Juliet Maharaj

Professor William Williams

5 February 2022

Credit Card Approval Model

A credit card is a type of payment card in which charges can be made against a line of credit. A credit card issuer can accept or deny an application for a credit card with reason. There are many different factors considered by credit card issuers when considering a credit card applicant. Some factors that are considered, but not limited to, are the applicants age, property ownership, income level and even the education level. Throughout this project, we will analyze a credit card approval dataset. The credit card approval dataset is from Kaggle.com.

To begin, we import our dataset in Python. This data is split into two tables connected by the ID, which is a unique identification number for each applicant. The other factors being considered in this data are shown below.

|  |  |
| --- | --- |
| Code\_gender | The gender of the applicant (male or female) |
| Flag\_own\_car | if the applicant owns a car or not |
| Flag\_own\_Realty | If the applicant owns a property |
| Cnt\_children | The number of children the applicant has |
| Amt\_income\_total | The annual income amount of the applicant |
| Name\_income\_amount | The income category of the applicant (working, commercial associate or other) |
| Name\_education\_type | The education level of the applicant (secondary, higher education or other) |
| Name\_family\_status | The applicants marital status (married, single/not married or other) |
| Name\_housing\_type | The applicants way of living (house/apartment, with parents or other) |
| Months\_balance | The month of the extracted data is the starting point, backwards, 0 is the current month, -1 is the previous month, and so on |
| Status | 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 |

An important step to analyzing data is cleaning the data. We can use data visualization techniques to consider which factors need cleaning. We can see that there are a significant number of missing values from the factor occupation type (see Appendix line 7). To keep our intended model, the most accurate it could be, we will remove the applicants who did not give their occupation type. The decision to exclude these applicants from our model is in accordance with the actions of most financial institutions. If you don’t include your occupation on an application the requires it, your application won’t even be considered. Another important step in data cleaning is to remove duplicate entries (see Appendix line 15). We know that entries in this dataset can be duplicates if their unique identification number is duplicated. One reason for a duplicate entry in this dataset may be because they have applied for the credit card more than once. Another step used to clean the data is to covert categorical factors to numerical outputs that can be used in our model (see Appendix line 18).

We used data visualization tools to analyze the different factors included in this data set (see Appendix line 13). From the output, we can that there are about three hundred thousand applicants that have applied for this credit card that are married. When a person identifies as married, their income is combined with their spouse. A combined income could potentially contribute to their approval or disproval. About fifty thousand applicants are single or not married. And less than fifty thousand people are either in a civil marriage, separated, or a widow. It is imperative to consider all the possibilities with the data given to us because it can potentially affect the model we are trying to create for predictive use.

The next step in analyzing this dataset is to perform feature selection. Feature Selection is a feature engineering component that involves the removal of irrelevant features and picks the best set of features to train a robust machine learning model. Some of the methods

Once we have all the features that are going to be included in our model, we use different methods to create the best model. The methods we are going to test are logistic regression, KNeighbors classifier, SVC, decision tree, random forest classifier, xgb classifier and cat boost classifier. After creating a model with each method, we run a ROC curve to detect which model is statistically better.

Appendix

In [7]

missing1 = msno.matrix(AR)

missing1.set\_title("Missing data for Application Records dataset", fontsize = 30)

Out [7]

A picture containing chart

Description automatically generated

In [15]

AR = AR.drop\_duplicates('ID', keep='last') *#remove duplicate values and keep the last entry of the ID if its repeated.*

AR.drop('OCCUPATION\_TYPE', axis=1, inplace=**True**) *#dropped the missing values in occupation type.*

In [18]

open1=pd.DataFrame(CC.groupby(["ID"])["MONTHS\_BALANCE"].agg(min))

open1=open1.rename(columns={'MONTHS\_BALANCE':'begin\_month'})

indiv=pd.merge(AR,open1,how="left",on="ID") *#merge to record data*

*#convert categoric features into numeric*

indiv["Gender"] = indiv['Gender'].replace(['F','M'],[0,1])

indiv["Own\_Car"] = indiv["Own\_Car"].replace(["Y","N"],[1,0])

indiv["Own\_Realty"] = indiv["Own\_Realty"].replace(["Y","N"],[1,0])

indiv["Is\_Working"] = indiv["Income\_Type"].replace(["Working","Commercial associate","State servant","Pensioner","Student"],[1,1,1,0,0])

indiv["In\_Relationship"] = indiv["Family\_Status"].replace(["Civil marriage","Married","Single / not married",

"Separated","Widow"],[1,1,0,0,0])

housing = {'House / apartment' : 'House / apartment',

'With parents': 'With parents',

'Municipal apartment' : 'House / apartment',

'Rented apartment': 'House / apartment',

'Office apartment': 'House / apartment',

'Co-op apartment': 'House / apartment'}

indiv["Housing\_Type"] = indiv['Housing\_Type'].map(housing)

household = {'Single / not married':'Single',

'Separated':'Single',

'Widow':'Single',

'Civil marriage':'Married',

'Married':'Married'}

indiv["Family\_Status"] = indiv["Family\_Status"].map(household)

education = {'Secondary / secondary special':'secondary',

'Lower secondary':'secondary',

'Higher education':'Higher education',

'Incomplete higher':'Higher education',

'Academic degree':'Academic degree'}

indiv["Education"] = indiv["Education"].map(education)

income = {'Commercial associate':'Working',

'State servant':'Working',

'Working':'Working',

'Pensioner':'Pensioner',

'Student':'Student'}

indiv["Income\_Type"] = indiv["Income\_Type"].map(income)

indiv["Household\_Size"] = indiv["Children\_Count"] + indiv["In\_Relationship"].apply(**lambda** x: 2 **if** x==1 **else** 1)

indiv["Age"] = round((indiv.Birthday/365)\*-1)

indiv["Experience"] = indiv.Employment\_Date/365

indiv['Experience']=indiv['Experience'].apply(**lambda** v : int(v\*-1) **if** v <0 **else** 0)

indiv=indiv.drop(columns=['Employment\_Date','Birthday','Children\_Count'])

indiv= pd.get\_dummies(indiv, columns=['Income\_Type', 'Education','Family\_Status',"Housing\_Type"])

In [13]

fig, axes = plt.subplots(1,2)

plot3=sns.countplot(y=AR.NAME\_FAMILY\_STATUS,linewidth=1.2, ax=axes[1])

plot3.set\_title("Customer Distribution by Family Status")

plot3.set\_xlabel("Count")

fig.set\_size\_inches(14,5)

plt.tight\_layout()

plt.show()

Out [13]

Chart, bar chart

Description automatically generated

In [59]

classifiers = {

"LogisticRegression" : LogisticRegression(),

"KNeighbors" : KNeighborsClassifier(),

"SVC" : SVC(C = 0.8,kernel='linear',probability=**True**),

"DecisionTree" : DecisionTreeClassifier(),

"RandomForest" : RandomForestClassifier(n\_estimators=250,max\_depth=12,min\_samples\_leaf=16),

"XGBoost" : XGBClassifier(max\_depth=12,

n\_estimators=250,

min\_child\_weight=8,

subsample=0.8,

learning\_rate =0.02,

seed=42),

"CatBoost" : CatBoostClassifier(iterations=250,

learning\_rate=0.2,

od\_type='Iter',

verbose=25,

depth=16,

random\_seed=42)

}

result\_table = pd.DataFrame(columns=['classifiers','accuracy','presicion','recall','f1\_score','fpr','tpr','auc'])

**for** key, classifier **in** classifiers.items():

classifier.fit(X\_train, y\_train)

y\_predict = classifier.predict(X\_test)

yproba = classifier.predict\_proba(X\_test)[::,1]

fpr, tpr, \_ = roc\_curve(y\_test, yproba)

auc = roc\_auc\_score(y\_test, yproba)

conf\_matrix = confusion\_matrix(y\_test,y\_predict)

result\_table = result\_table.append({'classifiers':key,

'accuracy':accuracy\_score(y\_test, y\_predict),

'presicion':precision\_score(y\_test, y\_predict, average='weighted'),

'recall':recall\_score(y\_test, y\_predict, average='weighted'),

'f1\_score':f1\_score(y\_test, y\_predict, average='weighted'),

'fpr':fpr,

'tpr':tpr,

'auc':auc

}, ignore\_index=**True**)

result\_table.set\_index('classifiers', inplace=**True**)

In [60]

fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(15,10))

**for** cls, ax **in** zip(list(classifiers.values()), axes.flatten()):

plot\_confusion\_matrix(cls,

X\_test,

y\_test,

ax=ax,

cmap='Blues')

ax.title.set\_text(type(cls).\_\_name\_\_)

plt.tight\_layout()

plt.show()

Diagram, Teams

Description automatically generatedOut [60]

In [61]

fig = plt.figure(figsize=(8,6))

**for** i **in** result\_table.index:

plt.plot(result\_table.loc[i]['fpr'],

result\_table.loc[i]['tpr'],

label="**{}**, AUC=**{:.3f}**".format(i, result\_table.loc[i]['auc']))

plt.plot([0,1], [0,1], color='orange', linestyle='--')

plt.xticks(np.arange(0.0, 1.1, step=0.1))

plt.xlabel("Flase Positive Rate", fontsize=15)

plt.yticks(np.arange(0.0, 1.1, step=0.1))

plt.ylabel("True Positive Rate", fontsize=15)

plt.title('ROC Curve Analysis', fontweight='bold', fontsize=15)

plt.legend(prop={'size':13}, loc='lower right')

plt.show()

Chart

Description automatically generatedOut [61]

In [62]

result\_table.iloc[:,:4]

Out [62]

Table

Description automatically generated