writeup**

Sajid Anjum

Raymond Bell

Han Le

April Gao

Contents:

1. Introduction and Motivation (Proposal) (Ray)
2. Research Questions (Later)
3. Data Cleaning/Wrangling/Processing/Filtering/Machine Learning (Sajid)
4. Machine Learning (Sajid)
5. Tableau (Ray, Han, and April) (Screenshots)
6. Web App:

a. Home Page (Sajid)

b. Forms (April)

c. Tableau (Sajid)

d. Data table (Han)

e. About (Sajid)

f. References (Sajid)

V. Conclusions and Future Work (Later)

**I. Introduction:**

In the twentieth century, banking underwent significant changes that made financial services more accessible to more people. Advancements in technology and government policies played key roles in this transformation. Banks began to offer new products and services, such as personal loans and credit cards, that allowed individuals to access credit and manage their finances more easily. The development of automated teller machines (ATMs) made banking more convenient, as people no longer had to wait in long lines to make deposits or withdrawals. Additionally, the creation of online banking platforms and mobile apps further increased accessibility, allowing customers to check their account balances, transfer funds, and pay bills from the comfort of their homes. These advancements helped to democratize banking, making it more accessible to people across different socioeconomic backgrounds.

However, no matter how accessible a bank makes its services, if the banker does not make sound decisions about which services to offer which customers, it will not be able to maintain solvency. Our goal is to develop a machine learning web app that helps a banker make decisions about whether to approve or deny credit card services or loan services. In addition, we would also like to use the predictions of our machine learning algorithm to give customers some more insight about how the bank makes its decisions, and what the bank looks for when it approves or denies customers access to services in a fair and impartial manner.

**II. Research Questions:**

1. Which machine learning algorithm is best suited to correctly predicting whether credit cards are approved based on our data set? What about for Loans?
2. How did we optimize the machine learning algorithm?
3. What were the most important features of our machine learning algorithm, and how do those features affect the probability of credit card approval?
4. Some questions from the dataset:
5. What is the relationship between home ownership and credit-card/loan approval?
6. What is the relationship between total income and credit card approval?
7. How does income vary with marital status?
8. What are the most common family-sizes for credit card approval?
9. What are the most common purposes for loan approval?
10. Do our model’s predictions match up with trends that we observe in our raw data?

**III. Data Cleaning and Processing and Machine Learning in Jupyter Notebook:**

1. **Credit-Card Data Set:**

We worked with two different datasets for this project as approval for credit cards and loans require different criteria. The first dataset was sourced from Samuel Cortinhas on Kaggle, and it is about credit card data in China. It consists of 9709 rows of individuals and 20 columns. The dataset was already very clean with no null values. We decided that to drop the [“Work\_phone”, “Phone”, and “Email”] columns as even though those columns may be predictive in China, we do not think they will have much predictive power in America. We acknowledge that using a Chinese dataset is going to have sub-optimal predictive power for an American clientele, but we feel that for the scope of this project, it is still instructive to carry out the analysis.

The final columns that we chose were:

['ID', 'Gender', 'Own\_car', 'Own\_property', 'Unemployed', 'Num\_children',

'Num\_family', 'Account\_length', 'Total\_income', 'Age', 'Years\_employed',

'Income\_type', 'Education\_type', 'Family\_status', 'Housing\_type',

'Occupation\_type', 'Target']

The explanations for all those columns can be found at the Kaggle page for that data:

<https://www.kaggle.com/datasets/samuelcortinhas/credit-card-classification-clean-data>

In order to process the data for our machine-learning model, we dropped the ‘ID’ column, and label-encoded the [‘Own Car’, ‘Own Property’, ‘Unemployed’] columns as only two options were available for each. We employed a standard scaler for the ['Num\_children','Num\_family', 'Account\_length', 'Total\_income', 'Age', 'Years\_employed']columns, but it didn’t make much difference so we didn’t use it in our model. We one-hot encoded the ['Income\_type', 'Education\_type', 'Family\_status', 'Housing\_type', 'Occupation\_type']columns. The “Target” column was our prediction column, and we used that to train our classification models.

Table

Description automatically generated with medium confidenceA picture containing table

Description automatically generatedWe also modified the ‘Num\_children’ and ‘Num\_family’ columns to collapse amounts of 3 and 4 respectively into the numbers 3 and 4. This is because there wasn’t much data for family sizes larger than those numbers and we didn’t want to skew our model. We understand that this is a limitation because clearly a family size of 10 is not the same as a family size of 5. However, there is no point in differentiating between the two sizes if our model does not have the data to do so.

Text

Description automatically generatedText

Description automatically generatedOur final dataset had 9709 rows and 40 columns. However, our dataset was imbalanced, so we used the SMOTE library to amend our data. The SMOTE library oversamples the imbalanced columns so that there are an equal number of Approved and Denied columns for the model to be trained on. On the right, we can see the value counts of our Target before and after using SMOTE on the dataset.

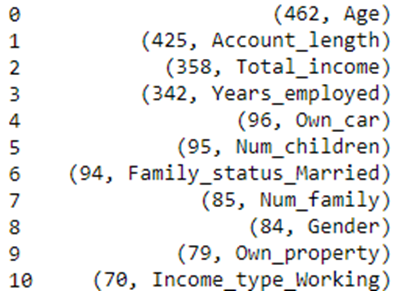
After using SMOTE, we did a 0.75:0.25 train-test split on our data and used our dataset to train several different classifier machine learning models such as logistic, KMeans, and Tree-based algorithms. We also used a keras based neural network. The models that worked the best were two tree-based algorithms--RandomForestClassifer() and LGBMClassifier(). Both models has comparable ROC curves and confusion matrices, but the LGBM classifier showed less evidence for overfitting. Not only did the neural network model not perform as well, the tree-based algorithms did so well that we figured the lack of explainability and extra processing power that neural networks demand were not Chart, line chart

Description automatically generatednecessary to solve our problem. Below are the results of our LGBMClassifer Model:

Table

Description automatically generated

Our two most important numbers were “precision” and “recall” for the target of 1. A precision of 1 means that the model was able to identify every single credit-card default as a credit-card default, which is extremely useful for any bank. A recall of 0.83 means that the model declined 17% of customers who would probably have not defaulted on their loans. This means that it loses the legitimate business of one customer every time it rejects five customers, which is not bad, but also not ideal. Perhaps a higher recall at the expense of a lower precision may boost the profit margin slightly. However, there is no doubt that if our dataset rejects reality, this model is very useful.

Lastly, we wanted to see which features of our model had the most predictive power. The top ten feature importances on our LGBMClassifer model are show on the right:

It is no surprise that age, income, employment, and family status are important predictors. What is surprising is that the second most important predictor is the length of time that the person had another credit card.

Chart, line chart

Description automatically generatedIn order to determine how our model’s predictions vary with these features, we created a “most common customer” such that the values for all the numerical columns of this customer were the means of the dataset, and the values for all the categorical columns were modes. Then we varied just the age and saw how the probability varied, and did the same with a few other features. The graphs below are what we saw:

A picture containing line chart

Description automatically generatedChart, line chart

Description automatically generated

We see that you are less likely to be rejected the older you are. However, the probability of being rejected increases very quickly with the number of months a person has already owned a credit card up to about ten months, after which it flattens out. It seems that banks prefer to give new customers a chance at having a credit card. Lastly, the probability of being rejected falls precipitously with higher income, which is also to be expected.

Shape, line chart, polygon

Description automatically generatedA picture containing logo

Description automatically generatedLogo

Description automatically generated

The years employed category varies very strangely with approval rating. The probability of being rejected rises dramatically the more a person works, then falls steeply, then sort of plateaus. We feel this maybe a reflection of the spending habits and application habits of people and how they vary versus how long they have worked. This is an interesting feature that would be difficult to notice without machine learning. The model also identifies males as a riskier investment than females and property and car owners as less risky investments. Lastly, the model does not consider a family with more children to be a riskier investment but does consider a person with no children to be a more risky investment.

A blue and white rectangle

Description automatically generated with low confidenceLogo

Description automatically generated

1. **Loan Data Set**

The second dataset that we worked with was from Arbaz Khan, also on Kaggle, and can be found at this link:

<https://www.kaggle.com/datasets/arbazkhan971/loan-approval-analysis>

**Table

Description automatically generated**This dataset connected the status of a loan (Charged Off, Current, or Fully Paid) to several other facts about the people who took the loan out. We felt that we could use this data to build a machine learning model to predict whether a bank should approve or deny the loan.

This was a large dataset (39,716 rows and 111 columns) and was very unclean. We decided to use this for two reasons—to see whether we could use the large amount of information to make a better model and to take on the challenge of cleaning the data. First, we dropped all columns that had at least 30000 non-null values, which reduced our number of columns to 53. There also weren’t very many rows with null values, so we dropped all rows that had null values and ended up with 36,431 rows.

Our goal was to use the ‘loan\_status’ column as the Target for our machine learning model. However, a status of **Current** was irrelevant for our model, and only 1066 rows had the **Current** status, so we dropped those rows. This gave us a final row count of 35365 rows. Our next step was to whittle our columns down to a more manageable number and process the data. We settled on a final amount of 22 columns. They are listed to the right.

The remainder of the cleaning was mostly converting the datatypes from string to integer or float, and from datetime to integer. We made sure that there weren’t too many minor categories in the categorical columns, and finally, we converted the states (48 different categories) into Federal Reserve regions (only 12 different categories) to make our dataset more machine-learning friendly.

We were left with only four categorical columns: ['home\_ownership', 'verification\_status', 'purpose', 'fed\_reserve\_region']

Text

Description automatically generatedWe used pd.get\_dummies to one-hot encode those columns, dropped our ID column, and our dataset was ready for machine learning.

However, our dataset was imbalanced. As with the credit card data, we used SMOTE to balance our dataset, did a 0.75:0.25 train-test split, and fed the dataset into various machine learning algorithms and neural networks. The best performing model was a RandomForestClassifier. The model was perfect!

**Chart, line chart

Description automatically generated**Table

Description automatically generated

Text, letter

Description automatically generatedWe think it is very suspicious that the model is so perfect, but this is what we got. Our list of the most important features is given on the right. The most important features were the length of the term, the interest rate, the dates of the last credit pull, the grade of the loan, and the income of the applicant. Unfortunately, we did not have time to do a detailed analysis on how the model varies with these features.

**IV. Tableau**

Credit Card

First dashboard: showing the statistics of the dataset population including the family status, education type, average of total income and average years of employment. We have a couple of filters where we can sort by the approval status, average total income, family status and education type.

+The bar chart with family status, education type and average years of employment: Most of our candidates have a tenure of 2-3 years with their employers and the applicants with higher education seem to have higher tenure than the candidates in other education groups. The highest tenure of the population is around 7.5 years

+The bar chart with family status, education type and average total income: It looks like the candidate with academic degree make the highest incomes among the group and double the income of the candidate to didn’t complete their education.



Second dashboard: showing the relationship between the approval status and family status, average total Income, property ownership, and family size. We have a couple of filters on the right where we sort by.

Application Status: we can choose to see Approved only, or Denied only, or both together

Average Income: we can show all the values or narrow down the range

Family Size: we can choose the exact family size or a range of family sizes to review all the sizes on the same visualization

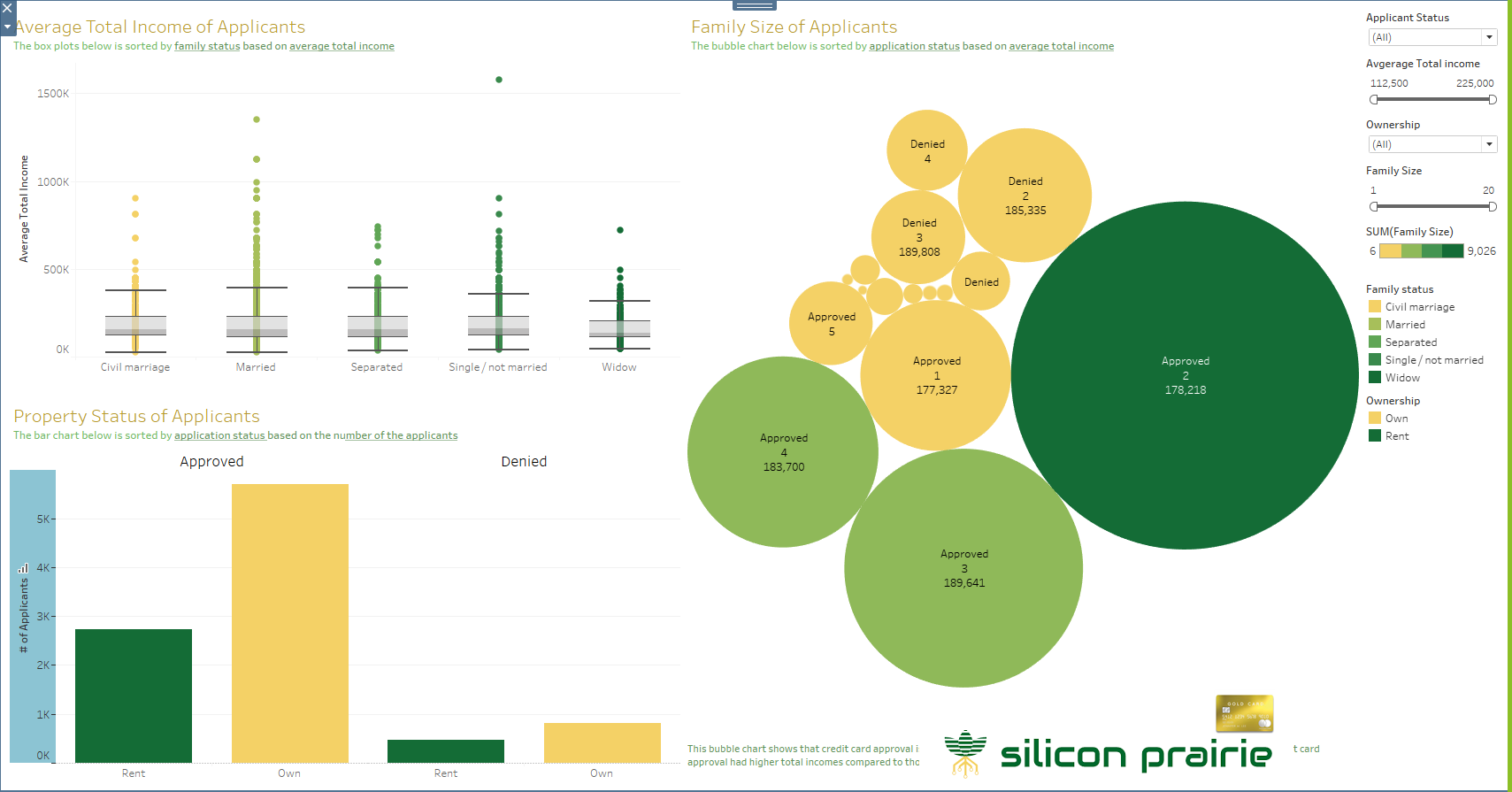
Family Status: we can select all or only candidates that are married, single, separated, or widow

Ownership: we can choose both or either whether the candidates own properties or not.

+Box plots: the box plots are sorted by the family status based on average income. The family status broke down into civil marriage, married, separated, single/not married or widow. Overall, the median income of the first 4 groups is quite close to each other, and relatively higher than the last group, widow.

+Bar chart: The bar chart is sorted by application status, property status, and based on the number of applicants. It has been consistent throughout the analysis that more candidates are receiving approvals than denials. From the chart, more house owners received approvals than non-house owners (renters), approximately double the number.

+Bubble chart: The bubble chart is sorted by the application status (approved vs. denied) based on the applicant's average Income and family size. The bubble size reflects the number of candidates. The bigger the bubbles are, the more applicants are approved or denied. Using the family size of 2,3, and 4 (most giant bubbles), more candidates receive approvals than denials for their credit applications. For those with the same family size, those receiving approvals are making less than those who received denials. A few key points concluded from the bubbles are that Income is not the sole determinant driving the decision, as even though you can make good money, there is a chance you get denied. There must be other criteria factoring into the decision on credit card applications. Also, the family size seems irrelevant to the decision.



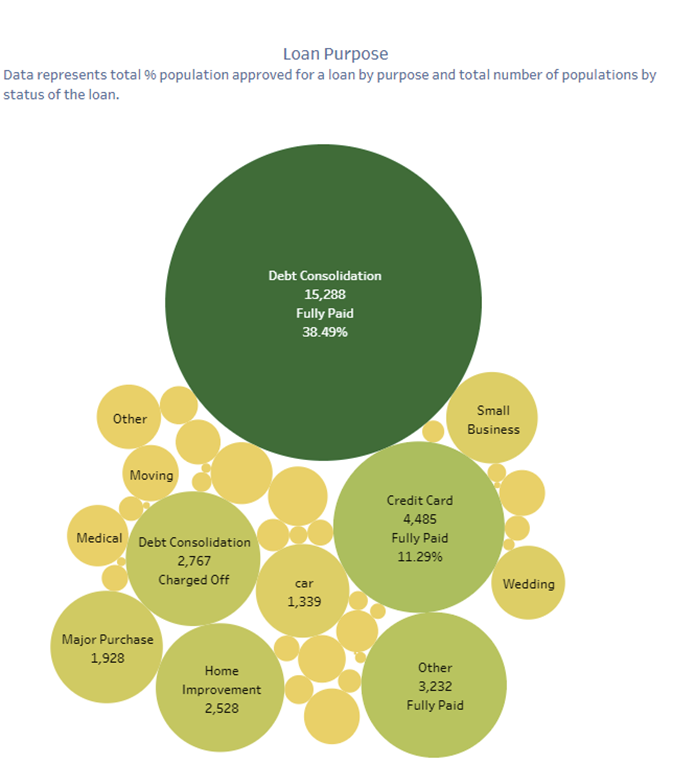
Loan

There are three Tableau Dashboards for Loan application dataset.

Dashboard one is a combination of bobble charts and bar charts showing the relationship and the driven factors of loan Status, Loan Amount and Loan Purpose.

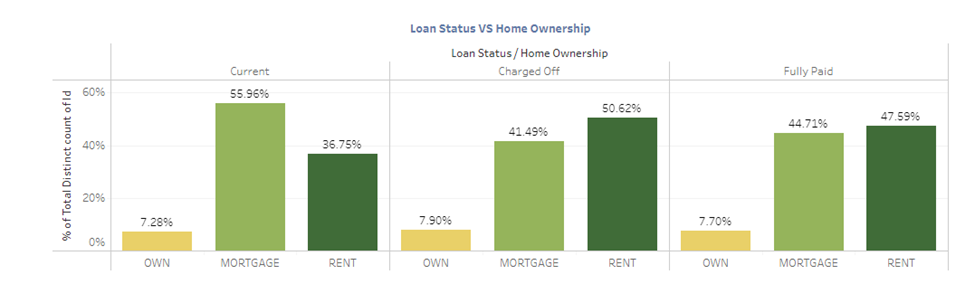
The bobble chart “Loan Purpose”:

We grouped the applicant by loan purpose then divided by total number of the applicants. It returned the percentage of the loan total by purpose. We then filtered the chart by loan status.



Bar Chart “Loan Status VS. Home ownership”

This chart breaks down the loan applicants by their Home Ownership (own, Mortgage, or Rent) then filtered by their loan status. With this chart, we can see people who own their house not necessarily most current or less likely to default on a Loan. Mortgage applicants are 56% current 42% Charged off and 45% Fully paid on their loan. Home owners are about 7%-8% regardless of the loan status.



Bar Chart “Loan Amount to Annual income Ratio by Loan Purpose and Loan Status”

This is chart showing some interesting factors on the loan applications. We are using Loan (debt) to Income(annual income) ratio to we if debt to income ratio truly plays a role on the loan status. Small business loan, debt consolidation and credit card debt makes most of the charge of population, their debt to income ratio is also high. However, this is also the same population have their loan fully aid. From this dataset we can say debt to income ratio not the golden driven factor for the loan status.

The second dashboard is a map of the loan applicant population by state. It shows the total number of loan applicants in each state as well as the percentage of loan status by each state. Color of the map is scaled by the percentage of fully paid to state total. It also shows the charge off, current and fully paid percentage by state.

**V. Website App**

We used a Flask App with a Python back end to build our website. Our website has five main sections and eleven pages in all. We designed our website on a Bootstrap grid with a Bootswatch theme.

1. **Home Page:** The home page introduces our bank and allows you to quickly connect to the forms and Tableau dashboards web pages. The About, DataTables, and References pages can be accessed from the Navigation Bar.
2. **Forms:** We created two forms—one for credit card approval and one for Loan Approval. The forms were built upon bootstrap and take the inputs are values in our datatable. We did not build error corrections into our forms, so the data that we enter into the forms has to be in the correct format and within the correct range. If we had time, we could have built this into the data input. Our form looks like this:

Graphical user interface, text, application

Description automatically generated

After you press the submission button (not shown), the data is read into JavaScript using the D3 library. We then use Ajax from JQuery to send the data to Flask, where the data is read in as a dictionary from a JSON format. The values of the dictionary are then converted into a DataFrame, and the datatypes and values are changed to reflect those in the dataset. We then one-hot encode the categorical columns, add the missing categorical columns, set those values to 0, and reorder the columns so that they will match the order that is needed for the machine learning model. Finally, we unpickle our LGB model, feed the data into it, obtain a prediction and a probability, and feed that back to the website to get a probability.

Unfortunately, we did not have a chance to connect the loan data to the machine learning model that we built.

1. **Tableau**: We created four Tableau dashboards—two for the credit card, one for the loan data, and one map. I used Tableau’s API to load the dashboards onto our website.
2. **Data Tables:** Our data tables for both data sets were really large, so we took a random sample of 1000 rows from each dataset. Then, I used the DataTable library from JavaScript to upload the data to our website
3. **About Us**: This section consists of the pictures from everyone in our group and some brief information about who we are.
4. **References:** This section talks about where we obtained our datatables from and points the reader to some links for more research
5. **Machine Learning Model:** This model talks a little bit about how the machine learning model works, how it makes predictions, and what the applicant should do so that they will have better luck next time.

**VI. Conclusions and Future Work**:

1. Which machine learning algorithm is best suited to correctly predicting whether credit cards are approved based on our data set? What about for Loans? How did we optimize the machine learning models

The tree models worked the best for the classification algorithms that we made. For credit cards, both the LGBMClassifier and the RandomForestClassifier gave spectacular results, but we picked the LGBMClassifier because it showed less evidence of overfitting. We did not try to optimize the algorithms too much, or even to use neural networks as the extra effort did not seem worth it based on how well the models are already working.

For our loan data, the RandomForestclassifier fit perfectly, so we did not need to check any other models.

1. What were the most important features of our machine learning algorithm, and how do those features affect the probability of credit card approval?

Age, number of months with a previous credit card, total income, and total years of employment are not surprisingly the categories that had the largest impact on the approval algorithm. For more details on how the probability of approval is affected by variation in these columns, please see section **III.**

1. Some questions from the dataset:
2. What is the relationship between home ownership and credit-card/loan approval?
3. What is the relationship between total income and credit card approval?
4. How does income vary with marital status?
5. What are the most common family-sizes for credit card approval?
6. What are the most common purposes for loan approval?
7. Do our model’s predictions match up with trends that we observe in our raw data?