VEHICLE LOAN DEFAULT PREDICTION

USING

MACHINE LEARNING ALGORITHM

FINAL REPORT

PGP-DSE

BY:

Group1

MENTOR:

Ankush Bansal

Members
Meghana S
Nikhitha Siddi
Sai Venkata Shreya Gudipati
Shubham Dey
Titash Sengupta

PROBLEM STATEMENT

- Financial institutions often face significant losses due to the default of vehicle loans. This has led to the tightening up of vehicle loan underwriting and there is an increase in vehicle loan rejection rates.
- The need for a better credit risk predicting model is what is demanded by these institutions. This warrants a study to estimate the determinants of vehicle loan default.
- A financial institution has a requirement to accurately predict the probability of loanee/borrower defaulting on a vehicle loan.
- Following Information regarding the loan and loanee are taken into consideration in the datasets:
    - Loanee Information (Demographic data like age, Identity proof etc.)
    - Loan Information (Disbursal details, loan to value ratio etc.)
    - Bureau data & history (Bureau score, number of active accounts, the status of other loans, credit history etc.)
- This project will help to ensure that clients who are capable of repayment are not rejected and important determinants can be identified which can be further used for minimizing the default rates on these loans.

PROBLEM SOLVING METHODOLOGY

Step 1: Data preprocessing and EDA

- Null value treatment - 7661 null values in the Employment Type column and we imputed with mode.
- Outlier Treatment using IQR capping, square root transformation, log transformation and box cox transformation for relevant features.
- Univariate / Bivariate Analysis to understand the relationship between the independent features and also the relationship between the independent and dependent features.
- Feature Engineering: loanee age, loan age, state ID, credit history length and average account age were all feature engineered to reduce the complexity in the column.

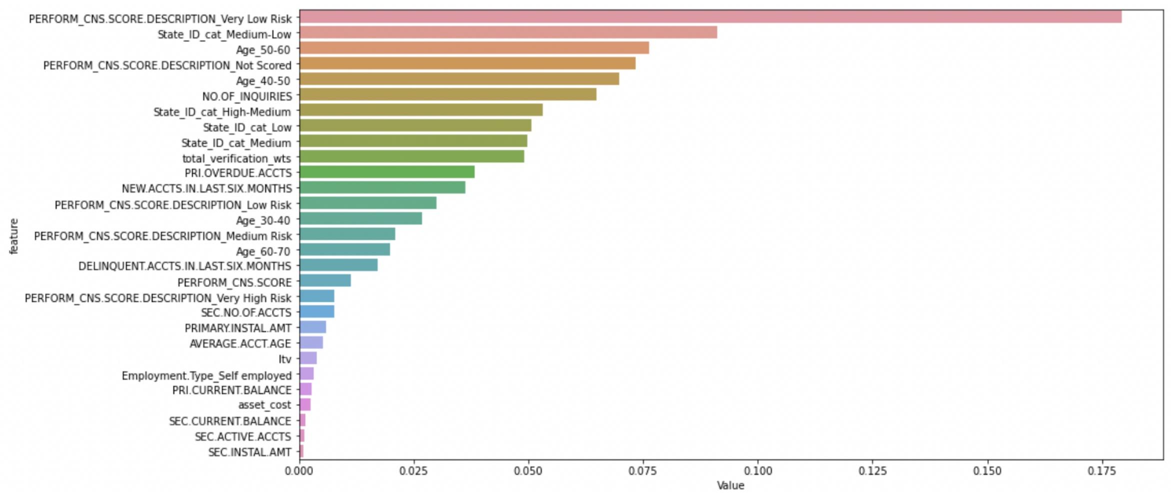
Step 2: Base Model Building

- Checking significance of the features using Chi Square Test for Categorical Features and Independent T-test for Numerical Features.
- Removing features with a lot of uniqueness.
- Multicollinearity check for Numerical Features using VIF by taking the threshold as 5.
- Reducing imbalance in data by oversampling using SMOTE.
- Models Built for Base Model: Logistic Regression, Decision Tree, Random Forest, AdaBoost, Gradient Boost, XGBoost, Naive Bayes, Ensemble of all models.

Steps 3: Tuning Base Model into Final Model

- We chose XG Boost as our base model as it had the highest accuracy scores. We tuned the model by trying individual parameter tuning and selecting the parameter value with the best score.

Step 4: Salient Features of the data



The above plot shows the significance of each feature in the model. The top 5 features are -

1. Perform_CNS_Score_Description_Very Low Risk
2. State_ID-Medium Low
3. Age - 50-60
4. Perform_CNS_Score_Description_Not scored
5. Age - 40-50

WALK THROUGH

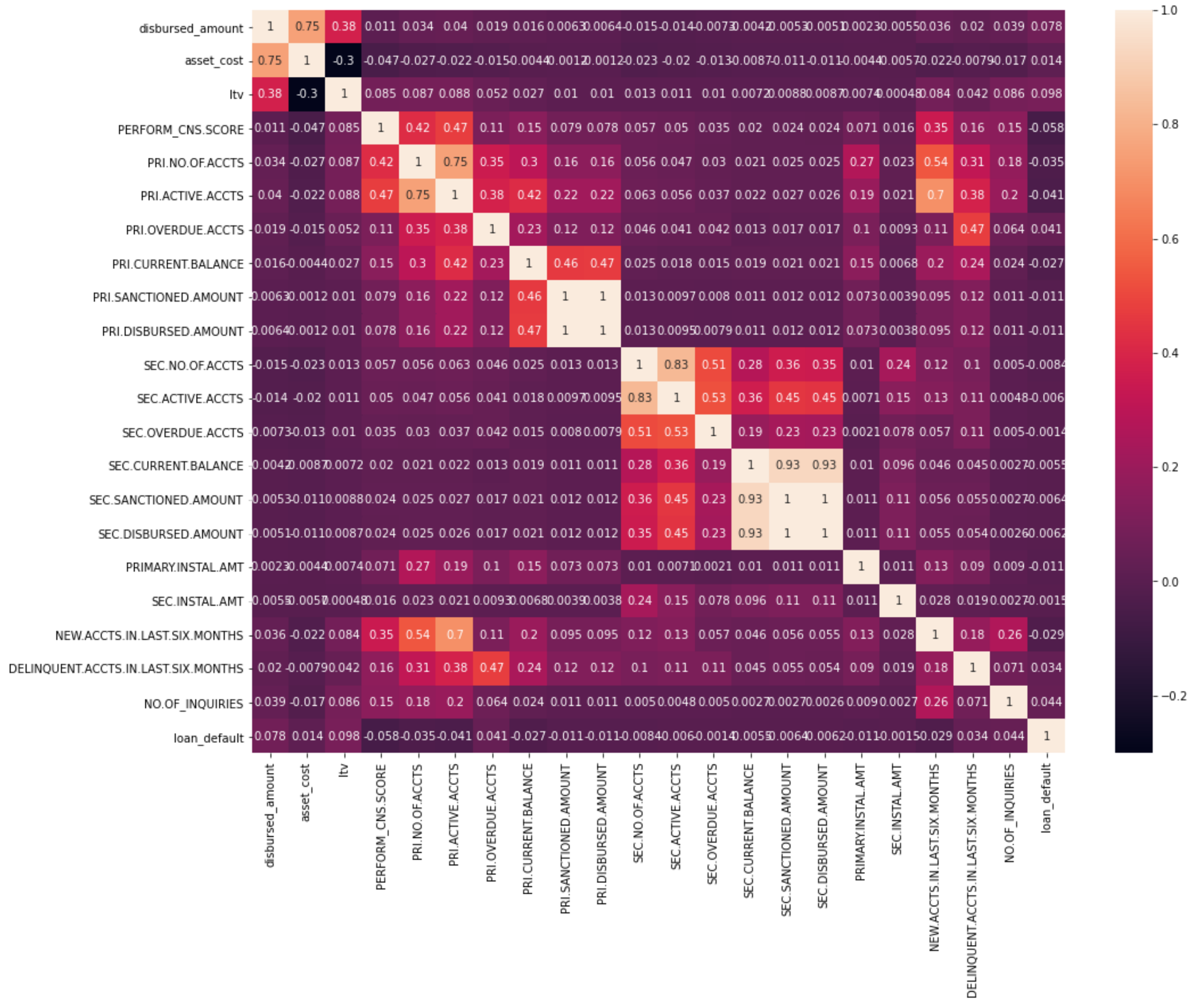
The steps taken to solve the problem was in the following stages:

- Business Understanding
  - Understanding the problem and the business requirement of the financial institutes along with the business pain-points due to which this project was undertaken.
  - Here we came to the conclusion that banks are losing out businesses in the money lending segment (loans) as due to strict underwriting laws. Therefore there is a need for a loan default prediction system which will help banks to identify who will default and who will not, thereby making the lending business profitable by reducing any sort of bad debts.
- Data Understanding
  - In this stage we took up historic data of the year 2018 to get some insights of a banking dataset and the different types of features that are usually essential of a financial institute to give out loans to the public.

- Some important features that we came across are like Asset-cost, Loan to Value of asset(ltv), CNS-score, Primary Account details(applicant), Secondary Account details(co-applicant), Age,Average account age, Verification flags for numerical features. In categorical features - Employment type, State_ID (after feature engineering), loanee age, CNS Score description are the important features.

- Data Preparation
  - In this stage, we performed Data Preprocessing and EDA.
  - We removed the null values, did outlier treatment and performed feature engineering. We also performed univariate analysis on the independent features and bivariate analysis on the independent features with the dependent feature.

Correlation:



Inferring from the above heat map, the following variables have the highest correlation of **0.93**.

- Sec Current Balance and Sec Sanctioned Amount

- Sec Current Balance and Sec Disbursed Amount
- LTV and Asset Cost

The following variables have the second highest correlation of **0.83.**

- Sec Number of Accts and Sec Active Accts.

The following variables have the correlation of **0.75.**

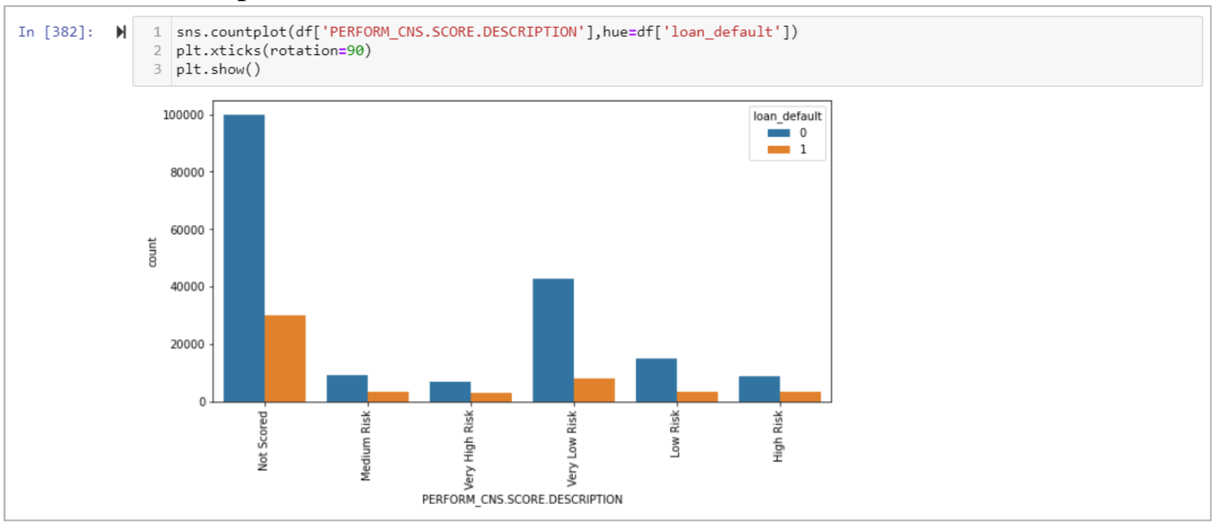
- Primary No of Accts and Primary Active Accounts
- Disbursed amount and Asset Cost

The following variables have the correlation of **0.7.**

- Primary Act Accts and New Accts in Last six Months

Bivariate Analysis:

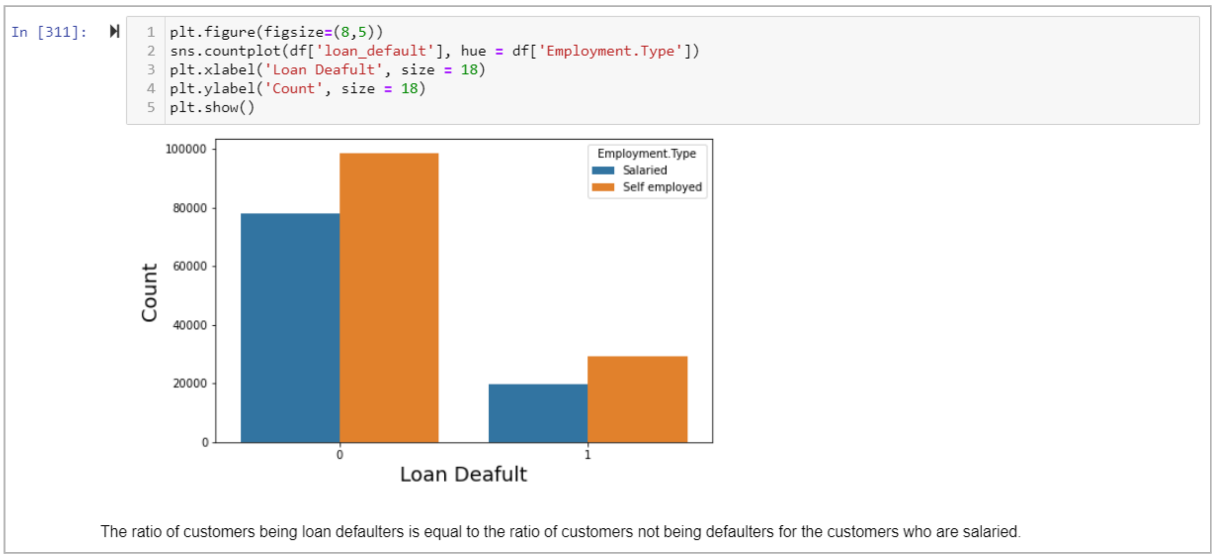
CNS score Description and Loan Default



```
In [382]:
sns.countplot(df['PERFORM_CNS.SCORE.DESCRIPTION'],hue=df['loan_default'])
plt.xticks(rotation=90)
plt.show()
```

As compared to scored ones, the probability of defaulters in Not Scored customers is higher. Similarly, non-defaulters are also higher in Not Scored customers, so it is difficult to infer from this plot.

Employment Type and Loan Default

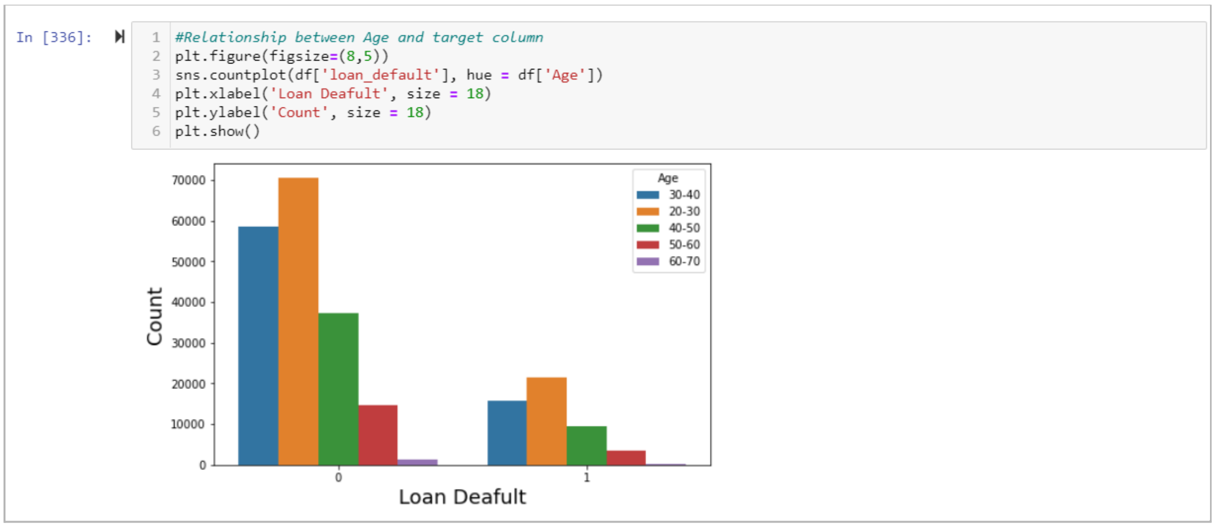


```
In [311]:
plt.figure(figsize=(8,5))
sns.countplot(df['loan_default'], hue = df['Employment.Type'])
plt.xlabel('Loan Deafult', size = 18)
plt.ylabel('Count', size = 18)
plt.show()
```

The ratio of customers being loan defaulters is equal to the ratio of customers not being defaulters for the customers who are salaried.

The ratio of customers being loan defaulters is equal to the ratio of customers not being defaulters for the customers who are salaried.

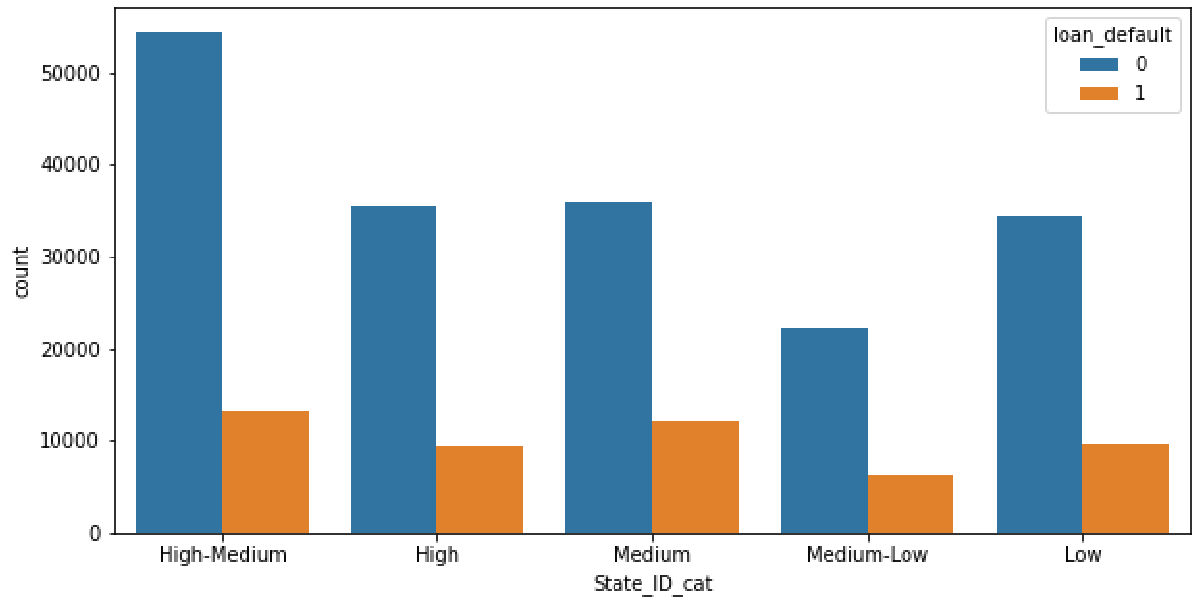
Age and Loan Default



```
In [336]:
#Relationship between Age and target column
plt.figure(figsize=(8,5))
sns.countplot(df['loan_default'], hue = df['Age'])
plt.xlabel('Loan Deafult', size = 18)
plt.ylabel('Count', size = 18)
plt.show()
```

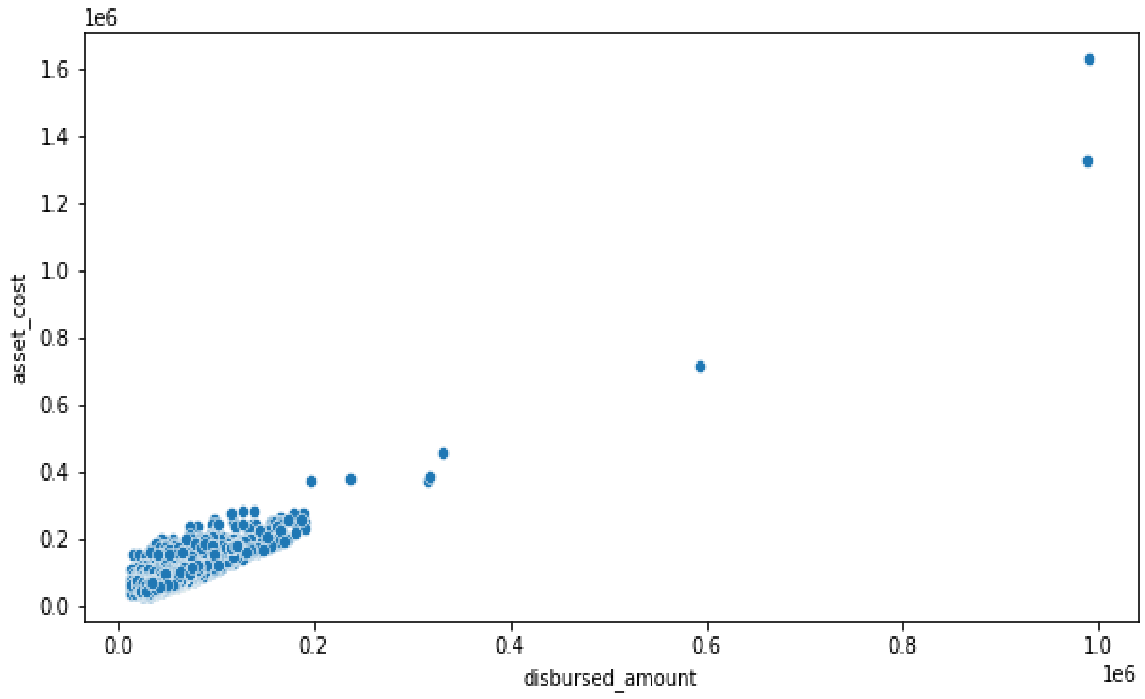
The highest number of loanees are in the range of 20-30 years followed by 30-40 years.

State_ID_Cat and Loan Default



The highest number of loanees is for the High-Medium category. The lowest number of loanees are from the Medium-Low category.

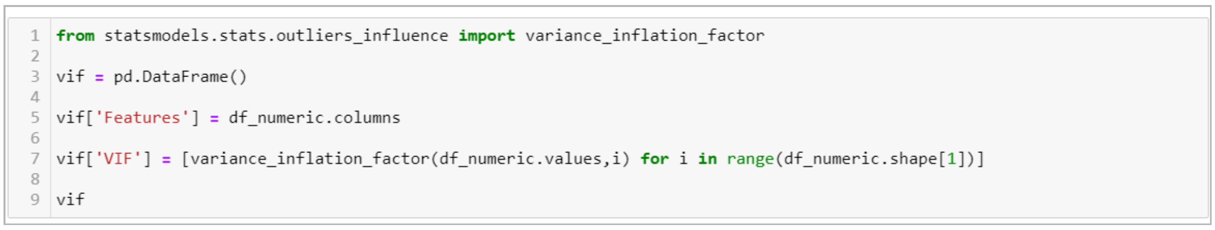
Asset Cost and Disbursed Amount



We infer that asset cost and disbursed amount have positive correlation.

Check for multicollinearity:

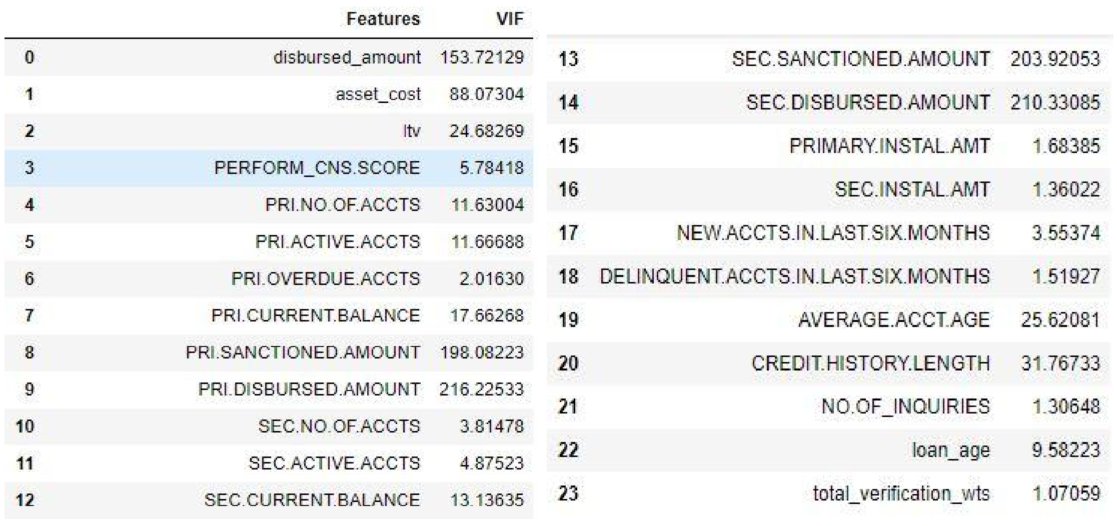
Multicollinearity is checked through VIF as shown below



```python
from statsmodels.stats.outliers_influence import variance_inflation_factor

vif = pd.DataFrame()

vif['Features'] = df_numeric.columns

vif['VIF'] = [variance_inflation_factor(df_numeric.values,i) for i in range(df_numeric.shape[1])]

vif
```

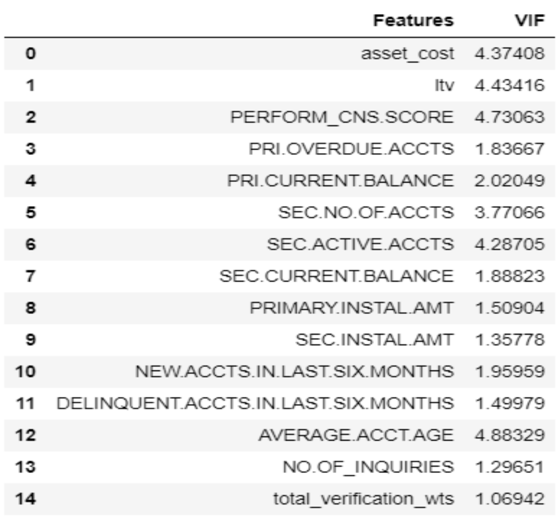
This code is run multiple times by removing the feature that has a high VIF value. The before and after filtering of the features is shown below.

Before:



	Features	VIF
0	disbursed_amount	153.72129
1	asset_cost	88.07304
2	ltv	24.68269
3	PERFORM_CNS.SCORE	5.78418
4	PRI.NO.OF.ACCTS	11.63004
5	PRI.ACTIVE.ACCTS	11.66688
6	PRI.OVERDUE.ACCTS	2.01630
7	PRI.CURRENT.BALANCE	17.66268
8	PRI.SANCTIONED.AMOUNT	198.08223
9	PRI.DISBURSED.AMOUNT	216.22533
10	SEC.NO.OF.ACCTS	3.81478
11	SEC.ACTIVE.ACCTS	4.87523
12	SEC.CURRENT.BALANCE	13.13635
13	SEC.SANCTIONED.AMOUNT	203.92053
14	SEC.DISBURSED.AMOUNT	210.33085
15	PRIMARY.INSTAL.AMT	1.68385
16	SEC.INSTAL.AMT	1.36022
17	NEW.ACCTS.IN.LAST.SIX.MONTHS	3.55374
18	DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS	1.51927
19	AVERAGE.ACCT.AGE	25.62081
20	CREDIT.HISTORY.LENGTH	31.76733
21	NO.OF_INQUIRIES	1.30648
22	loan_age	9.58223
23	total_verification_wts	1.07059

VIF Threshold taken was 5 and the output is shown below:



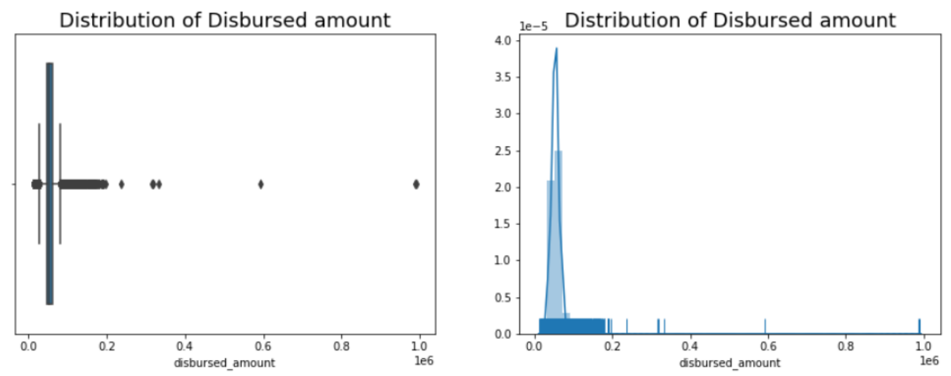
	Features	VIF
0	asset_cost	4.37408
1	ltv	4.43416
2	PERFORM_CNS.SCORE	4.73063
3	PRI.OVERDUE.ACCTS	1.83667
4	PRI.CURRENT.BALANCE	2.02049
5	SEC.NO.OF.ACCTS	3.77066
6	SEC.ACTIVE.ACCTS	4.28705
7	SEC.CURRENT.BALANCE	1.88823
8	PRIMARY.INSTAL.AMT	1.50904
9	SEC.INSTAL.AMT	1.35778
10	NEW.ACCTS.IN.LAST.SIX.MONTHS	1.95959
11	DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS	1.49979
12	AVERAGE.ACCT.AGE	4.88329
13	NO.OF_INQUIRIES	1.29651
14	total_verification_wts	1.06942

Presence of outliers and its treatment

For the outliers present in the data, based on the column, different treatments were done.

- For Disbursed Amount column, clipping was done

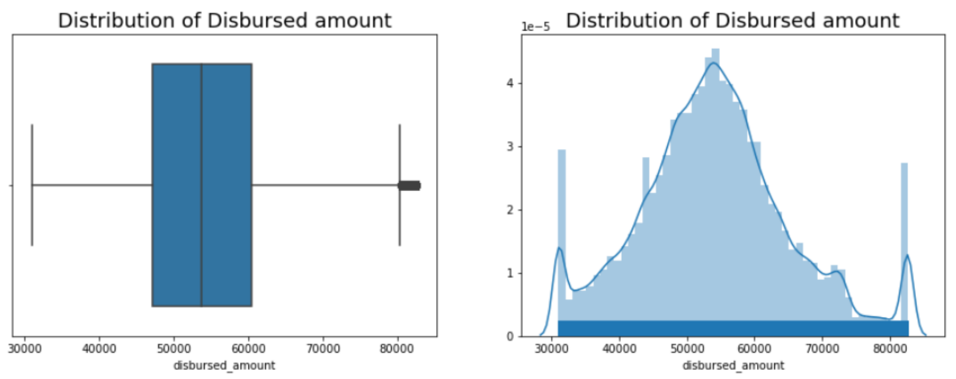
```
In [367]:
plt.figure(figsize=(15,5))
plt.subplot(1,2,1)
dist_plot = sns.boxplot(df['disbursed_amount'])
plt.title('Distribution of Disbursed amount', size = 18)
plt.subplot(1,2,2)
dist_plot = sns.distplot(df['disbursed_amount'], rug= True)
plt.title('Distribution of Disbursed amount', size = 18)

plt.show()
```



Capping to get rid of Outliers

```
In [368]:
df['disbursed_amount'] = df['disbursed_amount'].clip(lower =df['disbursed_amount'].quantile(0.025),
                                                      upper =  df['disbursed_amount'].quantile(0.975))
```

```
In [369]:
plt.figure(figsize=(15,5))
plt.subplot(1,2,1)
dist_plot = sns.boxplot(df['disbursed_amount'])
plt.title('Distribution of Disbursed amount', size = 18)
plt.subplot(1,2,2)
dist_plot = sns.distplot(df['disbursed_amount'], rug= True)
plt.title('Distribution of Disbursed amount', size = 18)

plt.show()
```



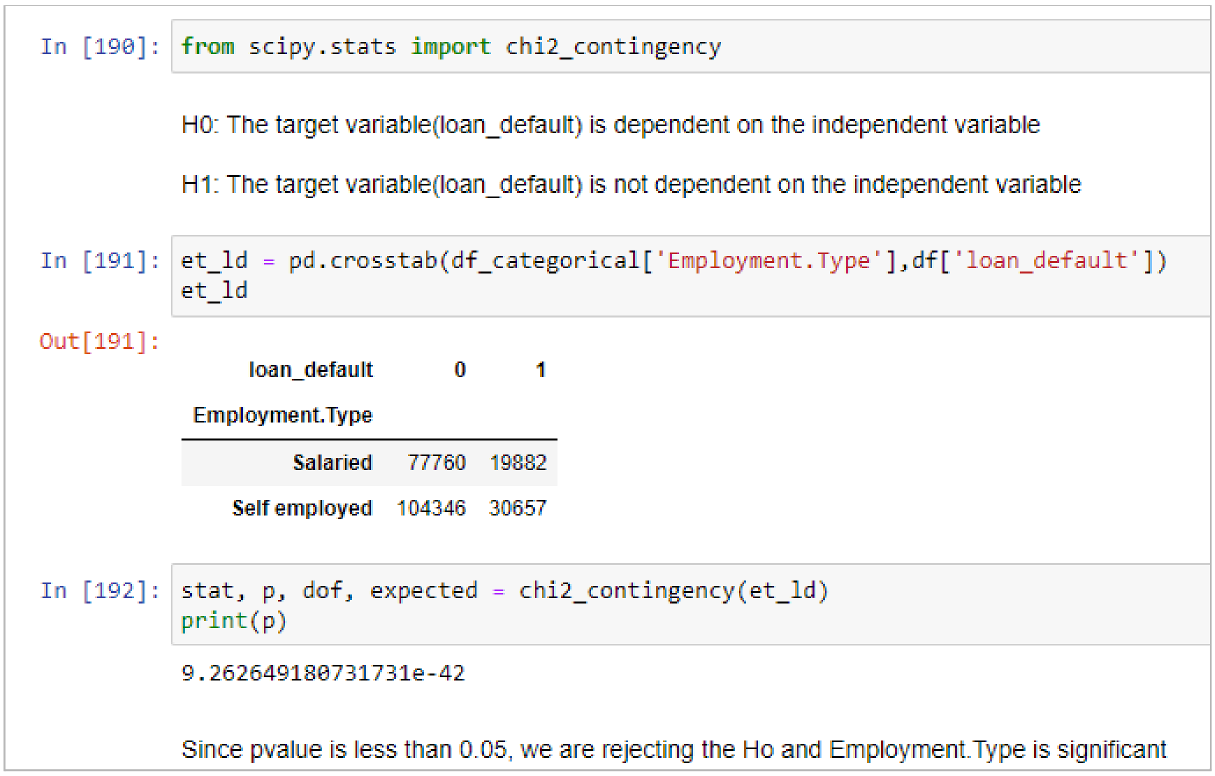
- For the ltv column, box cox transformation was done to reduce the outliers.
- For asset cost column log transformation was done.
- For Perform_cns.Score, Perform_cns.Score.Description, Pri.No.Of.Accts Pri.Active.Accts, Pri.Overdue.Accts, Pri.Current.Balance, Pri.Sanctioned.Amount, Pri.Disbursed.Amount, Sec.No.Of.Accts, Sec.Active.Accts, Sec.Overdue.Accts, Sec.Current.Balance, Sec.Sanctioned.Amount, Sec.Disbursed.Amount, Primary.Instal.Amt, Sec.Instal.Amt, New.Accts.In.Last.Six.Months, Delinquent.Accts.In.Last.Six.Months, Average.Acct.Age, Credit.History.Length,

No.Of_inquiries columns Squareroot Transformation was done.

Statistical significance of variables

Statistical significance of categorical features with the target column is checked through chi2 and that of numerical and target column is checked through ttest_ind.

Chi2 for each categorical Vs target is shown below



```
In [190]: from scipy.stats import chi2_contingency
```

H0: The target variable(loan_default) is dependent on the independent variable

H1: The target variable(loan_default) is not dependent on the independent variable

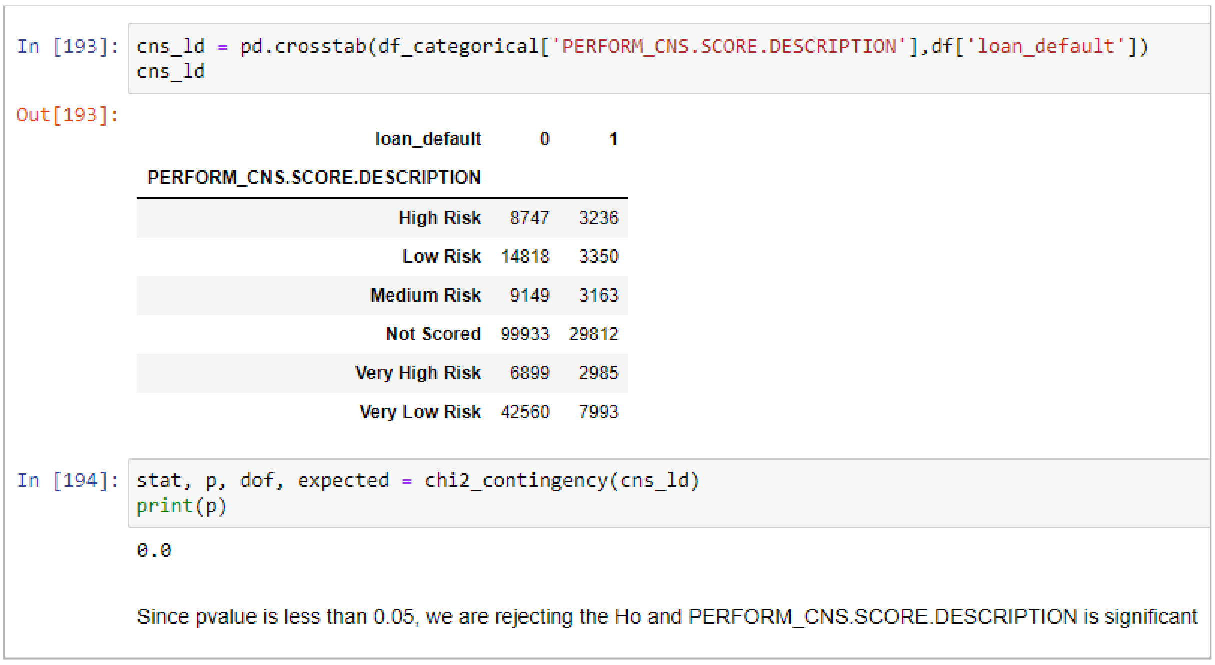
```
In [191]: et_ld = pd.crosstab(df_categorical['Employment.Type'],df['loan_default'])
          et_ld
```

Out[191]:

loan_default	0	1
Employment.Type		
Salaried	77760	19882
Self employed	104346	30657

```
In [192]: stat, p, dof, expected = chi2_contingency(et_ld)
          print(p)

9.262649180731731e-42
```

Since pvalue is less than 0.05, we are rejecting the Ho and Employment.Type is significant



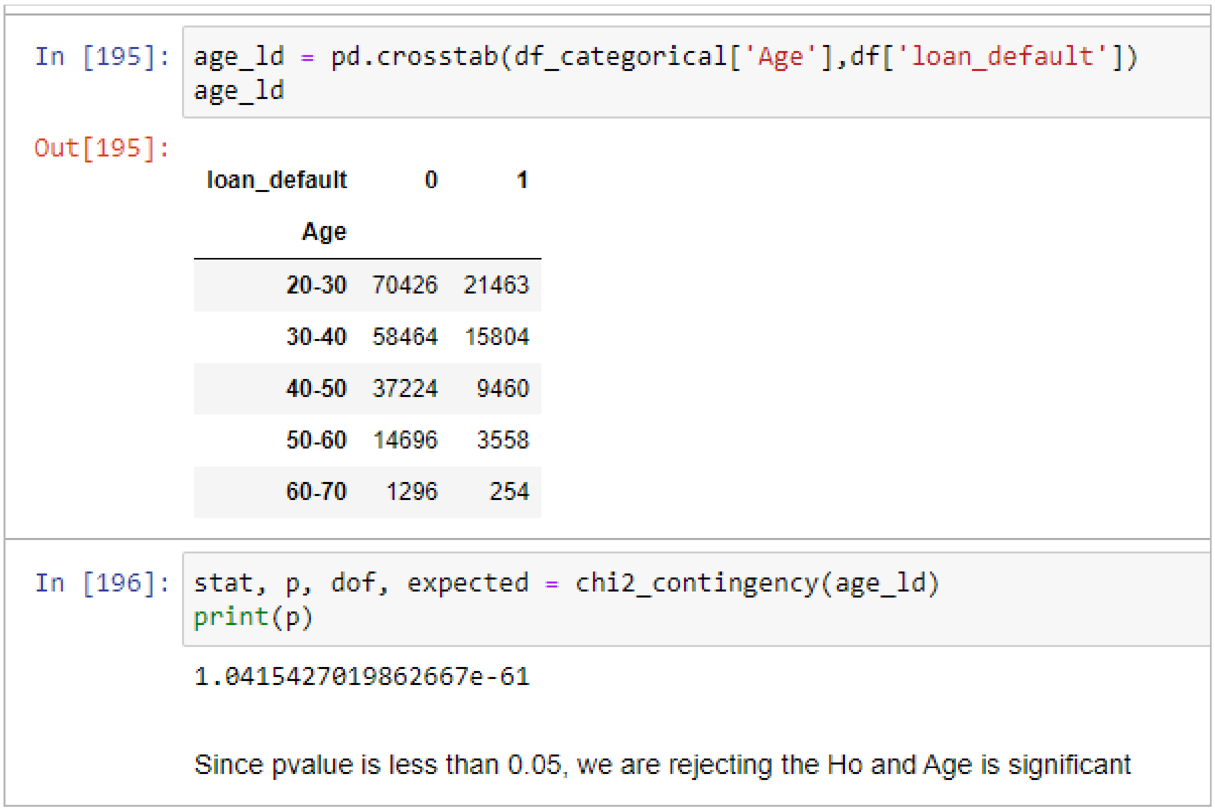
```
In [193]: cns_ld = pd.crosstab(df_categorical['PERFORM_CNS.SCORE.DESCRIPTION'],df['loan_default'])
          cns_ld
```

Out[193]:

loan_default	0	1
PERFORM_CNS.SCORE.DESCRIPTION		
High Risk	8747	3236
Low Risk	14818	3350
Medium Risk	9149	3163
Not Scored	99933	29812
Very High Risk	6899	2985
Very Low Risk	42560	7993

```
In [194]: stat, p, dof, expected = chi2_contingency(cns_ld)
          print(p)

0.0
```

Since pvalue is less than 0.05, we are rejecting the Ho and PERFORM_CNS.SCORE.DESCRIPTION is significant



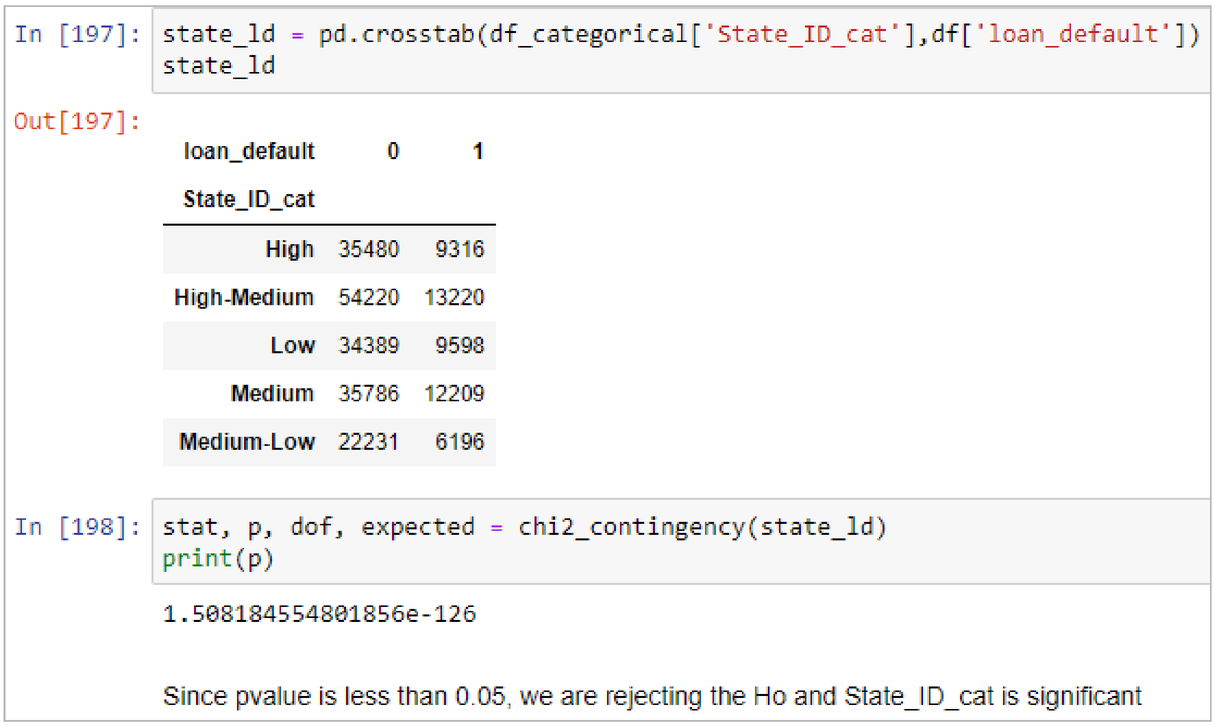
```
In [195]: age_ld = pd.crosstab(df_categorical['Age'],df['loan_default'])
          age_ld
```

Out[195]:

loan_default	0	1
Age		
20-30	70426	21463
30-40	58464	15804
40-50	37224	9460
50-60	14696	3558
60-70	1296	254

```
In [196]: stat, p, dof, expected = chi2_contingency(age_ld)
          print(p)

1.0415427019862667e-61
```

Since pvalue is less than 0.05, we are rejecting the Ho and Age is significant



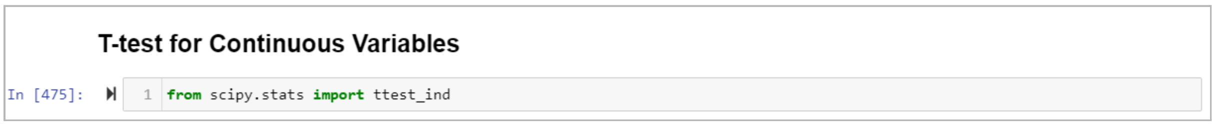
```
In [197]: state_ld = pd.crosstab(df_categorical['State_ID_cat'],df['loan_default'])
          state_ld
Out[197]:
```

loan_default	0	1
State_ID_cat		
High	35480	9316
High-Medium	54220	13220
Low	34389	9598
Medium	35786	12209
Medium-Low	22231	6196

```
In [198]: stat, p, dof, expected = chi2_contingency(state_ld)
          print(p)

1.508184554801856e-126
```

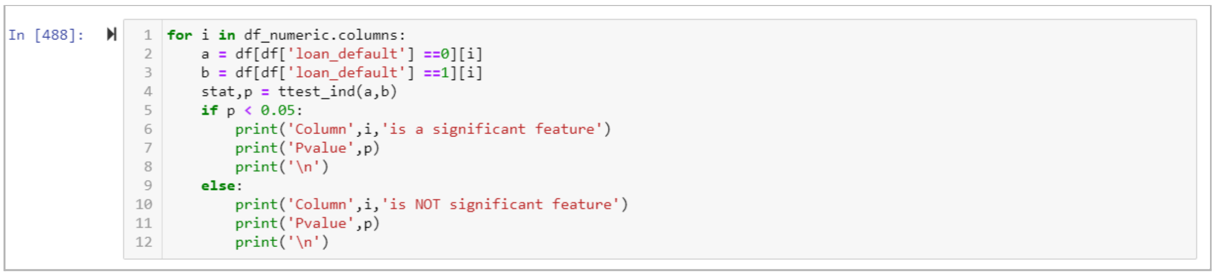
Since pvalue is less than 0.05, we are rejecting the Ho and State_ID_cat is significant

Ttest for each numerical Vs target is done with a for loop.



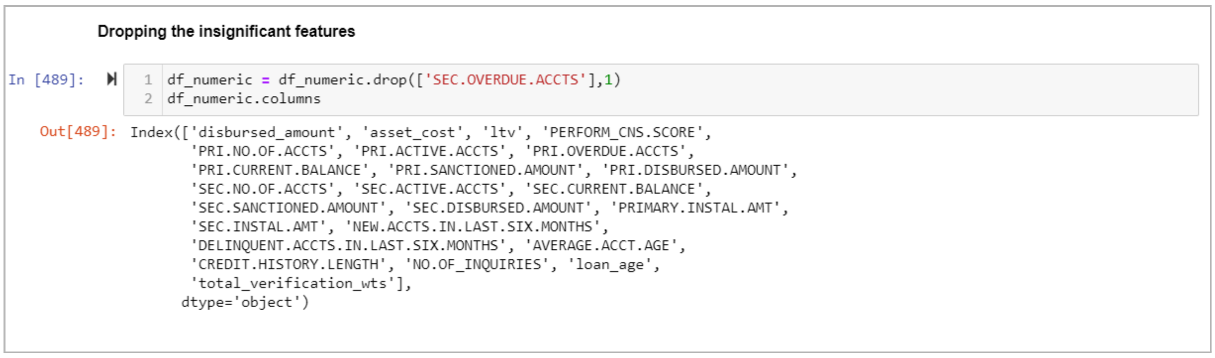
T-test for Continuous Variables

```
In [475]: from scipy.stats import ttest_ind
```



```
In [488]: for i in df_numeric.columns:
              a = df[df['loan_default'] ==0][i]
              b = df[df['loan_default'] ==1][i]
              stat,p = ttest_ind(a,b)
              if p < 0.05:
                  print('Column',i,'is a significant feature')
                  print('Pvalue',p)
                  print('\n')
              else:
                  print('Column',i,'is NOT significant feature')
                  print('Pvalue',p)
                  print('\n')
```

All the features in both chi2 and ttest have pvalue less than 0.05 except SEC.OVERDUE.ACCTS, hence dropping this insignificant feature.

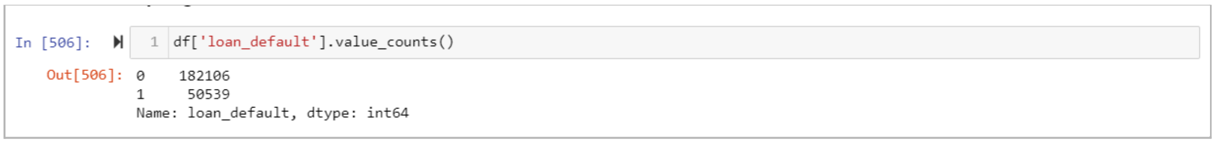


Dropping the insignificant features

```
In [489]: df_numeric = df_numeric.drop(['SEC.OVERDUE.ACCTS'],1)
          df_numeric.columns

Out[489]: Index(['disbursed_amount', 'asset_cost', 'ltv', 'PERFORM_CNS.SCORE',
                 'PRI.NO.OF.ACCTS', 'PRI.ACTIVE.ACCTS', 'PRI.OVERDUE.ACCTS',
                 'PRI.CURRENT.BALANCE', 'PRI.SANCTIONED.AMOUNT', 'PRI.DISBURSED.AMOUNT',
                 'SEC.NO.OF.ACCTS', 'SEC.ACTIVE.ACCTS', 'SEC.CURRENT.BALANCE',
                 'SEC.SANCTIONED.AMOUNT', 'SEC.DISBURSED.AMOUNT', 'PRIMARY.INSTAL.AMT',
                 'SEC.INSTAL.AMT', 'NEW.ACCTS.IN.LAST.SIX.MONTHS',
                 'DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS', 'AVERAGE.ACCT.AGE',
                 'CREDIT.HISTORY.LENGTH', 'NO.OF_INQUIRIES', 'loan_age',
                 'total_verification_wts'],
                dtype='object')
```

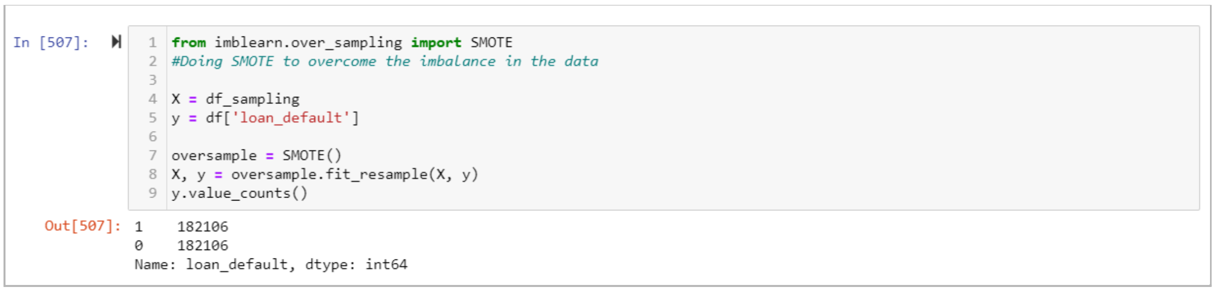
Class imbalance and its treatment

The classes in the target column were found to be in the ratio 80:20 as shown below



```
In [506]:  df['loan_default'].value_counts()

Out[506]:  0    182106
           1     50539
           Name: loan_default, dtype: int64
```

To overcome the bias in prediction towards class 0, over sampling was done with SMOTE. Ratios after SMOTE are as below

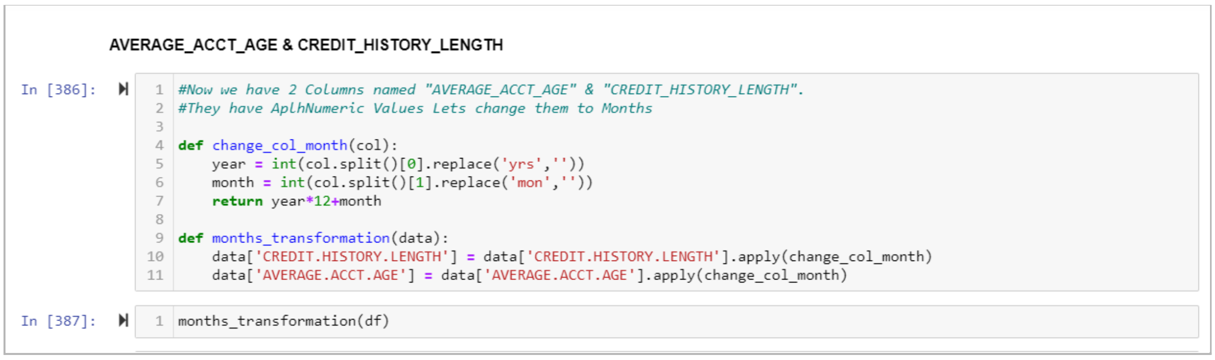


```
In [507]:  from imblearn.over_sampling import SMOTE
           #Doing SMOTE to overcome the imbalance in the data

           X = df_sampling
           y = df['loan_default']

           oversample = SMOTE()
           X, y = oversample.fit_resample(X, y)
           y.value_counts()

Out[507]:  1    182106
           0    182106
           Name: loan_default, dtype: int64
```

Feature Engineering

New features were formed with the help of available features in order to make the data more meaningful.

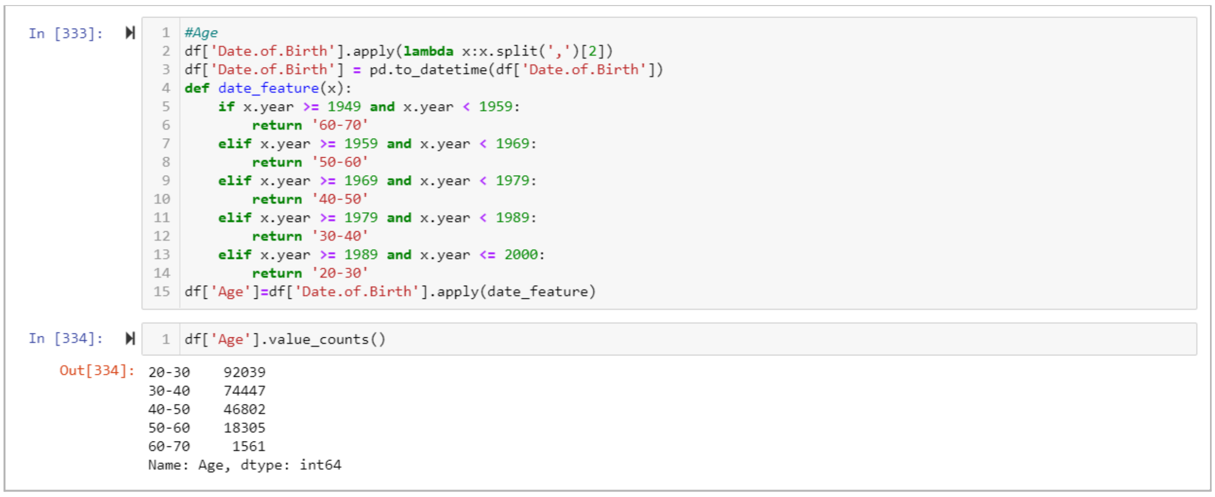
Average_acct_age and credit_history_length is converted to Number of Months



AVERAGE_ACCT_AGE & CREDIT_HISTORY_LENGTH

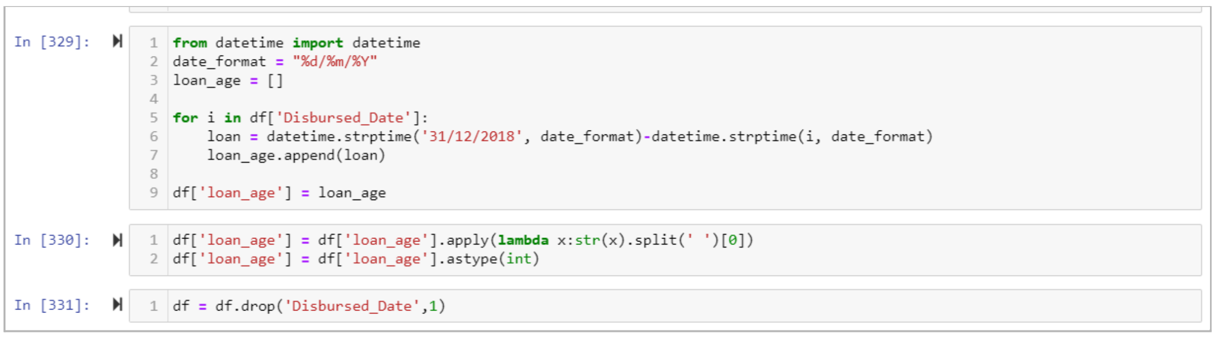
```
In [386]:  #Now we have 2 Columns named "AVERAGE_ACCT_AGE" & "CREDIT_HISTORY_LENGTH".
           #They have AplhNumeric Values Lets change them to Months

           def change_col_month(col):
               year = int(col.split()[0].replace('yrs',''))
               month = int(col.split()[1].replace('mon',''))
               return year*12+month

           def months_transformation(data):
               data['CREDIT.HISTORY.LENGTH'] = data['CREDIT.HISTORY.LENGTH'].apply(change_col_month)
               data['AVERAGE.ACCT.AGE'] = data['AVERAGE.ACCT.AGE'].apply(change_col_month)

In [387]:  months_transformation(df)
```

Date of Birth column is converted to Age with 5 levels as shown below



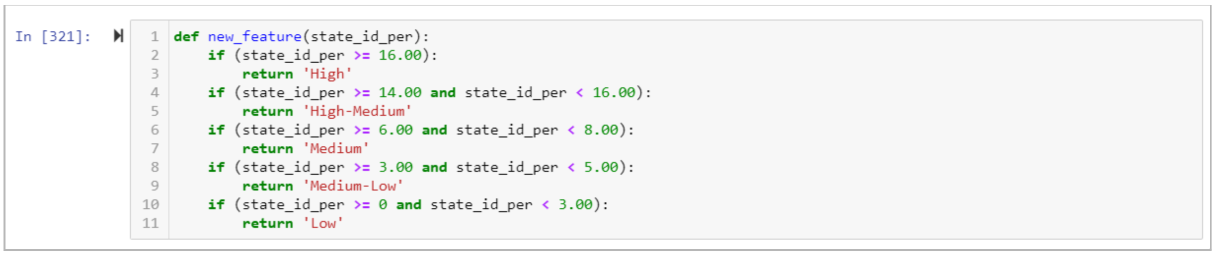
```
In [333]:
#Age
df['Date.of.Birth'].apply(lambda x:x.split(',')[2])
df['Date.of.Birth'] = pd.to_datetime(df['Date.of.Birth'])
def date_feature(x):
    if x.year >= 1949 and x.year < 1959:
        return '60-70'
    elif x.year >= 1959 and x.year < 1969:
        return '50-60'
    elif x.year >= 1969 and x.year < 1979:
        return '40-50'
    elif x.year >= 1979 and x.year < 1989:
        return '30-40'
    elif x.year >= 1989 and x.year <= 2000:
        return '20-30'
df['Age']=df['Date.of.Birth'].apply(date_feature)

In [334]:
df['Age'].value_counts()

Out[334]: 20-30    92039
30-40    74447
40-50    46802
50-60    18305
60-70     1561
Name: Age, dtype: int64
```

Keeping Dec 31 2018 as reference, loan age is calculated by subtracting Disbursed date and the reference date.

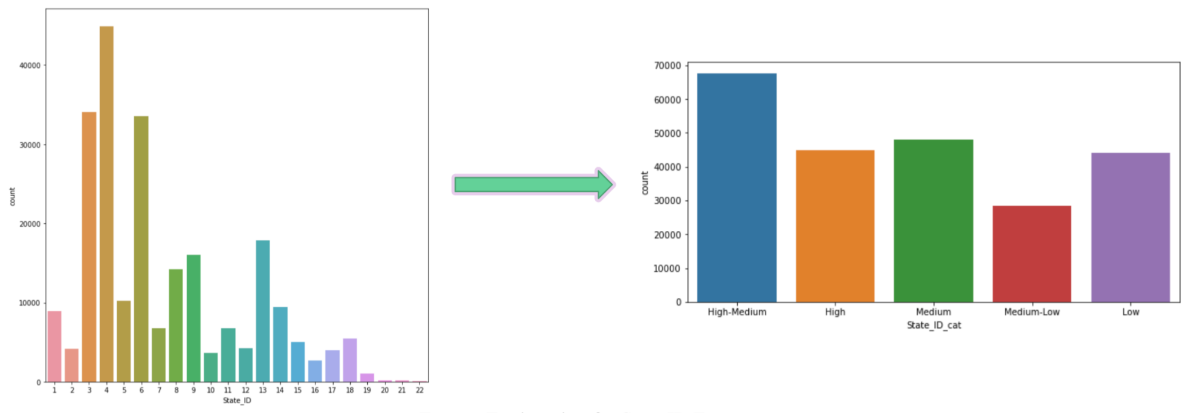


```
In [329]:
from datetime import datetime
date_format = "%d/%m/%Y"
loan_age = []

for i in df['Disbursed_Date']:
    loan = datetime.strptime('31/12/2018', date_format)-datetime.strptime(i, date_format)
    loan_age.append(loan)

df['loan_age'] = loan_age

In [330]:
df['loan_age'] = df['loan_age'].apply(lambda x:str(x).split(' ')[0])
df['loan_age'] = df['loan_age'].astype(int)

In [331]:
df = df.drop('Disbursed_Date',1)
```

State ID column is converted to 5 levels based on the total percentage of occurrence.

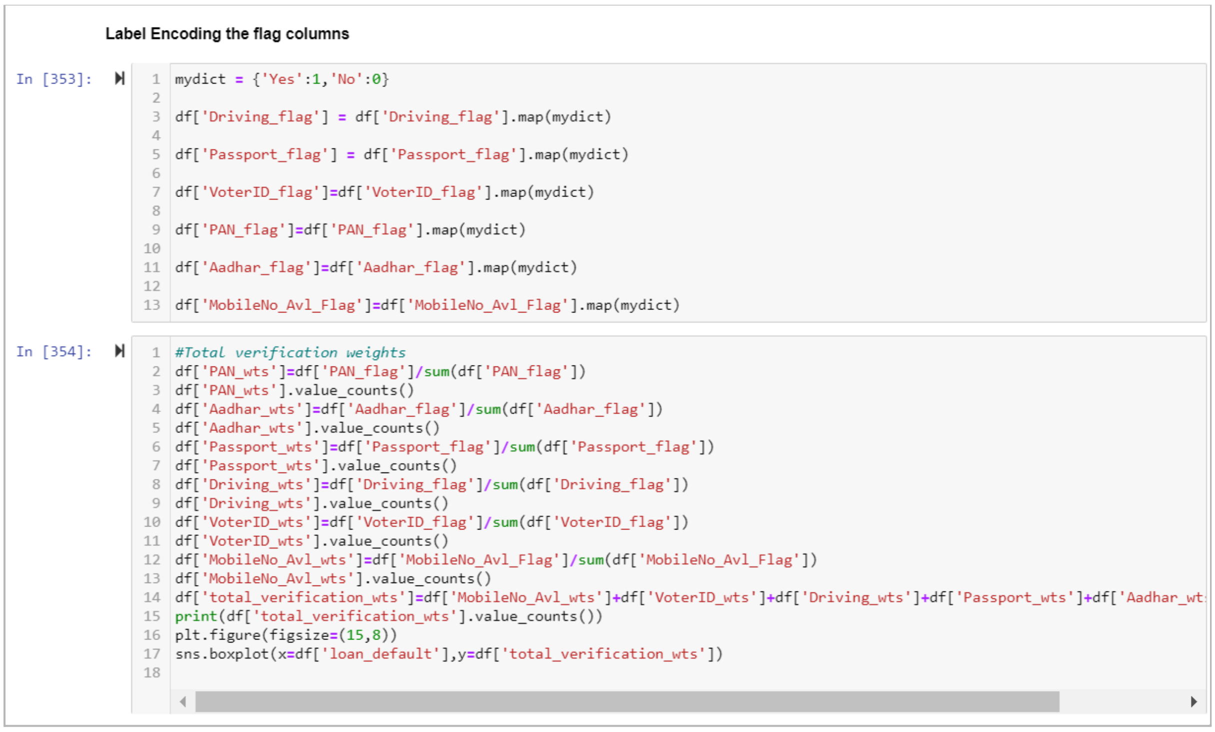


```
In [321]:
def new_feature(state_id_per):
    if (state_id_per >= 16.00):
        return 'High'
    if (state_id_per >= 14.00 and state_id_per < 16.00):
        return 'High-Medium'
    if (state_id_per >= 6.00 and state_id_per < 8.00):
        return 'Medium'
    if (state_id_per >= 3.00 and state_id_per < 5.00):
        return 'Medium-Low'
    if (state_id_per >= 0 and state_id_per < 3.00):
        return 'Low'
```



Feature Engineering for State ID Feature

All the flag columns are combined to form Total verifications that have a weighted average of all the positive flags put together as shown below.



Label Encoding the flag columns

In [353]:

```
mydict = {'Yes':1,'No':0}

df['Driving_flag'] = df['Driving_flag'].map(mydict)

df['Passport_flag'] = df['Passport_flag'].map(mydict)

df['VoterID_flag']=df['VoterID_flag'].map(mydict)

df['PAN_flag']=df['PAN_flag'].map(mydict)

df['Aadhar_flag']=df['Aadhar_flag'].map(mydict)

df['MobileNo_Avl_Flag']=df['MobileNo_Avl_Flag'].map(mydict)
```

In [354]:

```
#Total verification weights
df['PAN_wts']=df['PAN_flag']/sum(df['PAN_flag'])
df['PAN_wts'].value_counts()
df['Aadhar_wts']=df['Aadhar_flag']/sum(df['Aadhar_flag'])
df['Aadhar_wts'].value_counts()
df['Passport_wts']=df['Passport_flag']/sum(df['Passport_flag'])
df['Passport_wts'].value_counts()
df['Driving_wts']=df['Driving_flag']/sum(df['Driving_flag'])
df['Driving_wts'].value_counts()
df['VoterID_wts']=df['VoterID_flag']/sum(df['VoterID_flag'])
df['VoterID_wts'].value_counts()
df['MobileNo_Avl_wts']=df['MobileNo_Avl_Flag']/sum(df['MobileNo_Avl_Flag'])
df['MobileNo_Avl_wts'].value_counts()
df['total_verification_wts']=df['MobileNo_Avl_wts']+df['VoterID_wts']+df['Driving_wts']+df['Passport_wts']+df['Aadhar_wt
print(df['total_verification_wts'].value_counts())
plt.figure(figsize=(15,8))
sns.boxplot(x=df['loan_default'],y=df['total_verification_wts'])
```

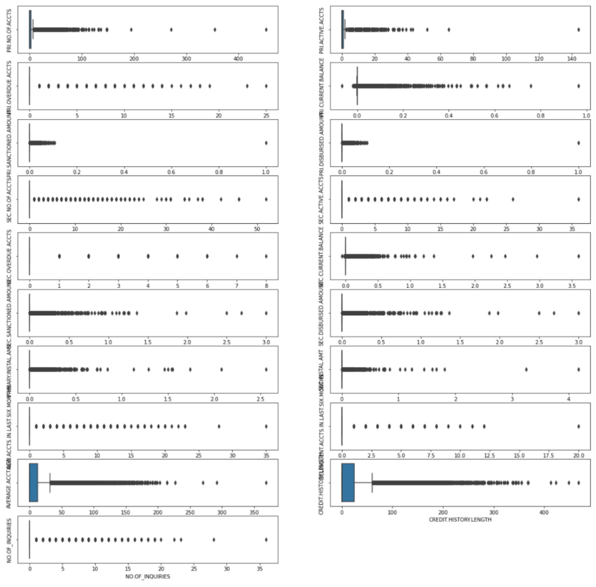
Transformations

For all the features, square root, log and box cox transformations were checked. Whichever reduced the skewness the most, that transformation was considered for the particular feature.

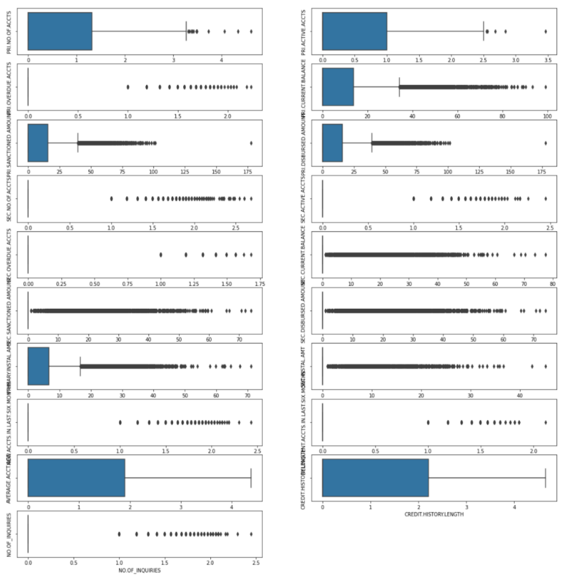
Square root transformation

- Perform_cns.Score, Perform_cns.Score.Description, Pri.No.Of.Accts Pri.Active.Accts, Pri.Overdue.Accts, Pri.Current.Balance, Pri.Sanctioned.Amount, Pri.Disbursed.Amount, Sec.No.Of.Accts, Sec.Active.Accts, Sec.Overdue.Accts, Sec.Current.Balance, Sec.Sanctioned.Amount, Sec.Disbursed.Amount, Primary.Instal.Amt, Sec.Instal.Amt, New.Accts.In.Last.Six.Months, Delinquent.Accts.In.Last.Six.Months, Average.Acct.Age, Credit.History.Length, No.Of_inquiries.

BEFORE SQRT TRANSFORM



AFTER SQRT TRANSFORM



Given the nature of the data even after outlier treatment there are still outliers present which must be considered as extreme values.

Box Cox transformation

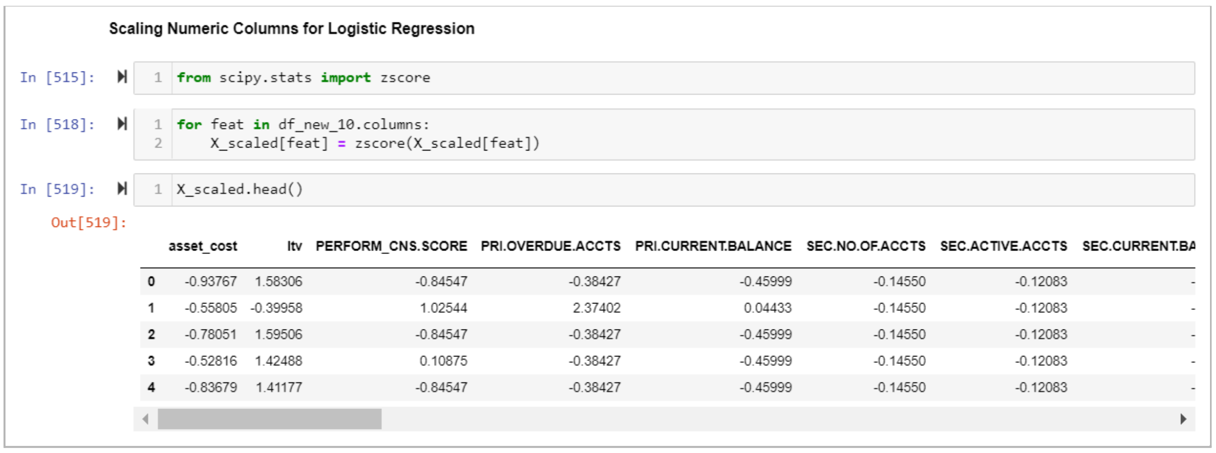
- LTV column

Log transformation

- Asset Cost column.

Scaling the data

Standard Scaling/Z Score is used to scale the data



Scaling Numeric Columns for Logistic Regression

```
In [515]:  from scipy.stats import zscore

In [518]:  for feat in df_new_10.columns:
               X_scaled[feat] = zscore(X_scaled[feat])

In [519]:  X_scaled.head()
Out[519]:
```

	asset_cost	ltv	PERFORM_CNS.SCORE	PRI.OVERDUE.ACCTS	PRI.CURRENT.BALANCE	SEC.NO.OF.ACCTS	SEC.ACTIVE.ACCTS	SEC.CURRENT.BA
0	-0.93767	1.58306	-0.84547	-0.38427	-0.45999	-0.14550	-0.12083	-
1	-0.55805	-0.39958	1.02544	2.37402	0.04433	-0.14550	-0.12083	-
2	-0.78051	1.59506	-0.84547	-0.38427	-0.45999	-0.14550	-0.12083	-
3	-0.52816	1.42488	0.10875	-0.38427	-0.45999	-0.14550	-0.12083	-
4	-0.83679	1.41177	-0.84547	-0.38427	-0.45999	-0.14550	-0.12083	-

Feature selection

In the EDA, we found the below features insignificant and removed it accordingly.

- Employee_code_ID
- UniqueID
- Manufacturer_id
- Supplier_id
- Current_pincode_ID
- Branch_id

Applied statistical tests for feature selection, T-Test for Continuous Variable and removed the insignificant column.

- Sec.overdue.accts

Also VIF was applied and the below insignificant variables were dropped.

- Pri.Disbursed.Amount
- Sec.Disbursed.Amount
- Disbursed_Amount
- Credit.History.Length
- Pri.Sanctioned.Amount
- Sec.Sanctioned.Amount
- Pri.Active.Accts
- Loan_Age
- Pri.No.Of.Accts

- Modelling
  - Since it is a classification model, we tried Logistic Regression, Decision Tree, Random Forest, AdaBoost, Gradient Boost, XGBoost, Naive Bayes. We also took an Ensemble of all the base models. We found that XGBoost had a good test accuracy and recall score, we considered this our model for further tuning.

- Evaluation
  - We have considered Accuracy and Recall as our Evaluation Metrics. The model which has high accuracy and high recall is chosen as the best model, which is XGBoost in our case. We are looking for high recall as our aim is to lower the False Negative.
- Deployment – Future scope.

MODEL EVALUATION

MODEL SCORES BEFORE TUNING

	Logistic Regression	**Decision Tree**	**Random Forest**	**Ada Boost**	**Gradient Boost**	**XG Boost**	**Naive Bayes**	**Ensemble**

Train Accuracy(%)	76	73	76	77	79	83	53	79
Test Accuracy(%)	76	73	76	77	79	82	53	79
Train Recall(%)	71	61	71	73	72	71	93	70
Test Recall(%)	71	61	71	73	72	71	93	70

Final Model

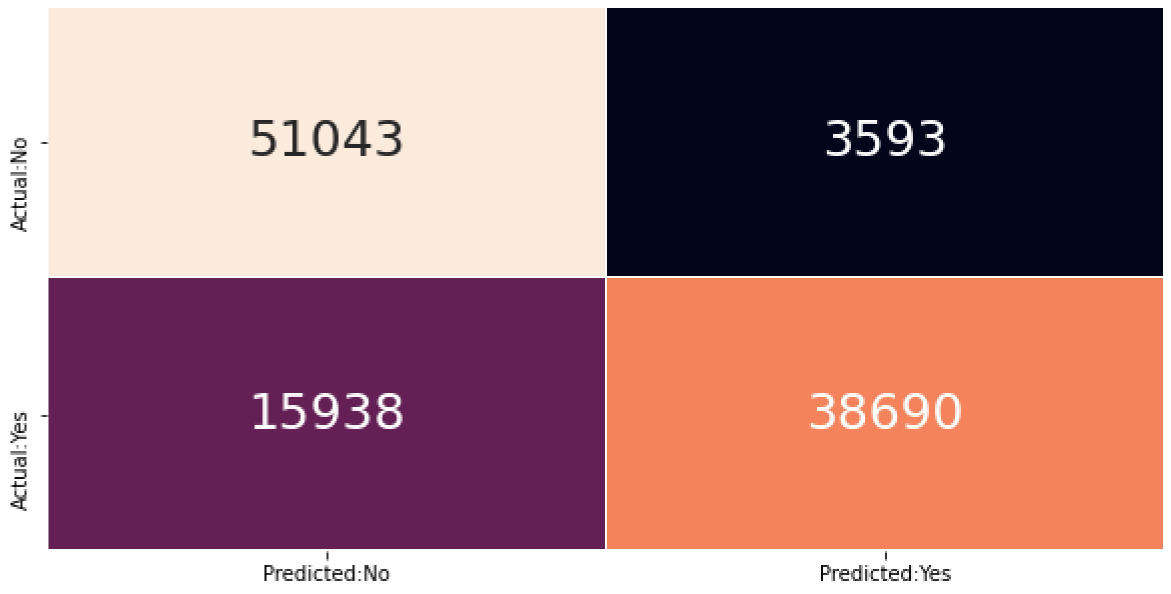
Since the XG Boost model has a good test accuracy and recall score, we considered this our model for further tuning.

- XG Boost is an Ensemble Technique.It uses a boosting mechanism.
- In boosting, the trees are built sequentially such that each subsequent tree aims to reduce the errors of the previous tree. Each tree learns from its predecessors and updates the residual errors. Hence, the tree that grows next in the sequence will learn from an updated version of the residuals.
- We tried different hyperparameters
    - Learning_rate
    - Max_depth
    - Gamma
    - N_estimators
- Our main objective here is to improve the Accuracy and Recall.
- On Tuning, we inferred that the parameter Learning_rate is prominent, as Accuracy and Recall both increased by 1%.
- We inferred from the tuning of above parameters that there is a slight improvement in overall Test and Train Accuracy and Test and Train Recall.
- When the tuned model was compared with the base model, we decided to keep the tuned model as it was regularized and also the model improved slightly.
- The model evaluation and its success is mainly focused on the Recall Score. The Type-2 error here is more critical. This is because a customer getting predicted as a non-default payer when the customer actually defaults, incurs losses to the financial organisation.
- The Accuracy of 82% and Recall of 72% suggests that the model build is quite robust for new sets of data to be predicted accurately.

COMPARISON OF BENCHMARK SCORES VS FINAL SCORES

As mentioned above, max depth = 6, n estimators = 100, gamma = 0, learning rate = 0.3 are the 4 main hyper parameters considered for tuning. The benchmark scores of the XGboost model keeping these constant are given below.

Base Model Confusion Matrix:



Base Model Test Classification Report:

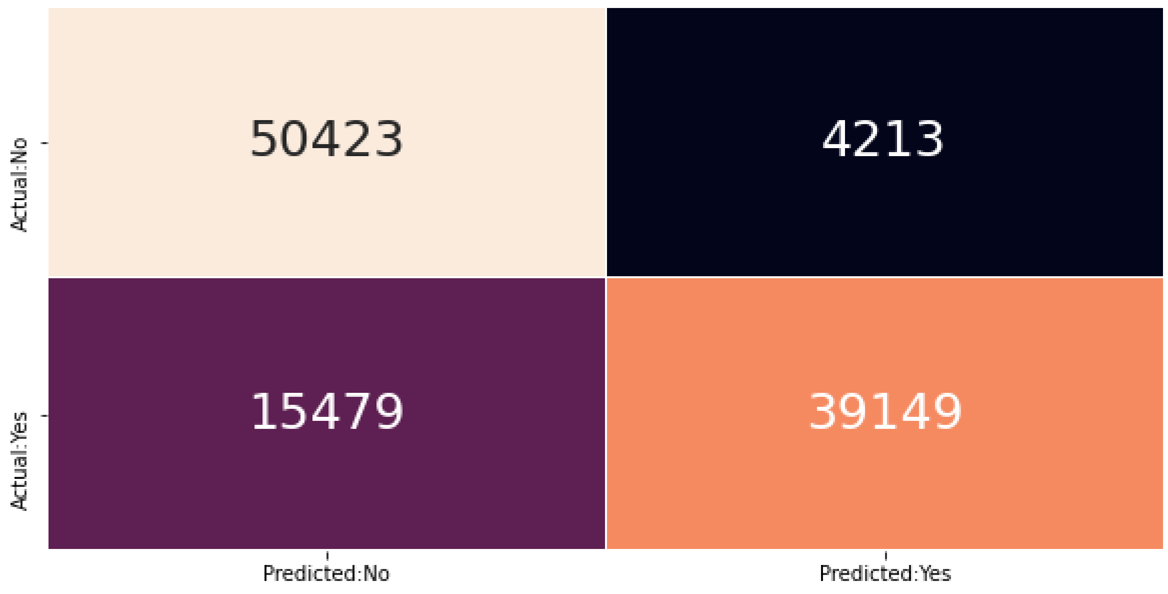
```
              precision    recall  f1-score   support

           0       0.76      0.93      0.84     54636
           1       0.92      0.71      0.80     54628

    accuracy                           0.82    109264
   macro avg       0.84      0.82      0.82    109264
weighted avg       0.84      0.82      0.82    109264
```

Base Model Train Classification Report:

```
              precision    recall  f1-score   support

           0       0.77      0.94      0.85    127470
           1       0.92      0.71      0.81    127478

    accuracy                           0.83    254948
   macro avg       0.85      0.83      0.83    254948
weighted avg       0.85      0.83      0.83    254948
```

After performing multiple iterations of parameter tuning with multiple combinations and evaluations, learning rate with 0.6 gave the best score while the rest of the combinations resulted in overfitting. The final scores are given below.

Final Model Confusion Matrix:



Final Model Test Classification Report:

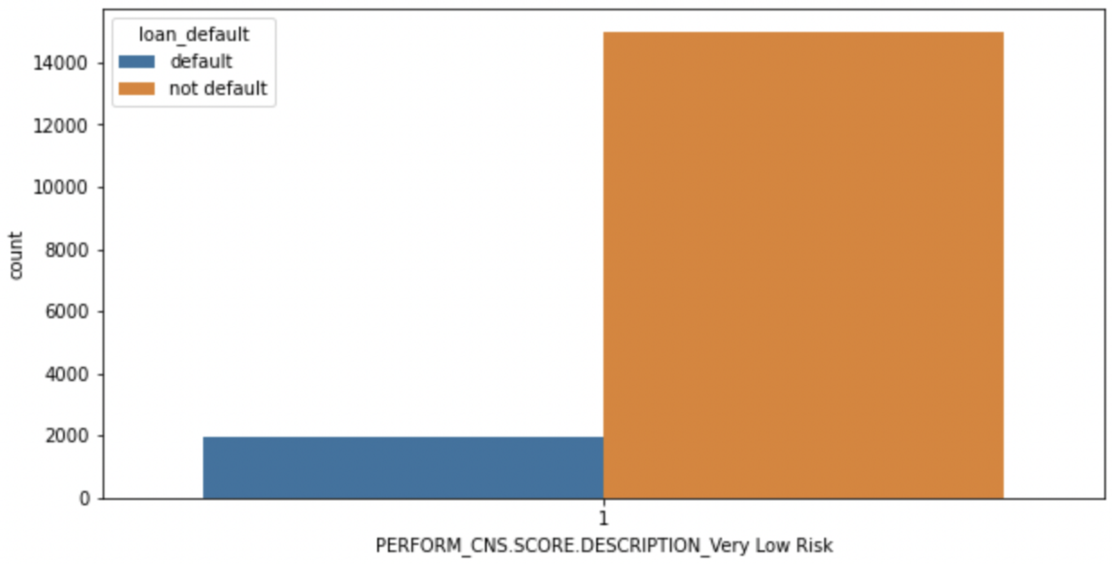
```
              precision    recall  f1-score   support

           0       0.77      0.92      0.84     54636
           1       0.90      0.72      0.80     54628

    accuracy                           0.82    109264
   macro avg       0.83      0.82      0.82    109264
weighted avg       0.83      0.82      0.82    109264
```

Final Model Train Classification Report:

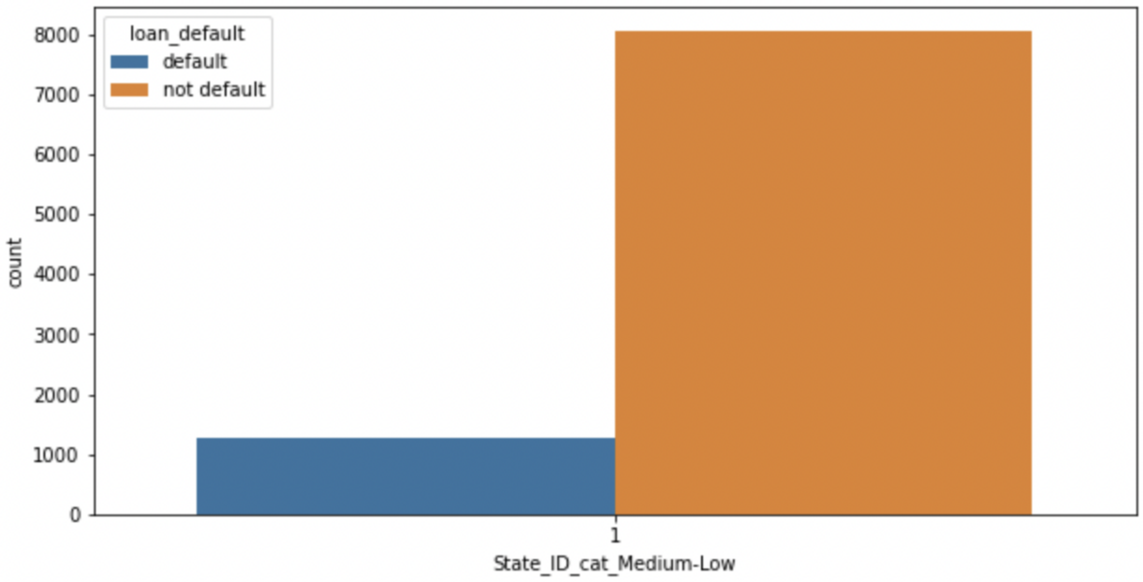
```
              precision    recall  f1-score   support

           0       0.78      0.94      0.85    127470
           1       0.92      0.73      0.82    127478

    accuracy                           0.83    254948
   macro avg       0.85      0.83      0.83    254948
weighted avg       0.85      0.83      0.83    254948
```

There is 1% improvement of recall on the benchmark scores for both train and test data by increasing the learning rate from 0.3 to 0.6.

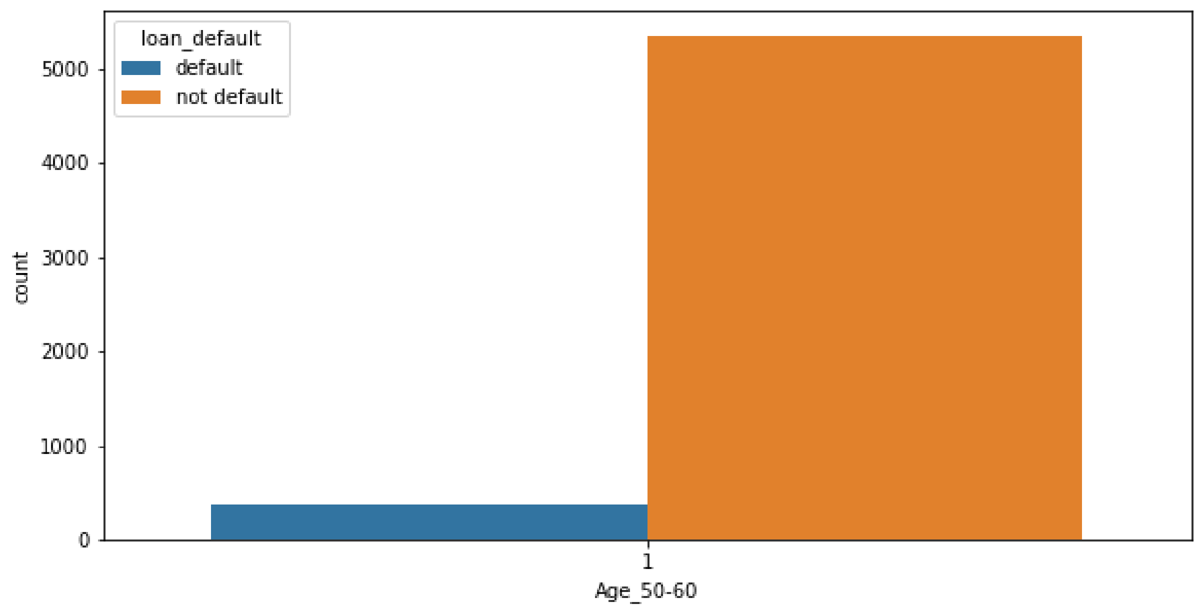
INSIGHTS FROM MODEL - VISUALISATION



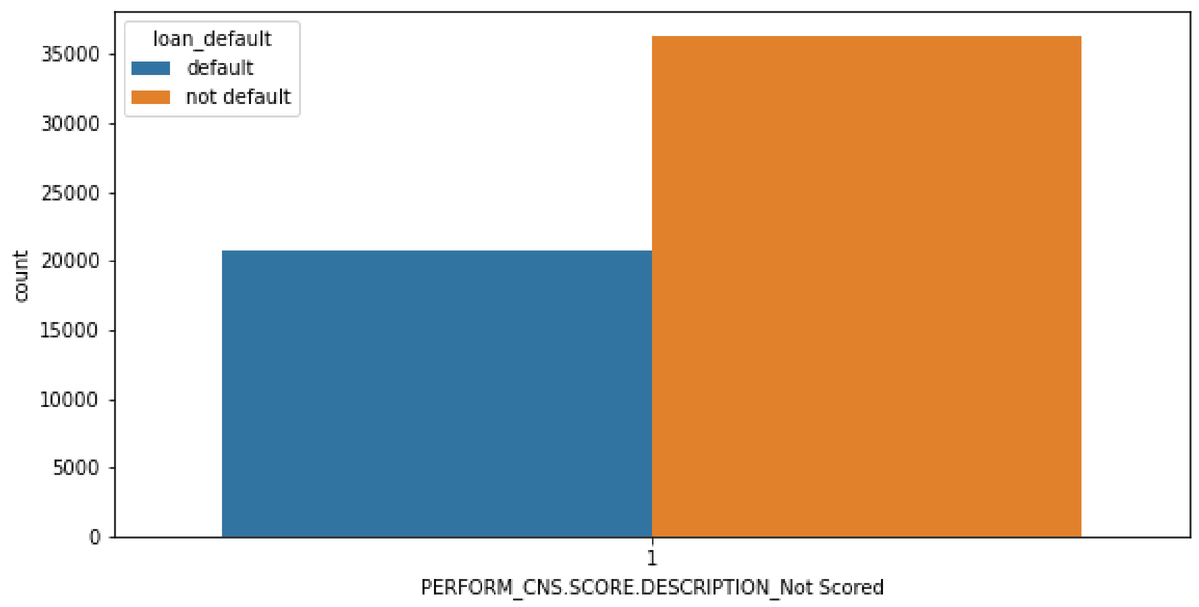
People with very low risk in cns score have low chances of loan defaulting



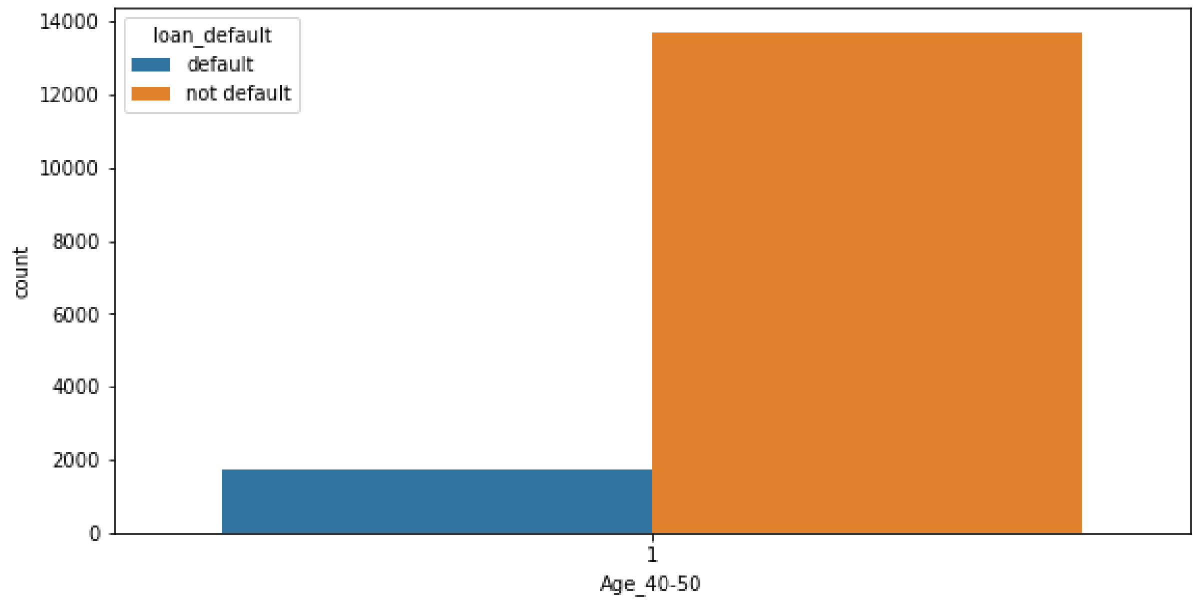
People who belong to med-low state ID category i.e, states with state ID as 1,14 and 5 have low chances of loan defaulting



People belonging to 50-60 years category have low chances of loan defaulting



People with no cns score have low chances of loan defaulting

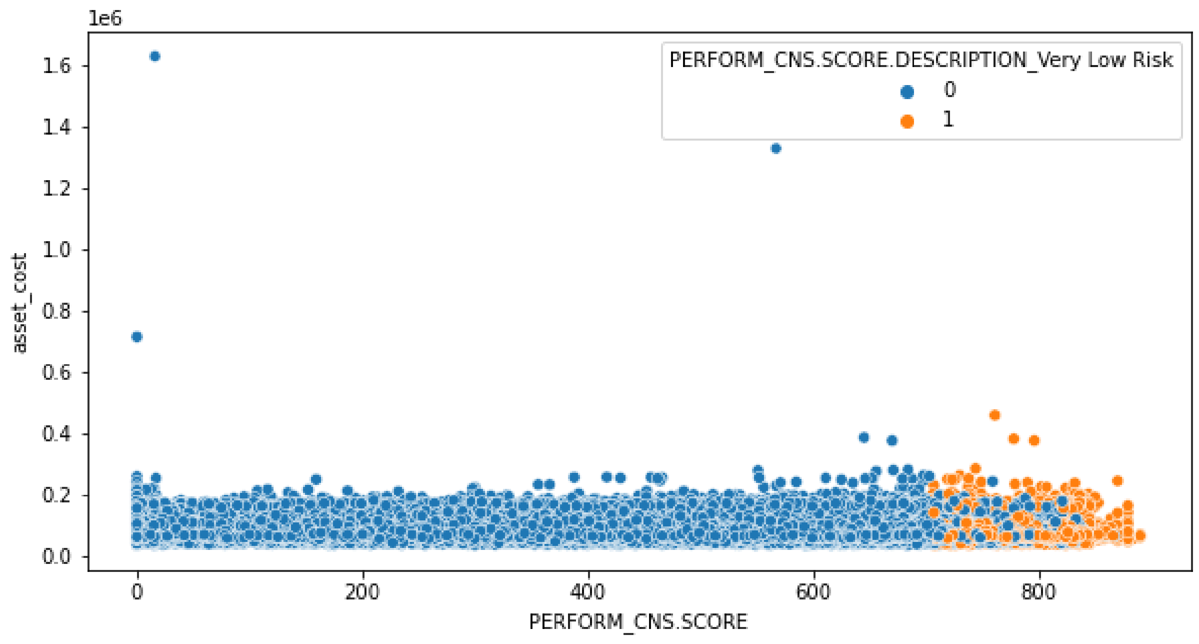


People belonging to 40-50 years category have low chances of loan defaulting

RECOMMENDATIONS AND STRATEGIES

The financial Institutions should focus on the people who belong to the following categories -

1. Loanees having Very Low Risk as CNS Score Description, i.e, having a good CNS score of 706-890 and loanees who do not have a CNS Score.
2. Loanees belonging to 40-60 Age Category.
3. People belonging to States with State IDs 1, 5 and 14.



Regardless of what the asset cost is, as long as loanees have a CNS Score ranging between 700-900, their chances of loan defaulting will be less. (The orange scatter plots in the above plot)

Our solution will help to reduce the False Negative which will minimise the risk of giving out loans to individuals. i.e by identifying individuals who can default going ahead but are considered as non-defaulters due to some parameters in their financial history.

LIMITATIONS AND ENHANCEMENTS FOR MODEL

- Limitations-
    - The recall of the final model is 0.72 which means there is still a 0.28 chance of predicting False Negatives, i.e., loan defaulters who are predicted as non-defaulters.
    - Sensitive data - Complete information on the Loaness cannot be disclosed due to the nature of the data, because of which there can be false predictions made
    - Outlier presence - Loan requirements differ from person to person, therefore predicting for all of the loanees together can lead to misclassifications.
    - Real world problem - Getting complete information from the loanees and being able to verify it completely is not easily achievable. Also having a better CNS score is subjected to everyone having to get a CNS report, which might not be possible in reality.
- Enhancing Solutions -
    - Currently 52% of the loanees fall into the CNS Not Scored Category which comprises of all of the following cases
        - No Bureau History Available
        - More than 50 active Accounts found
        - No Activity seen on the customer (Inactive)
        - No Updates available in last 36 months
        - Not Enough Info available on the customer
        - Only a Guarantor.
        - Sufficient History Not Available.
    - As CNS Not scored is an important feature, providing updates on the loanees or gathering more information on them, might lead to better prediction. So financial institutions should start laying more importance on CNS reports and complete information submission, which can reduce the problems faced in this aspect.

LEARNINGS AND REFLECTIONS

- Effective Model Building:
    - Achieving best scores through multiple models and iterating over multiple combinations of hyperparameters.
- Domain Knowledge:
    - Learnt the parameters required to understand the loan requirement and how lending business is an important aspect of any banking institute.
- Attention to detail:
    - We learnt the importance of paying attention to the details and how a small mistake can make a huge difference to the overall model scores.