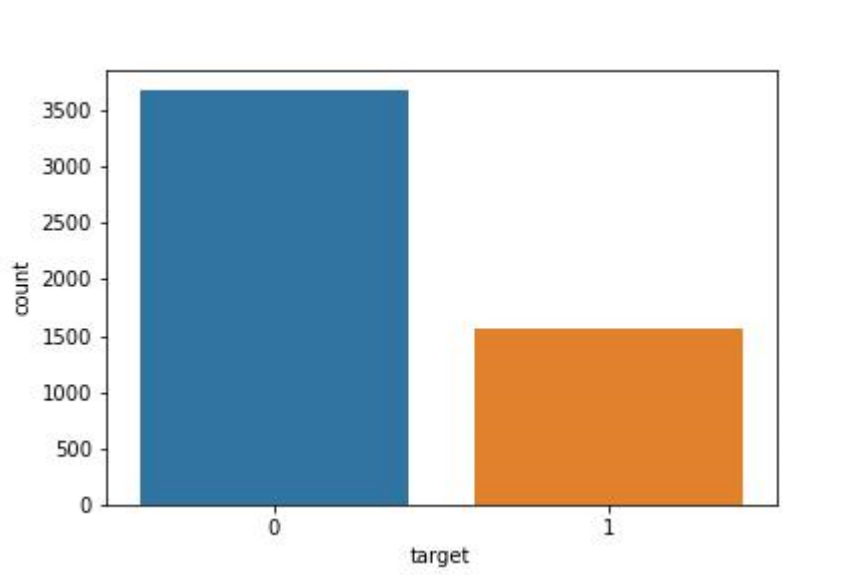
PREDICTION ANALYSIS FOR LOAN DELINQUENCY

By

FULE CHI BEMIEH

QUESTION

a). Did the Loan Roll?
Null hypothesis: The Loan did not roll

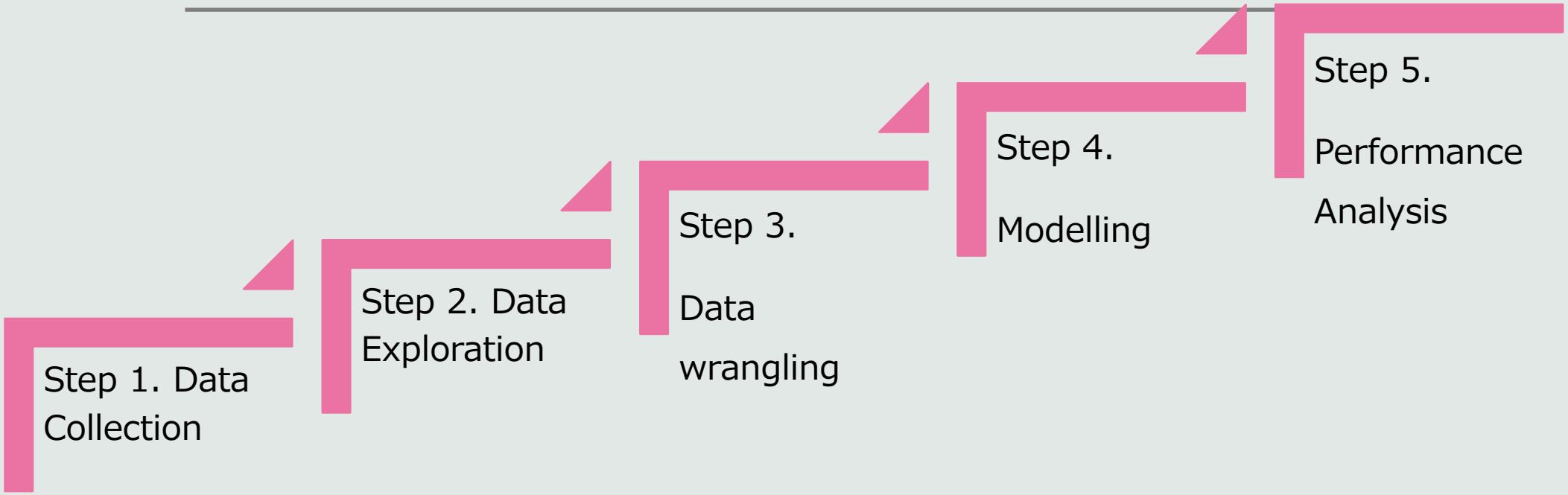


b). What was the Accuracy level?
Accuracy = 75%

```
from sklearn.ensemble import RandomForestClassifier,ExtraTreesClassifier
model = RandomForestClassifier()
classify(model, x, y)
```

```
Accuracy is 74.94287890327494
Cross Validation is 79.75238095238095
```

The gymnastics behind the scene



Step 1. Data collection

- Methodology: Machine learning using python – Jupyter Notebook
- Import libraries into Jupyter Notebook
- Load data (csv file)

Step 1. Data Collection

```
import pandas as pd
import numpy as np
import seaborn as sns
import sklearn.linear_model as lr
import sklearn.model_selection as train_test_split
from matplotlib import pyplot as plt
import matplotlib
```

```
df = pd.read_csv('dataset_risk_analytics.csv')
```

Step 2. Data exploration

11 columns or entities and 5783 entries

Data type: 5 INT, 4 Float and 2 Object

Step2. Data Exploration

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5783 entries, 0 to 5782
Data columns (total 11 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   loan_id                   5783 non-null   int64
 1   monthly_income            5406 non-null   float64
 2   origination_score_band    5783 non-null   int64
 3   TOB_months                5406 non-null   float64
 4   closing_principal_balance 5406 non-null   float64
 5   original_loan_amount      5406 non-null   float64
 6   product                   5783 non-null   object
 7   original_loan_term        5783 non-null   int64
 8   remaining_loan_term       5783 non-null   int64
 9   delq_history              5406 non-null   object
 10  target                    5783 non-null   int64
dtypes: float64(4), int64(5), object(2)
memory usage: 497.1+ KB
```

```
df.describe()
```

11]:

	loan_id	monthly_income	origination_score_band	TOB_months	closing_principal_balance	original_loan_amount	original_loan_term	remaining_l
count	5783.000000	5406.000000	5783.000000	5406.000000	5406.000000	5406.000000	5783.000000	57
mean	3625.042711	14314.372919	3.904029	28.385683	5714.391417	6630.701073	62.023517	
std	1516.292643	13799.514988	1.490877	16.357494	7837.483534	8491.265603	23.774170	
min	1000.000000	750.000000	1.000000	6.000000	100.000000	100.000000	12.000000	
25%	2310.500000	6000.000000	3.000000	17.000000	1380.000000	1800.000000	60.000000	
50%	3623.000000	9750.000000	4.000000	24.000000	3000.000000	3750.000000	60.000000	
75%	4934.500000	17250.000000	5.000000	38.000000	6800.000000	8100.000000	60.000000	
max	6249.000000	99750.000000	8.000000	83.000000	88200.000000	90000.000000	192.000000	17

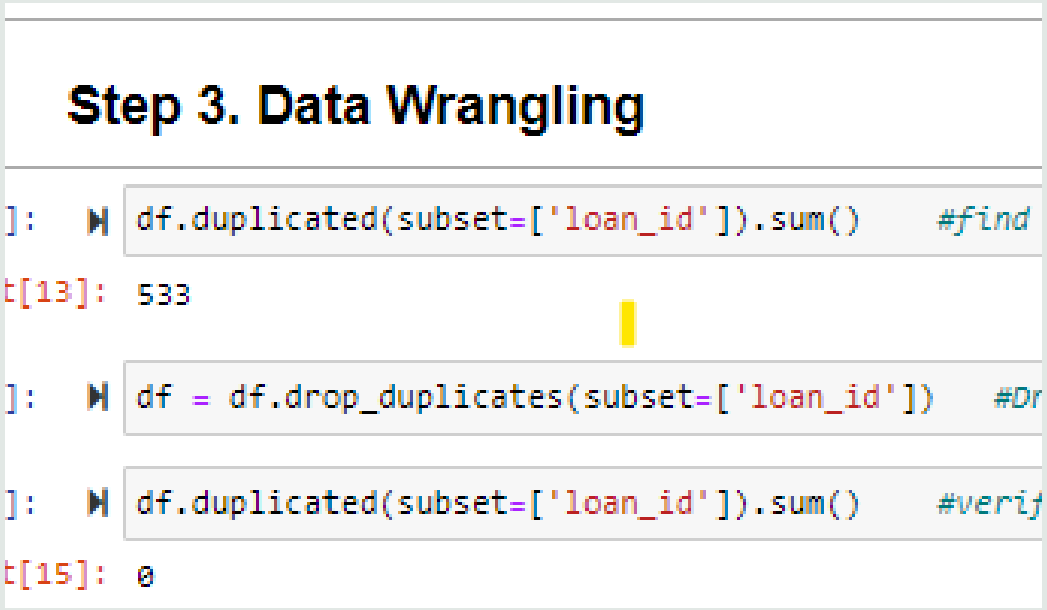
```
df
```

12]:

	loan_id	monthly_income	origination_score_band	TOB_months	closing_principal_balance	original_loan_amount	product	original_loan_term	remaini
0	1000	6000.0	5	83.0	300.0	1800.0	B	24	
1	1001	39000.0	5	82.0	7200.0	11700.0	B	60	
2	1002	18000.0	5	78.0	2700.0	5400.0	B	60	
3	1003	23250.0	3	76.0	3900.0	6900.0	B	60	
4	1004	12000.0	3	74.0	2100.0	3600.0	B	60	
...	...	...	...	...	...	...	...	...	

Step 3. Data wrangling - Duplicates

Duplicates: 533 duplicates identified and deleted



Step 3. Data wrangling – Nulls

- Null Values: 377 null values identified in 5 entities
- Null values in numeric datatype was replaced with the mean of each column
- Null values in object datatype was replaced with the mode of the entries of the column

```
df.isnull().sum()
```

```
loan_id                      0
monthly_income             377
origination_score_band       0
TOB_months                 377
closing_principal_balance  377
original_loan_amount       377
product                      0
original_loan_term           0
remaining_loan_term          0
delq_history               377
target                       0
dtype: int64
```

```
data = df.fillna({
    'monthly_income' : df['monthly_income'].mean(),
    'TOB_months' : df['TOB_months'].mean(),
    'closing_principal_balance' : df['closing_principal_balance'].mean(),
    'original_loan_amount' : df['original_loan_amount'].mean(),
    'delq_history' : df['delq_history'].mode()[0],
})

data.isnull().sum()        #Replacing null values with the mean(for int/float
```

```
loan_id                    0
monthly_income             0
origination_score_band     0
TOB_months                 0
closing_principal_balance  0
original_loan_amount       0
product                    0
original_loan_term         0
remaining_loan_term        0
delq_history               0
target                     0
dtype: int64
```

Step 4. Modelling

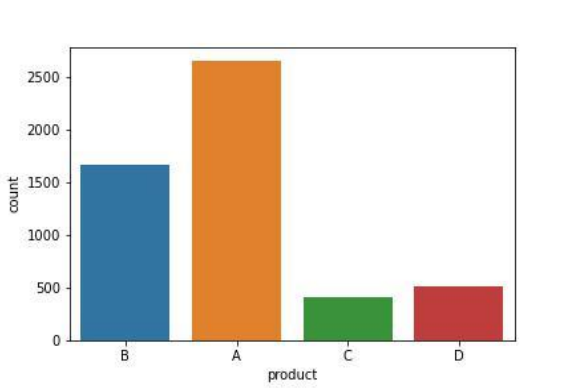
A.DATA EXPLORATION

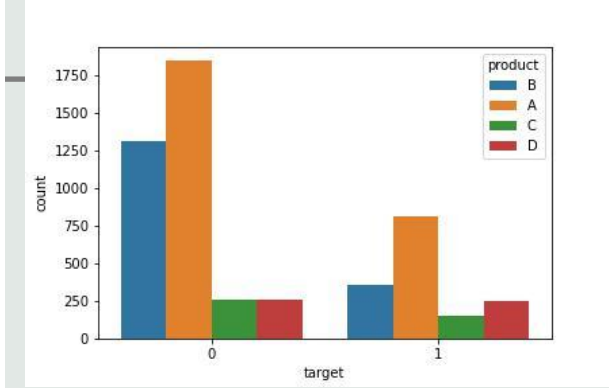
- Graphic observation of the data using bar charts for categorical data and histograms for continous data,
- Observe possible relationships between variables
- Use log function to normalize the continous data
- Transform columns with object data type to float data type

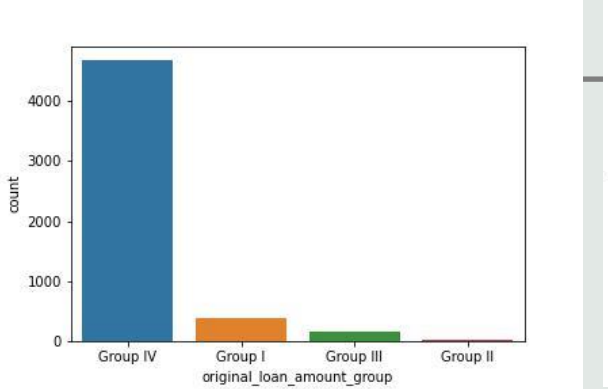
B. CORRELATION ANALYSIS

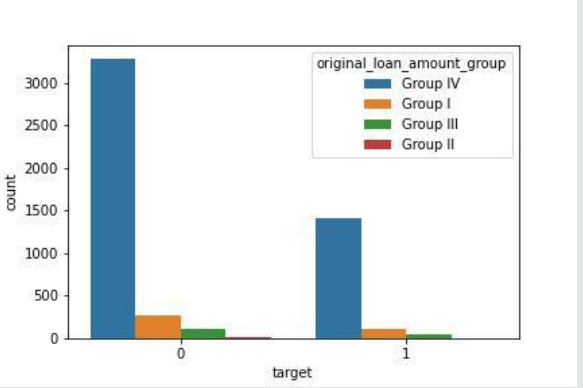
C. TRAIN MODEL

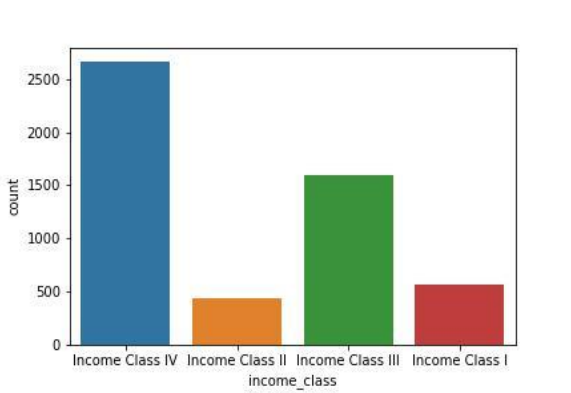
Step 4.
Modelling – Exploratory data analysis

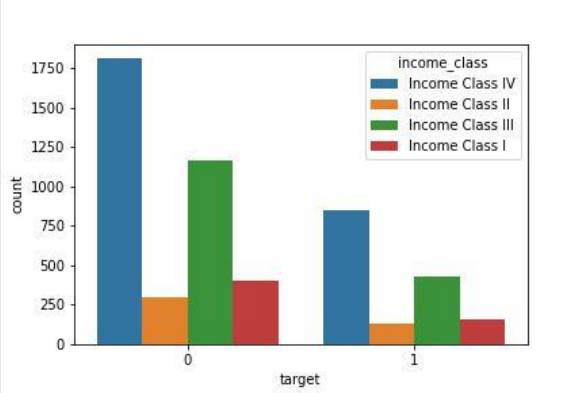


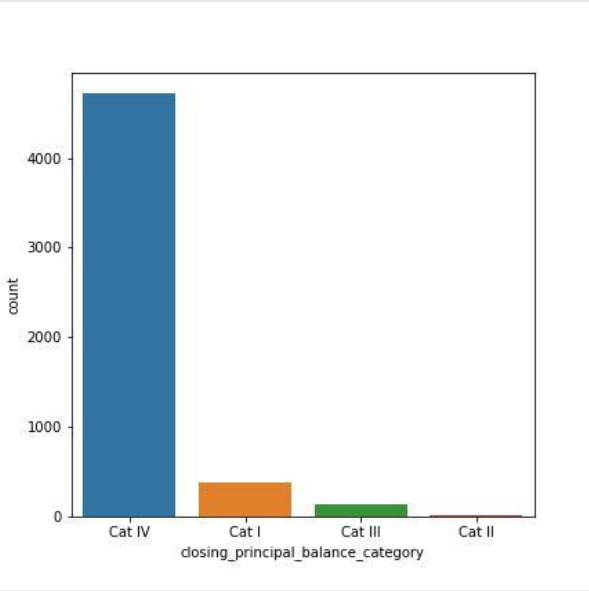


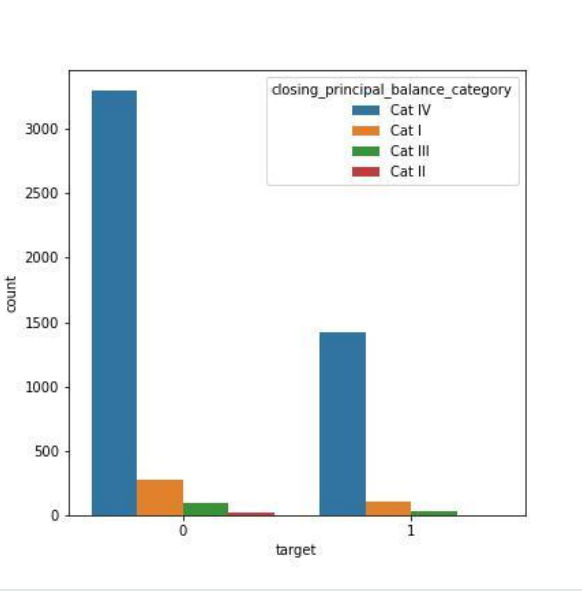




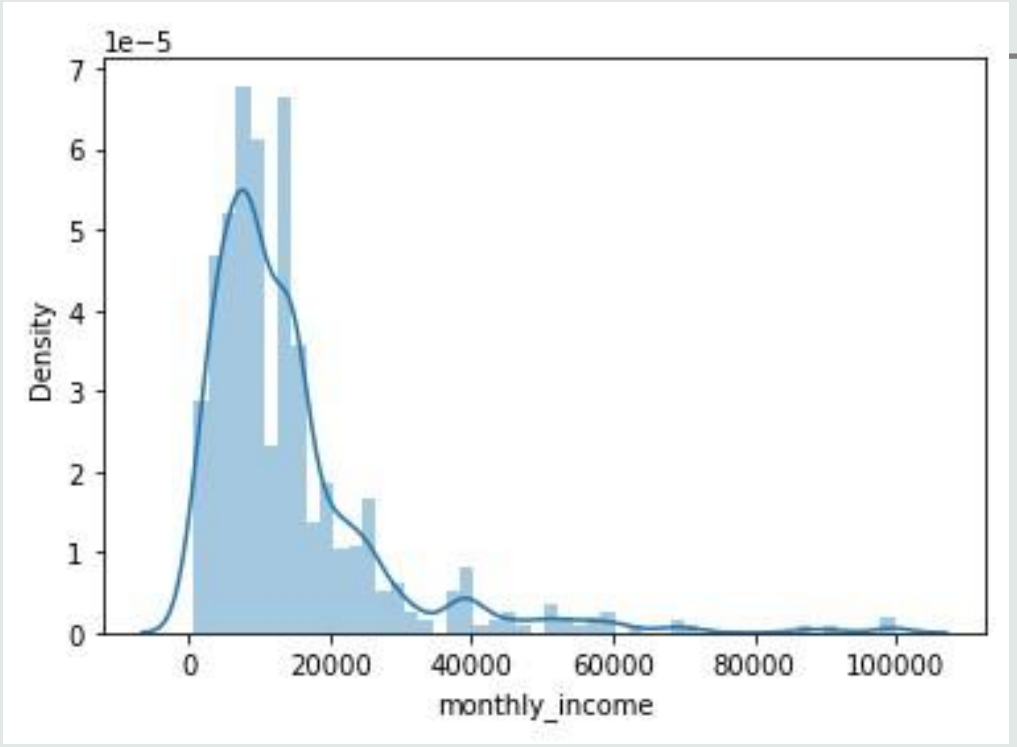


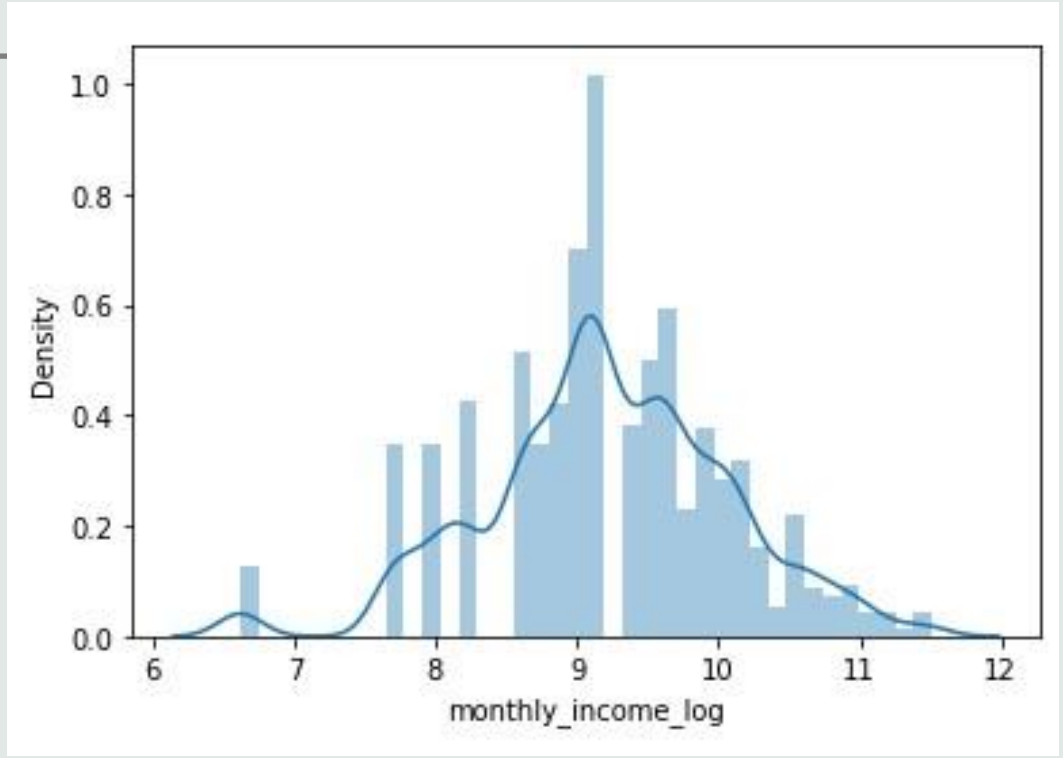






Step 4.
Modelling – Exploratory data analysis





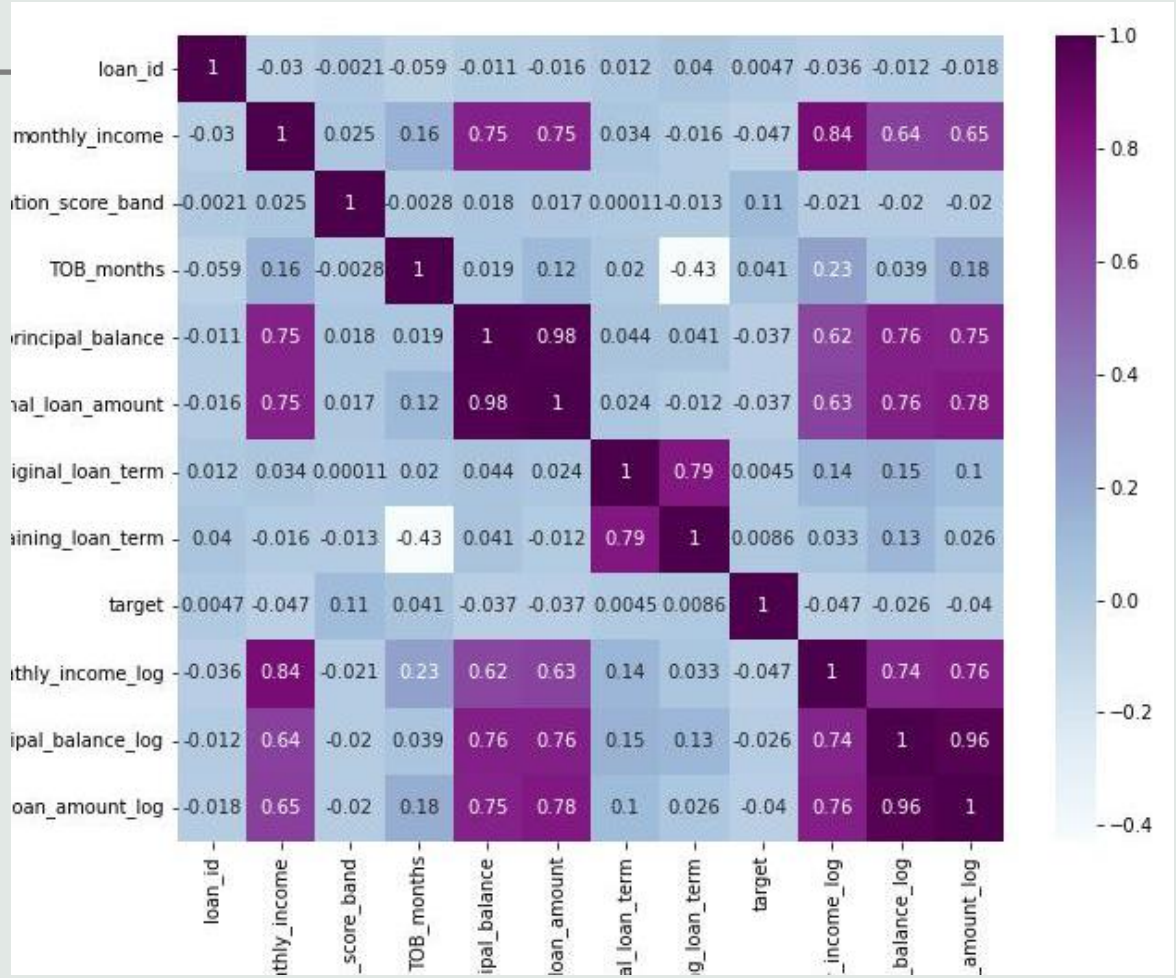
STEP 4.
MODELLING – CORRELATION ANALYSIS

A. DATA EXPLORATION

B. CORRELATION ANALYSIS

- Find correlation between variables
- Select only variables with high correlation to build the model
- 6 variables were dropped
- 3 independent variables were retained: monthly_income_log, principal_balance_log, initial_loan_amount.

C. TRAIN MODEL



STEP 4.

MODELLING – TRAIN MODEL

A. DATA EXPLORATION

B. CORRELATION ANALYSIS

C. TRAIN MODEL

- Define independent and dependent variables
- Split dataset into train and test datasets (75:25)
- Use Logistic Regression to train the model
- Use Decision Tree Classifer & Random Forest Classifier to improve model

```
#Train-Test Split
##specifying input and output attributes

x = dataset.drop(columns=['target'], axis=1)
y = dataset['target']
```

```
#Split Data

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.25, random_state=42)
```

```
#Train Model

from sklearn.model_selection import cross_val_score
def classify(model, x, y):
    x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.25, random_state=42)
    model.fit(x_train, y_train)
    print("Accuracy is", model.score(x_test, y_test)*100)
    #Cross validation is used for better validating the model
    #Eg. CV=5, train=4, test=1
    score = cross_val_score(model, x, y, cv=5)
    print("Cross Validation is", np.mean(score)*100)
```

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
classify(model, x, y)
```

```
Accuracy is 70.6016755521706
Cross Validation is 70.28571428571428
```

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
classify(model, x, y)
```

```
Accuracy is 77.45620715917745
Cross Validation is 78.9904761904762
```

STEP 5.
PERFORMANCE ANALYSIS – CONFUSION MATRIX

		ACTUAL	
		TRUE	FALSE
PREDICTED	Target = 0	NOT ROLLED	Type I error
	Target = 1	Type II error	ROLLED

MINIMIZE ERRORS:

- Delete Null values instead of substituting with means,
- Add new variables eg. Credit score, family income, family size, etc.

```
model = RandomForestClassifier()
model.fit(x_train, y_train)
```

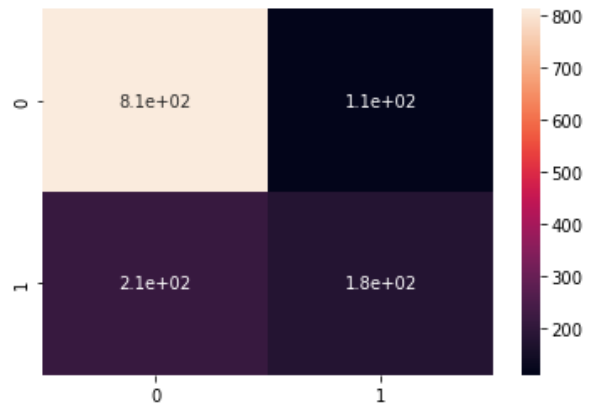
```
RandomForestClassifier()
```

```
from sklearn.metrics import confusion_matrix
y_pred = model.predict(x_test)
cm = confusion_matrix(y_test, y_pred)
cm
```

```
array([[813, 110],
       [212, 178]], dtype=int64)
```

```
sns.heatmap(cm, annot=True)
```

```
<AxesSubplot:>
```



Thanks for your keen attention!