RISK CLASSIFICATION FOR INDIVIDUAL LOAN

BY : STEPHEN JAMES

Objective

Making a machine learning model from the given individual loan dataset to classify loans based on their risk: Low risk or High risk.

About the Dataset

A little **Summary** of this dataset:

- Based on the application type, this is a loan for individual data.
- There are 75 features originally.
- 17 empty features (100% null value).
- 17 Target Leakage.
- Target comes from the loan status feature.
- 20 features with null values (exclude the 100% empty).
- No duplicated rows based on member_id.
- 21 numerical features (exclude index, id, and member id).
- 17 categorical features.

Defining Target

- Took the target from the loan status feature:

```
loan_status
Current
Fully Paid
Charged Off
Late (31-120 days)
In Grace Period
Does not meet the credit policy. Status:Fully Paid
Late (16-30 days)
Default
Does not meet the credit policy. Status:Charged Off
```

```
loanstat = {
    'Fully Paid' : 'Low',
    'Charged Off' : 'High',
    'Late (31-120 days)' : 'High',
    'Does not meet the credit policy. Status:Fully Paid' : 'Low',
    'Default' : 'High',
    'Does not meet the credit policy. Status:Charged Off' : 'High'
}
```

- Define the target based on the **finished loan cycle** to make sure that the loan is fully paid to the investor or defaulted. Remove values such as: current, in grace period, and late(16-30 days) because we don’t know whether the borrower could fully pay the loan in the end. Late(31-120 days) is also considered a high-risk loan.

- There are 9 statuses and split into 2 categories, low and high risk:

Low (risk) : Fully Paid, Does not meet the credit policy. Status: Fully Paid

High (risk) : charged off, late(31-120 days), default, Does not meet the credit policy. Status: Charged Off

Target Leakage

Some features will not exist until the customer starts the loan. These features are called target leakage.

To prevent target leakage, any variable updated or created after the target value is realized should be excluded.

```
#Target leakage
df1 = df1.drop(columns=['out_prncp', 'out_prncp_inv', 'total_pymnt', 'total_pymnt_inv', 'total_rec_prncp',
                        'total_rec_int', 'total_rec_late_fee','recoveries', 'collection_recovery_fee', 'last_pymnt_d',
                        'last_pymnt_amnt', 'next_pymnt_d','last_credit_pull_d','funded_amnt','funded_amnt_inv','pymnt_plan'])
```

Terminologies:

- **out_prncp and out_prncp_inv**: Principal means the initial size of the loan or amount still owed on a loan.
- **total_pymnt and total_pymnt_inv**: How much payment the borrower already paid.
- **total_rec_prncp, total_rec_int, total_rec_late_fee**: Payment received to recorded date (principal, interest, late fee).
- **recoveries, collection_recovery_fee**: Recovery is the amount of money collected from a loan that continues to go unpaid.
- **last_pymnt_d, last_pymnt_amnt, next_pymnt_d, last_credit_pull_d**: Date when the borrower pay.
- **funded_amnt, funded_amnt_inv**: The amount of loan funded by the investor.
- **pymnt_plan**: Paying off any outstanding debt by means of consolidation into an organized payment schedule.
- **issue_d** / issue date is also a target leakage but will be removed later since it will be used in feature engineering.

Table of Contents

01

Exploratory Data Analysis

Exploratory Data Analysis

- Checking Duplicate

```
#Checking Duplicate
df1.duplicated(subset=['member_id']).sum()
```

```
0
```

- Checking Null Values

```
#Checking null values
df1.isna().sum()[df1.isna().sum()>0]
```

- Splitting data into numerical and categorical values

```
#Splitting numerical and categorical data
num = df1.select_dtypes(exclude=['object'])
kat = df1.select_dtypes(include=['object'])
```

Features with null values

```
emp_title                       13390
emp_length                       9156
annual_inc                          4
desc                           145726
title                              15
delinq_2yrs                        29
earliest_cr_line                   29
inq_last_6mths                     29
mths_since_last_delinq         132992
mths_since_last_record         208951
open_acc                           29
pub_rec                            29
revol_util                        231
total_acc                          29
collections_12_mths_ex_med        145
mths_since_last_major_derog    195525
acc_now_delinq                     29
tot_coll_amt                    66596
tot_cur_bal                     66596
total_rev_hi_lim                66596
dtype: int64
```

Exploratory Data Analysis
(numerical)

- Statistical Descriptive

```
#Statistical Descriptive
num.describe().T
```

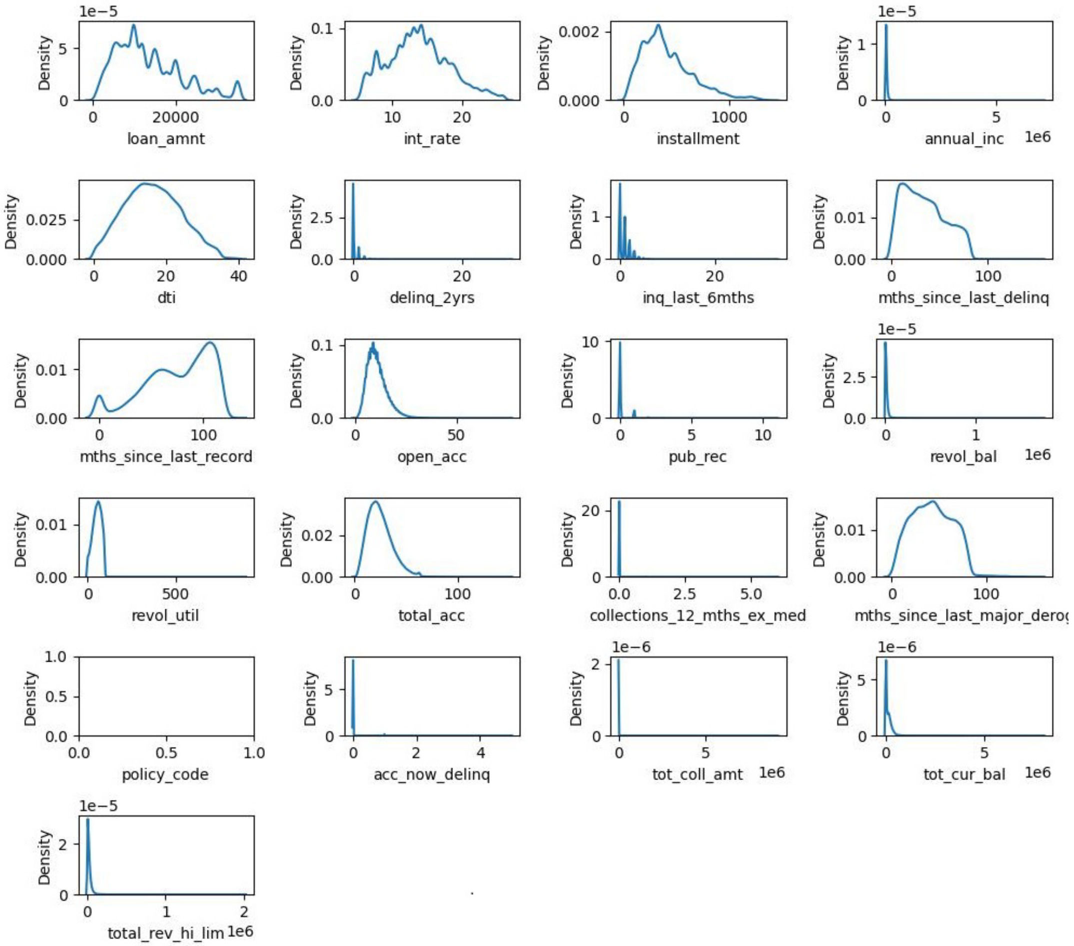
	count	mean	std	min	25%	50%	75%	max
loan_amnt	237695.0	13474.354320	8061.451689	500.00	7200.00	12000.00	18000.00	35000.00
int_rate	237695.0	13.844646	4.378830	5.42	10.99	13.67	16.59	26.06
installment	237695.0	416.623498	243.631791	15.67	239.18	365.01	545.33	1408.13
annual_inc	237691.0	71926.294970	55163.505912	1896.00	45000.00	61421.00	86000.00	7141778.00
dti	237695.0	16.428473	7.694727	0.00	10.71	16.13	21.87	39.99
delinq_2yrs	237666.0	0.247297	0.733771	0.00	0.00	0.00	0.00	29.00
inq_last_6mths	237666.0	0.906869	1.173854	0.00	0.00	1.00	1.00	33.00
mths_since_last_delinq	104703.0	34.936563	21.842771	0.00	16.00	32.00	51.00	152.00
mths_since_last_record	28744.0	75.259428	31.690372	0.00	54.00	80.00	102.00	129.00
open_acc	237666.0	10.854047	4.825149	0.00	7.00	10.00	13.00	76.00
pub_rec	237666.0	0.134567	0.420952	0.00	0.00	0.00	0.00	11.00
revol_bal	237695.0	15222.810421	19208.372721	0.00	5911.00	10988.00	19067.00	1746716.00
revol_util	237464.0	54.970257	24.679938	0.00	37.20	56.60	74.50	892.30
total_acc	237666.0	24.807402	11.663237	1.00	16.00	23.00	32.00	150.00
collections_12_mths_ex_med	237550.0	0.005860	0.082882	0.00	0.00	0.00	0.00	6.00
mths_since_last_major_derog	42170.0	42.933910	21.490371	0.00	26.00	42.00	60.00	154.00
policy_code	237695.0	1.000000	0.000000	1.00	1.00	1.00	1.00	1.00
acc_now_delinq	237666.0	0.002886	0.058452	0.00	0.00	0.00	0.00	5.00
tot_coll_amt	171099.0	200.864593	22186.615831	0.00	0.00	0.00	0.00	9152545.00
tot_cur_bal	171099.0	136587.550862	150226.180681	0.00	27953.50	79289.00	206419.00	8000078.00
total_rev_hi_lim	171099.0	29119.743511	28564.910190	0.00	13200.00	22000.00	36200.00	2013133.00

Exploratory Data Analysis
(numerical)

- Univariate

Numerical features distribution

```
#univariate
plt.figure(figsize=(10,10))
for i in range(0,len(num1)):
  plt.subplot(7,4,i+1)
  sns.kdeplot(x=df1[num1[i]])
  plt.tight_layout()
```



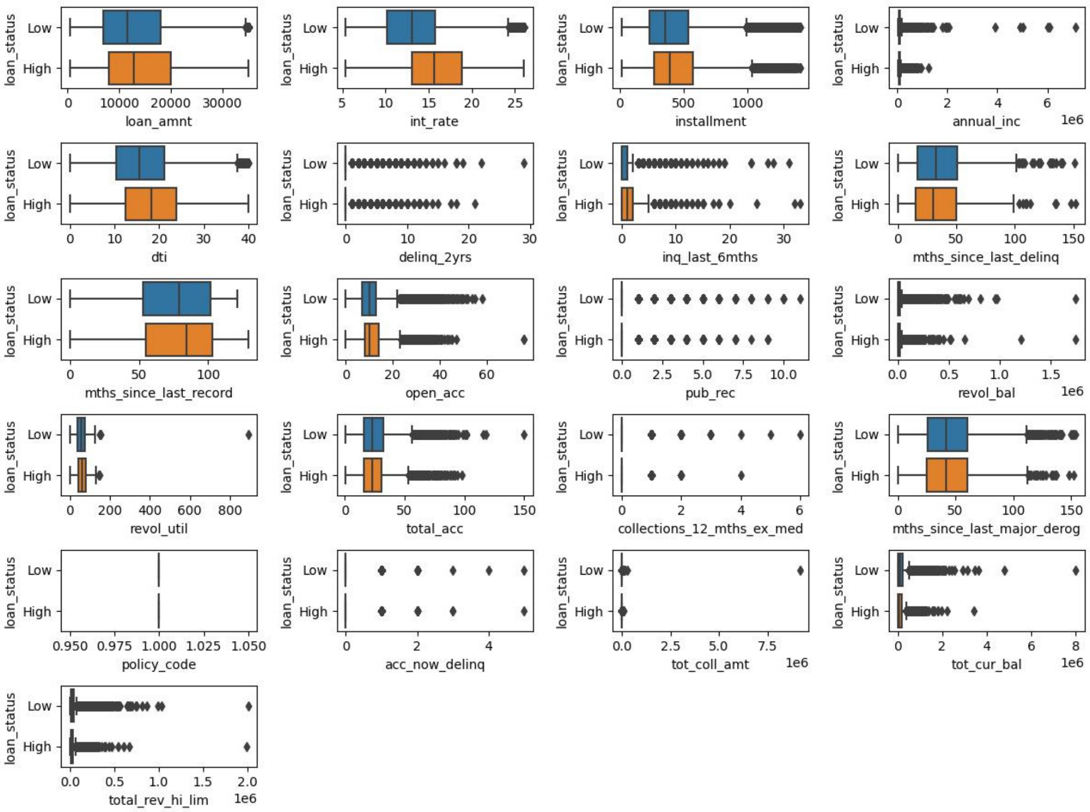
Exploratory Data Analysis
(numerical)

- Bivariate

Analyzing the correlation between the target and numerical features.

From this boxplot, we could also see that almost all numerical feature has outliers

```
#bivariate
plt.figure(figsize=(12,10))
for i in range (0,len(num1)):
  plt.subplot(7,4,i+1)
  sns.boxplot(data=df1,x=df1[num1[i]],y='loan_status')
  plt.tight_layout()
```



Exploratory Data Analysis
(categorical)

- Statistical Descriptive

```
#Statistical Descriptive
kat.describe().T
```

	count	unique	top	freq
term	237695	2	36 months	185700
grade	237695	7	B	71990
sub_grade	237695	35	B3	17316
emp_title	224305	129469	Teacher	1633
emp_length	228539	11	10+ years	70939
home_ownership	237695	6	MORTGAGE	116769
verification_status	237695	3	Verified	88337
issue_d	237695	91	Oct-14	9704
loan_status	237695	2	Low	186727
url	237695	237695	https://www.lendingclub.com/browse/loanDetail....	1
desc	91969	91234		230
purpose	237695	14	debt_consolidation	138318
title	237680	49854	Debt consolidation	59800
zip_code	237695	874	945xx	3044
addr_state	237695	50	CA	40386
earliest_cr_line	237666	634	Oct-00	2027
initial_list_status	237695	2	f	177046
application_type	237695	1	INDIVIDUAL	237695

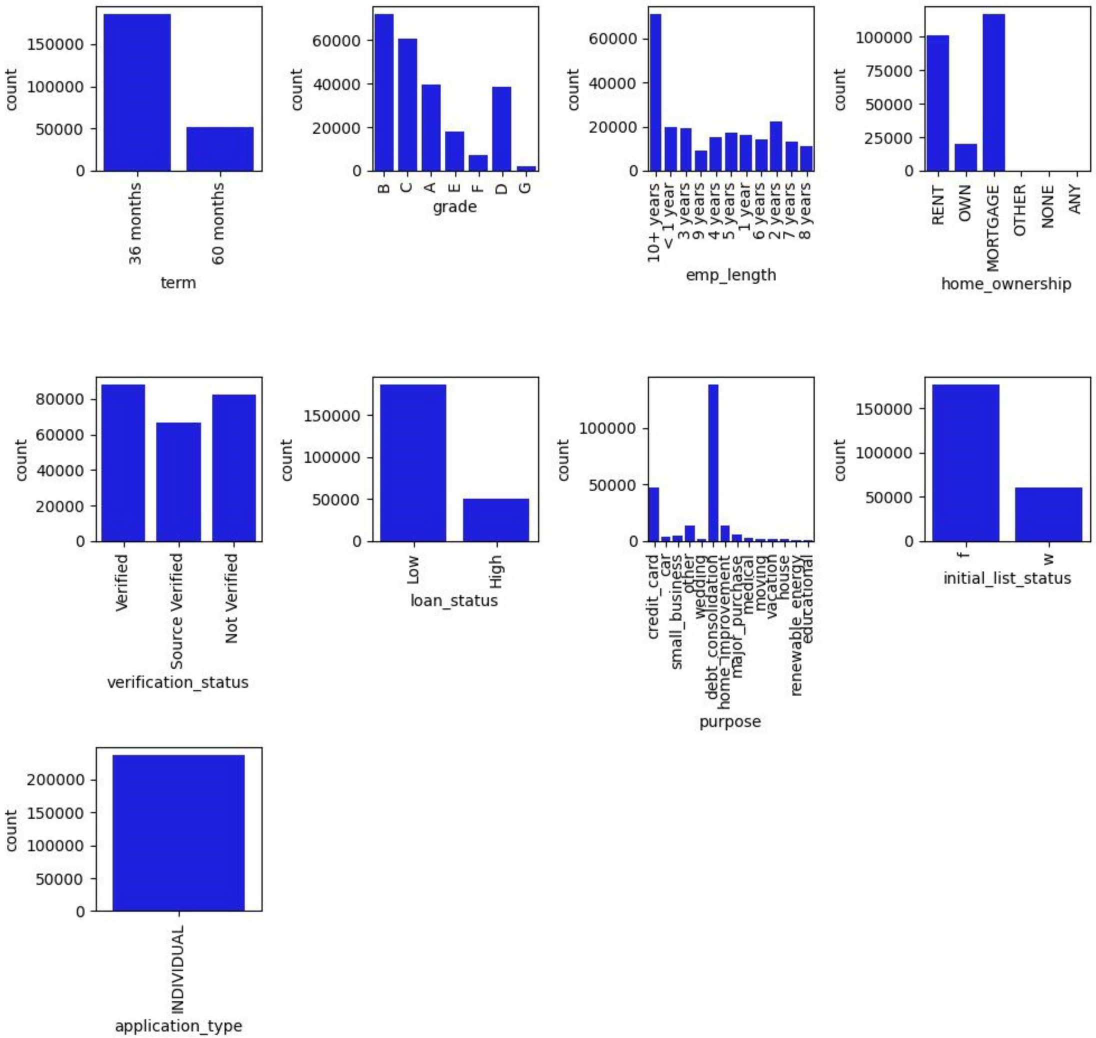
Exploratory Data Analysis
(categorical)

- Univariate

Categorical features count using countplot for nunique <15

```
kat1 = kat.loc[:,kat.nunique()<15]
kat1 = kat1.columns.tolist()
len(kat1)
```

```
#Univariate
plt.figure(figsize=(10,15))
for i in range(0,len(kat1)):
  plt.subplot(5,4,i+1)
  sns.countplot(x=df1[kat1[i]],color='blue')
  plt.tight_layout()
  plt.xticks(rotation=90)
```

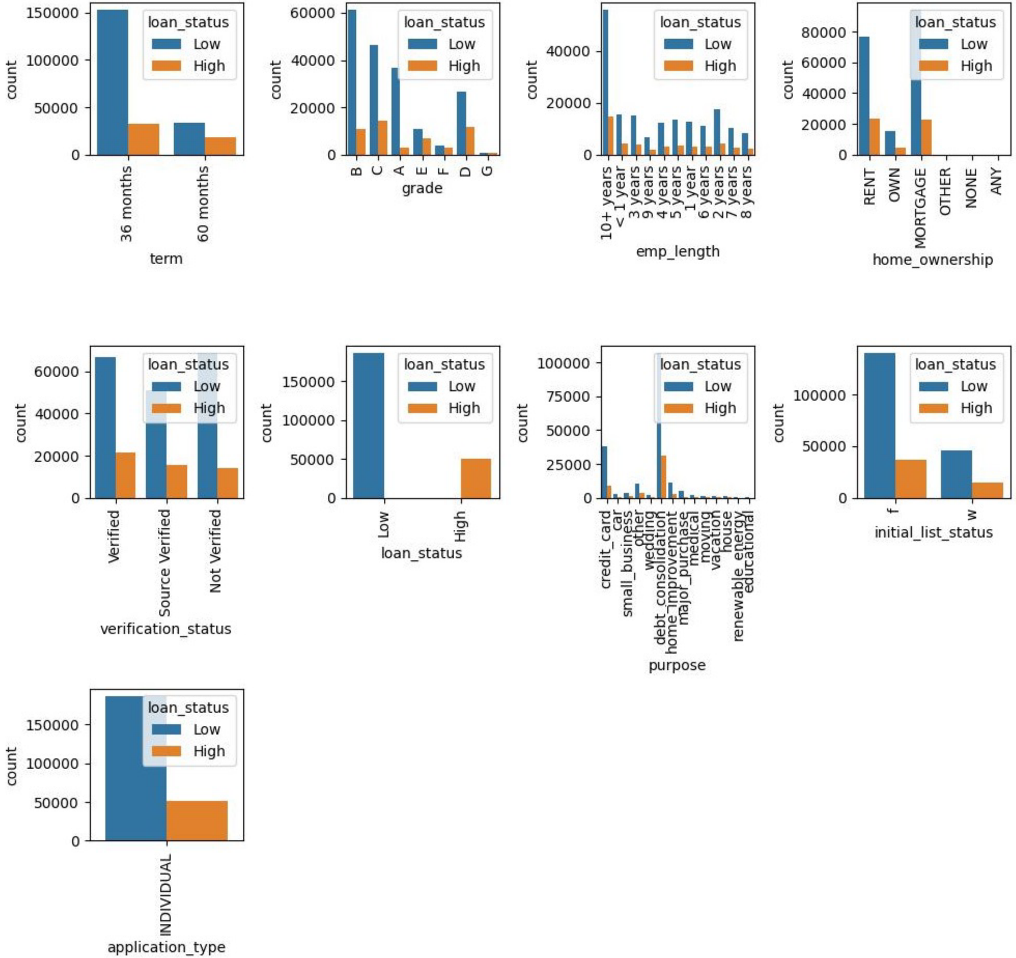


Exploratory Data Analysis
(categorical)

- Bivariate

Analyzing the correlation between the target and categorical features

```
#Bivariate
plt.figure(figsize=(10,15))
for i in range(0,len(kat1)):
  plt.subplot(5,4,i+1)
  sns.countplot(x=df1[kat1[i]],hue=df1['loan_status'])
  plt.tight_layout()
  plt.xticks(rotation=90)
```



02

Preprocessing

Preprocessing

- Unnecesary Columns

```
#Unnecessary Columns
df2 = df2.drop(columns=['policy_code','sub_grade','emp_title','url','desc','title',
                        'zip_code','addr_state','application_type'],axis=1)
```

- **policy_code and application_type**: only has 1 unique value
- **sub_grade**: already represented by grade
- **emp_title, url, desc, title, zip_code, addr_state**: too much unique values

- Handle missing value (1):

- Remove null value higher than 50%

```
for i in df2:
  if (df2[i].isna().mean())*100 > 50:
    df2 = df2.drop(i,axis=1)
```

- Fill missing emp_length value with zero

```
emplength = {
    '10+ years' : 10,
    '2 years' : 2,
    '< 1 year' : 0,
    '3 years' : 3,
    '5 years' : 5,
    '1 year' : 1,
    '4 years' : 4,
    '6 years' : 6,
    '7 years' : 7,
    '8 years' : 8,
    '9 years' : 9
}
df2['emp_length'] = df2['emp_length'].replace(emplength)
df2['emp_length'] = df2['emp_length'].fillna(0)
```

Handle Missing Values (2)

- Fill some features with median

```
fill_median = ['open_acc','delinq_2yrs','inq_last_6mths','pub_rec','revol_util','collections_12_mths_ex_med',
               'acc_now_delinq','tot_coll_amt','tot_cur_bal']

df2[fill_median] = df2[fill_median].fillna(df2[fill_median].median())
```

- Removing rows with missing value

```
df2 = df2.dropna(subset=['annual_inc','earliest_cr_line'], how='any', axis=0)
```

Feature Engineering

- Creating new features: Credit length history

Time deficit from earliest credit line to issue date.

In this dataset, the issue date should be a target leakage, but I will use it assuming that the issued date time will not be far away from the time borrower applying for the loan. Later, the issue date and earliest credit line features will be dropped.

```
#checking earliest credit line range

pd.set_option('display.max_rows',None)

year = df2['earliest_cr_line'].str.split('-').str[1]
month = df2['earliest_cr_line'].str.split('-').str[0]
year = pd.to_numeric(year)
year.value_counts(dropna=False)
```

- Checking the earliest credit manually because when it converted to DateTime, the year below 1969 automatically converted to 2068, etc.

- Turns out that the earliest credit is from 1946.

```
year = year.apply(lambda x: x+2000 if x<46 else x+1900)
year = year.apply(lambda x: f'{x:g}')
year = year.astype(str)
```

Function to get the correct year

```
date = month + "-" + year
date = pd.to_datetime(date, format = '%b-%Y')
```

```
df2['cr_length_history'] = (df2['issue_d'] - date).dt.days
df2['cr_length_history'].describe()
```

```
count    237666.000000
mean       5528.583251
std        2561.731327
min         184.000000
25%        3834.000000
50%        5053.000000
75%        6787.000000
max       24138.000000
```

Feature Transformation

- One hot encoding:

- purpose

Grouping other purposes besides debt consolidation and credit card since they have the most count.

- home ownership

Moved other and any values in home ownership into none because they have a very low count compared to other.

- initial list status, term, and verification status

```
#purpose
df2.loc[(df2.purpose != 'debt_consolidation')&(df2.purpose != 'credit_card'),'purpose'] = 'other'
df2['purpose'].value_counts()
df2 = pd.get_dummies(df2,columns=['purpose'])
```

```
#home_ownership
df2['home_ownership'] = df2['home_ownership'].str.lower()
ho = {
    'other' : 'none',
    'any' : 'none'
}
df2['home_ownership'] = df2['home_ownership'].replace(ho)
df2 = pd.get_dummies(df2,columns=['home_ownership'])
```

```
#initial_list_status
df2 = pd.get_dummies(df2,columns=['initial_list_status'])
```

```
#term
df2 = pd.get_dummies(df2,columns=['term'])
```

```
#verification_status
df2 = pd.get_dummies(df2,columns=['verification_status'])
```

Feature Transformation

- Label encoding:

 - grade

Changed from A to G became 1 to 7.

 - target

Changed low risk to 1 and high risk to 0

```
#grade
grade = {
    'A': 1,
    'B': 2,
    'C': 3,
    'D': 4,
    'E': 5,
    'F': 6,
    'G': 7
}
df2['grade'] = df2['grade'].replace(grade)
```

```
df2 = df2.rename(columns={'loan_status':'target'})
target= {
    'Low' : 1,
    'High' : 0
}
df2['target'] = df2['target'].replace(target)
```

Feature Selection

Based on the correlation heatmap, I will remove features that:

- Redundant

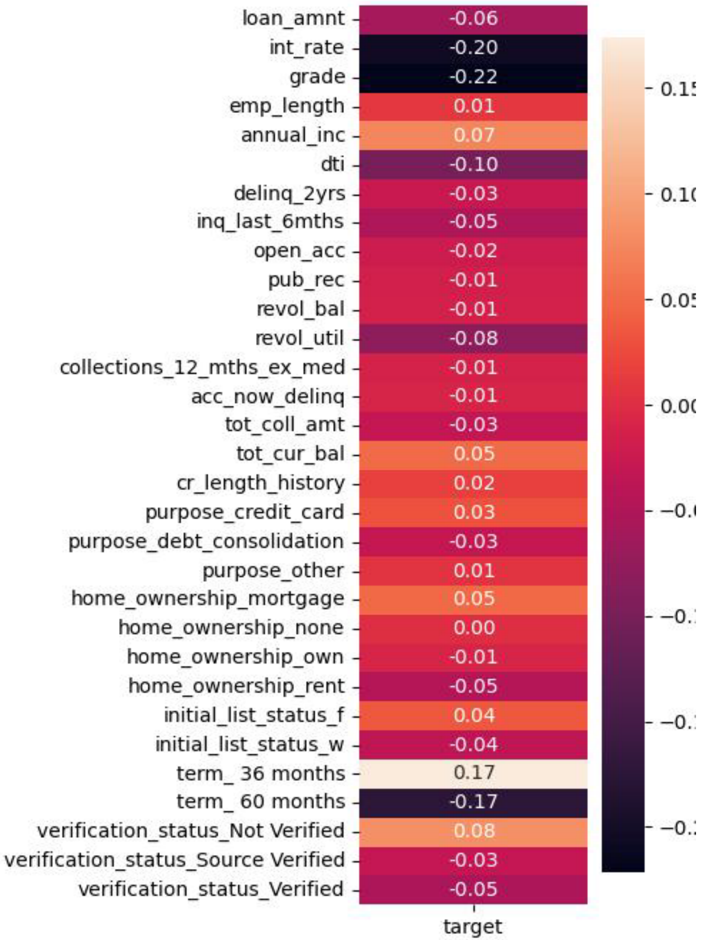
Two features with high correlation to each other (initial list status, term 36 months, int rate)

- Low correlation

Features with low correlation with target (emp length, pub rec, revol bal, collection 12 mhts ex med, acc now delinq, purpose other, home ownership none, home ownership own)

- Grade

Decided to remove grade features as that feature is really deciding the result. Wanted to see the impact of other features without the grade feature



03

Modelling

Modelling

- Split into 2 variable, target (y) and features (X)

```
y = df3['target']
X = df3[df3.columns.drop('target')]
```

```
y.value_counts()/len(df3)*100
```

```
1    78.556041
0    21.443959
Name: target, dtype: float64
```

- Train test split

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=20,stratify=y)
```

- Scaling with standardization

```
#Scaling
from sklearn.preprocessing import StandardScaler,MinMaxScaler,RobustScaler

ss = StandardScaler()

X_train = ss.fit_transform(X_train)
X_test = ss.fit_transform(X_test)
```

Modelling

- Machine learning

Fit some models into the dataset and this is the result:

	accuracy (train)	accuracy (test)	precision (train)	precision (test)	recall (train)	recall (test)	f1 (train)	f1 (test)	roc(train)	roc(test)
knn	0.817306	0.757966	0.833008	0.801455	0.959415	0.921186	0.891755	0.857159	0.833681	0.592641
logreg	0.785948	0.789818	0.790942	0.795084	0.988337	0.988027	0.878690	0.881116	0.679152	0.680225
decisiontree	1.000000	0.678331	1.000000	0.806884	1.000000	0.778217	1.000000	0.792291	1.000000	0.542269
adaboost	0.786483	0.789537	0.795681	0.799407	0.979240	0.978580	0.877969	0.879965	0.692164	0.690487
randomforst	0.999994	0.787223	0.999992	0.799317	1.000000	0.974843	0.999996	0.878398	1.000000	0.676135
gradientboost	0.788118	0.789986	0.792255	0.795220	0.989279	0.988027	0.879872	0.881200	0.700911	0.695673
xgboost	0.803259	0.786985	0.807204	0.801175	0.984260	0.970680	0.886982	0.877820	0.763007	0.687798

From this credit risk classification, F1 is used as a metrics evaluation in order to **minimalized false positive**. Investor doesn't want a loan that is indicated as a low risk loan but in the end, the borrower defaults and investor ended up losing money. Based on that, logistic regression and Adaboost used.

Modelling

- Logistic Regression

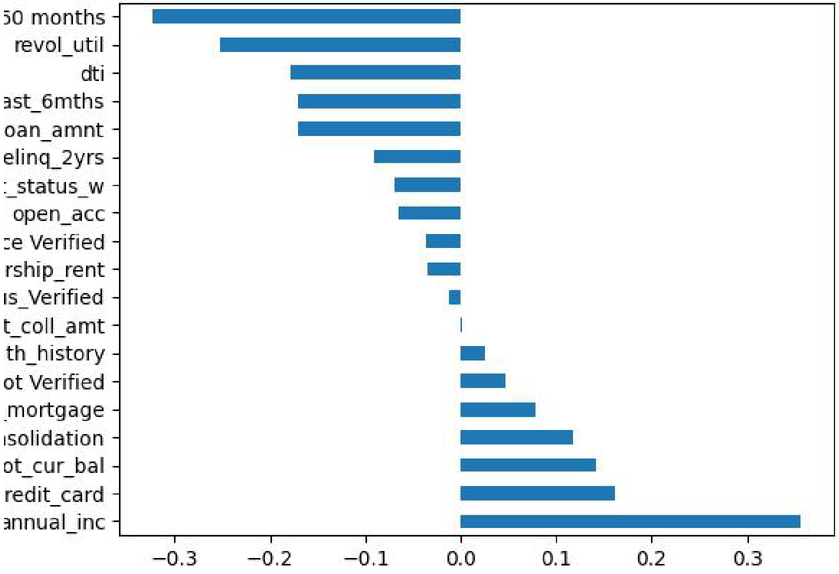
From this model, F1 Score resulted in 0,88 for test and train data

```
lr.fit(X_train,y_train)
y_pred_lr = lr.predict(X_test)
eval_classification(lr)
```

```
{'accuracy (train)': 0.7873002897226596,
 'accuracy (test)': 0.7864796633941094,
 'precision (train)': 0.7923053802224226,
 'precision (test)': 0.7915951727293525,
 'recall (train)': 0.9883159513661997,
 'recall (test)': 0.98841278343153,
 'f1 (train)': 0.8795222563446072,
 'f1 (test)': 0.8791227986597431,
 'roc(train)': 0.680588602256615,
 'roc(test)': 0.6759222128067619}
```

```
importance = lr.coef_[0]
importance =  pd.Series(importance,index=X.columns)
importance.nlargest(20).plot(kind='barh')
```

Feature Importance



From feature importance, we could see that annual income becomes the most important feature that would make a borrower paid their loan. 5 years term and revolving utilization rate play a big role in defaulted loan.

Modelling

- Adaboost

F1 resulted in 0,87 before and after tuning .

```
ab.fit(X_train,y_train)
y_pred_ab = ab.predict(X_test)
eval_classification(ab)
```

```
{'accuracy (train)': 0.7870237909188176,
 'accuracy (test)': 0.7859046283309958,
 'precision (train)': 0.7959034063728931,
 'precision (test)': 0.7949842897064998,
 'recall (train)': 0.980258778339748,
 'recall (test)': 0.9802535261560436,
 'f1 (train)': 0.878513581161239,
 'f1 (test)': 0.8779512764545504,
 'roc(train)': 0.6929091225773207,
 'roc(test)': 0.6774407360219825}
```

```
ada_importance = ab.feature_importances_
ada_importance = pd.Series(ada_importance,index=X.columns)
ada_importance.nlargest(20).plot(kind='barh')
```

tuning

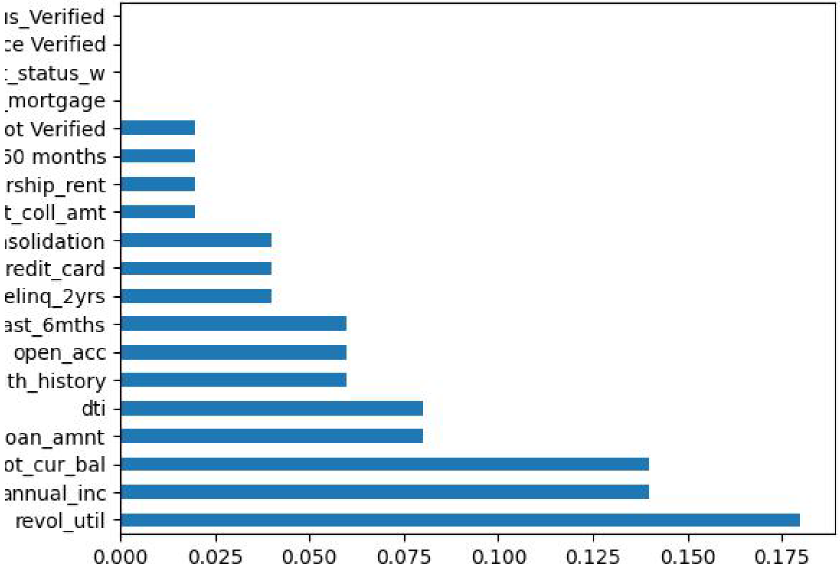
```
# List of hyperparameter
hyperparameters = dict(n_estimators = [int(x) for x in np.linspace(start = 50, stop = 2000, num = 200)],
                       learning_rate = [float(x) for x in np.linspace(start = 0.001, stop = 0.1, num = 100)],
                       algorithm = ['SAMME', 'SAMME.R']
                      )

# Init model
ab_tuned = RandomizedSearchCV(ab, hyperparameters, random_state=42, cv=5, scoring='f1')
ab_tuned.fit(X_train,y_train)

# Predict & Evaluation
eval_classification(ab_tuned)
```

```
{'accuracy (train)': 0.78689756320402,
 'accuracy (test)': 0.786437587657784,
 'precision (train)': 0.7889327103937868,
 'precision (test)': 0.7882965036546917,
 'recall (train)': 0.9948963585862837,
 'recall (test)': 0.9954829494733083,
 'f1 (train)': 0.8800240946731145,
 'f1 (test)': 0.8798573468727562,
 'roc(train)': 0.6887305041521602,
 'roc(test)': 0.67716132513476}
```

Modelling

- Adaboost

Feature importance



Revolving utilization rate, annual income, and total current balance have the most influence on AdaBoost results.

Conclusion

From both models used logistic regression and adaboost, the F1 score are quite the same about 0,87 -0,88.

Revolving utilization rate and annual income appeared in both models’ feature importances. Showing that these two features play an important role in classifying loan risk.

Thanks

Created by: Stephen James

Ign.james4@gmail.com
https://www.linkedin.com/in/stephenjames123/