PREDICTING POSSIBLE DEFAULTERS FOR A CONSUMER LOAN PRODUCT

<u>2021-22</u>

UNIVERSITY OF HYDERABAD

AUTHORED BY: AAYUSH AJMERA

Table of Contents

CHI-SQUARE TEST
- Hypothesis Testing....25

DATA 6

DATA ANALYSIS
- CATEGORICAL VARIABLES 17
- CORRELATION MATRIX 16
- INCOME DISTRIBUTION IN TOP DEFAULTING CATEGORIES....19
- PAIRPLOT....15

DATA PREPROCESSING 27

EDA....12
- CHI-SQUARE TEST 24
- DATA ANALYSIS 15
- DATA CLEANING....12
- DATA DESCRIPTION....13
- GRAPHS AND BOX PLOTS 13
- OBSERVATIONS....26

INTRODUCTION 3

LIBRARIES USED FOR DATA ANALYSIS....11

LITERATURE REVIEW....4

METHODOLOGY....9

MODELLING....28
- BASELINE MODEL....29
- HANDLING IMBALANCED DATA....31
- HYPER-PARAMETER TUNING....33

PERFORMANCE METRICS 7

REFERENCES 36

“Banks get in trouble for only one reason. They make bad loans.”

-Carl Webb

INTRODUCTION

Individuals all across the world rely on banks to lend them money for a variety of reasons. Assist them overcome financial limits, to achieve personal goals or to expand their personal ventures. The action of getting a loan has become unavoidable due to the ever-changing economy and ever-increasing competitiveness in the financial industry. Despite the fact that giving loans is beneficial to both parties, the activity is fraught with danger. These hazards apply to a borrower's inability to repay a loan by the agreed-upon due date, which was agreed upon by both the lender and the borrower, and is referred to as 'Credit Risk.'.

As a result, credit risk is one of the most critical hazards that a financial organization must manage. Because there is no profit without loan repayment, the issue of credit risk management affects all financial organizations that lends money to individuals and legal entities. Before issuing a loan, it is critical to assess the clients' credit eligibility. To evaluate a borrower in the traditional loan procedure, banking authorities primarily use the '5C concept,' i.e., Character, Capital, Capacity, Collateral, and Conditions. This assessment was based primarily on personal experience and expertise of banking officials. This approach however, has a lot of drawbacks. Although banks and large financial businesses grant loan requests after a regressive screening and validation process, there is no guarantee that the applicant chosen at the end of the process will be able to repay the loan on time.

A numerical score termed 'Credit Score' was used to determine a candidate's worthiness for loan acceptance or rejection. In general, a credit score assists authorities in calculating the likelihood of a borrower repaying a loan within the specified time frame based on their credit history or payment history as well as their past. Credit scoring in finance sector needed the use of experts and statistical algorithms to accurately predict an applicant's creditworthiness.

One of the first applications of Data Science was in the financial sector. Every year, financial organizations face bad debts and losses. However, they first undertake paper work before granting a loan, which results in a large amount of data being collected. This information can then be utilized to undertake a variety of analyses to determine the likelihood of risk and anticipate borrower status.

The prediction of borrower status i.e. in future borrower will be defaulter or non-defaulter is a binary classification problem. Researchers and banking authorities have recently chosen to train classifiers based on various algorithms to automatically predict an applicant's credit score based on their credit history and other historical data, making the process of selecting eligible candidates a lot easier before the loan is approved.

LITERATURE REVIEW

To start off with the literature review, let's start off with more general systematic literature reviews that focus on the application of machine learning in the general field of Banking. Iedunote in its blog (1) provide a complete 360 degree view on banking loans and how to ideally distinguish between an ideal loan and a problem loan. They emphasize on how periodic stress testing of the loan portfolio can help bankers access the potential risks better.

Moody's in their paper (2) have outlined how an integrated system with better technology can enhance a bank's risk management capabilities and thus reduce loan losses. Monitoring can now be conducted based on assessed loan-level risk rather than against inflexible portfolio policies. Monitoring can be simplified and streamlined, with borrowers experiencing credit deterioration receiving the most attention. The paper was optimistic in advancement of new tools such as machine learning and automated monitoring, thereby helping lenders derive meaningful insights into borrower behavior, credit events, and probability of default. Michael Strumpf, in his blog (3) on Analytics India Mag describes the current situation in the aftermath of 2008 financial crisis. The authors lists the advantages and disadvantages of various early warning system in loan default prediction and emphasized on how machine learning and stochastic models can help decision makers predict defaults fairly easily.

Anchal Goyal , Ranpreet Kaur in their paper (4) for IJARCCE constructs eleven machine learning models with nine properties that are used to predict the credit risk of customers who have applied for loan. They further construct an ensemble model through the existing models and check the accuracy on several parameters like Gini Coefficient, AUC, and ROC to do the comparison. There model resulted in Decision Tree performing the best followed by Random Forrest and SVM.

In their paper (5), Dr. A. Chitra and S. Uma proposed an ensemble learning method for time series prediction based on Radial Basis Function networks (RBF), K - Nearest Neighbor (KNN), and Self-Organizing Maps (SOM). They proposed the PAPEM model, which outperforms individual models. The study (6) by Sivasree M S and Rekha Sunny T provided a loan credibility prediction system that aids enterprises in making the best decision on whether to approve or deny consumer loan requests.

Jianwu Li, Haizhou Wei, and Wangli Hao in their study (7), used a weighted-selected attribute bagging method to conduct research on leveraging customer qualities to estimate credit risk. They tested their results using two credit databases and found that it performed exceptionally well in terms of prediction accuracy and stability when compared to other state-of-the-art systems.

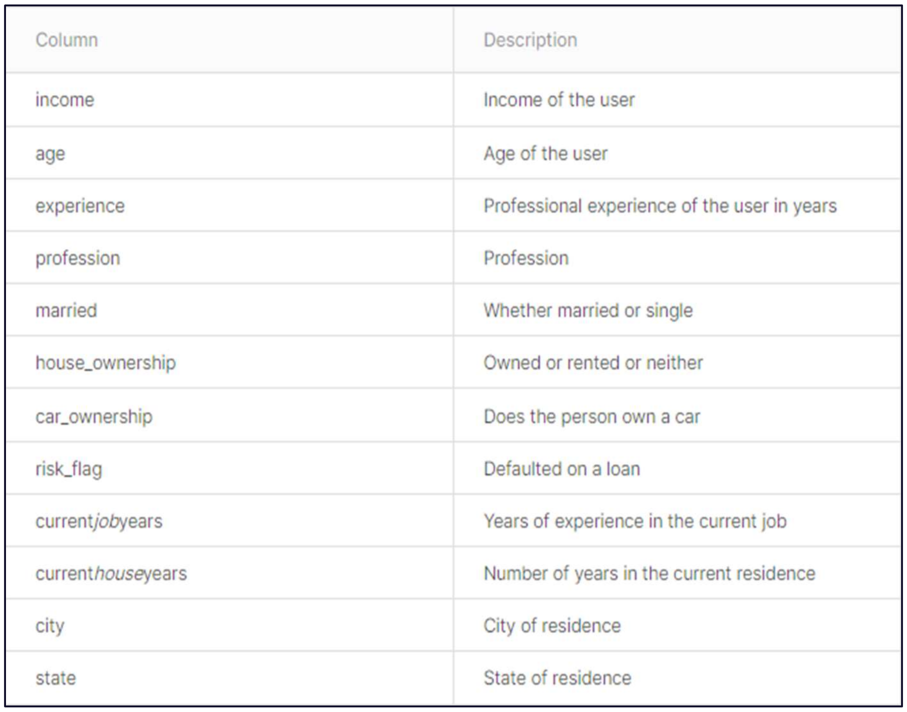
Paper published (8) in IOP Conference used random forest and decision tree methods with an accuracy of 80% and 73% respectively. The researchers advised caution to the data providers to practice restrain for borrowers taking loans who don't own houses as they are more likely to default. Whereas, Joseph Breeden in his study (9) reviewed several approaches, including logistic regression, k-nearest neighbors, random forest, neural networks, support vector machines, stochastic gradient boosting, Naive Bayes, and others, and it was decided that declaring one optimum method for all was nearly impossible.

Lin Zhu in paper [10] and Nazeeh Ghatasheh in his research [11] also used the Random Forest Algorithm to build a model for loan default prediction. Random forest has a substantially higher accuracy (98%) than other algorithms like logistic regression (73%), decision trees (95%), and support vector machines, according to paper [10]. (75 %). The random forest method is one of the best solutions for credit risk prediction, according to the results of the research [11].

In her paper (12) Ashlesha Vaidya used logistic regression as a probabilistic and predictive approach to loan approval prediction. Artificial neural networks and logistic regression are the most commonly employed for loan prediction, according to the author, because they are easy to create and deliver the most accurate predictive analysis. Other algorithms are often weak at forecasting from non-normalized data, which is one of the reasons for this. However, because the independent variables on which the prediction is made do not have to be regularly distributed, the nonlinear impact and power factors are simply handled by Logistic regression.

DATA

The data was provided by Univ.AI through an organization that wants to predict who possible defaulters are for the consumer loans product. The data is about historic customer behavior based on what they have observed. Hence when they acquire new customers they want to predict who is riskier and who is not.



Column	Description
income	Income of the user
age	Age of the user
experience	Professional experience of the user in years
profession	Profession
married	Whether married or single
house_ownership	Owned or rented or neither
car_ownership	Does the person own a car
risk_flag	Defaulted on a loan
current*job*years	Years of experience in the current job
current*house*years	Number of years in the current residence
city	City of residence
state	State of residence

Data

The dataset provides the following information:

- There are 252,000 data points in the training dataset and 28000 in test dataset.
- The entire dataset contains 13 different features containing 7 numerical and 6 categorical columns.
- There is high cardinality in features STATE, Profession and CITY wherein the no. of unique values are 28, 51 and 316 respectively.
- Risk flag is the target variable classifying whether an applicant will be able to pay off the loan or default.

PERFORMANCE METRICS

Models are analysed for performance of prediction. From a bank's point of view, there are 4 possible outcomes during the process of evaluating a loan:

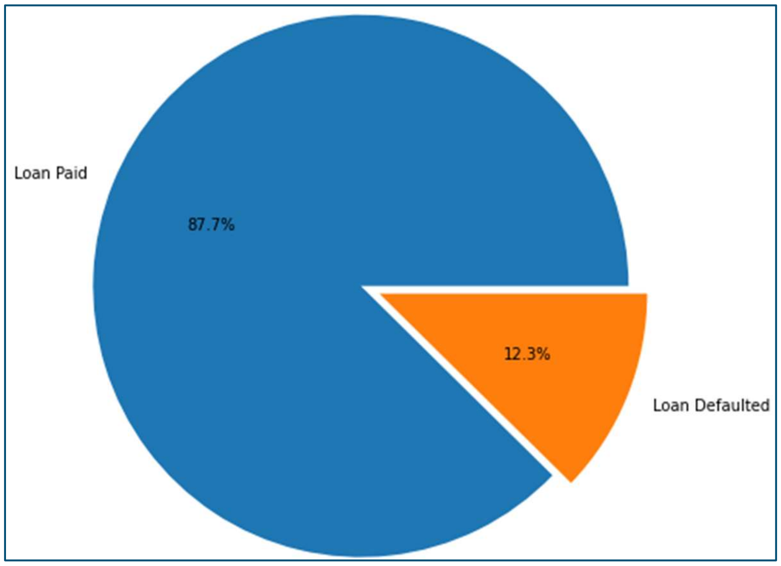
1. Bank to decide to give loan, and the borrower in actual could have repaid it in time. (True Positive)
2. Bank decided to reject loan, and the borrower in actual could not have repaid it back. (True Negative)
3. Bank decides to reject loan, and the borrower in actual could have repaid it back. Bank in this case will lose out on possible business. Ideally banks would not like to miss such potential opportunities but the loss here will be limited to the amount of interest the bank could have generated after giving the loan. (False Positive)
4. Banks decides to give loan, and the borrower in actual cannot repay it back. Banks in this case will incur loss. This is the worst possible outcome. (False Negative)

Every business involved in the process of lending money would ideally like to minimise their False Positive and at the very least avoid False Negatives since it leads to depletion of capital leading to creation of bad debt. Dealing and accounting for bad debts is another lengthy process which only leads to further cost to a business. Hence, for the purpose of this research, the models built will be evaluated on the following metrics:

F-BETA: is calculated as the harmonic mean of precision and recall that adds a configuration parameter called beta. A default beta value is 1.0, which is the same as the F-measure. A smaller beta value, such as 0.5, gives more weight to precision and less to recall, whereas a larger beta value, such as 2.0, gives less weight to precision and more weight to recall in the calculation of the score. In order to reduce the False Negative Errors, we have in our model given more importance to recall.

$$F_\beta = (1+\beta^2) \times \frac{\mathrm{precision} \times \mathrm{recall}}{(\beta^2 \times \mathrm{precision}) + \mathrm{recall}}$$
$$= \frac{(1+\beta^2)\mathrm{tp}}{(1+\beta^2)\mathrm{tp} + \beta^2\mathrm{fn} + \mathrm{fp}}$$

ROC SCORE: The Receiver Operator Characteristic (ROC) curve is a binary classification issue evaluation metric. It's a probability curve that displays the TPR against the FPR at different threshold levels, thereby separating the ‘signal' from the 'noise.' The Area under the Curve (AUC) is a summary of the ROC curve that measures a classifier's ability to distinguish between classes. The AUC indicates how well the model distinguishes between positive and negative classes. The greater the AUC, the better.

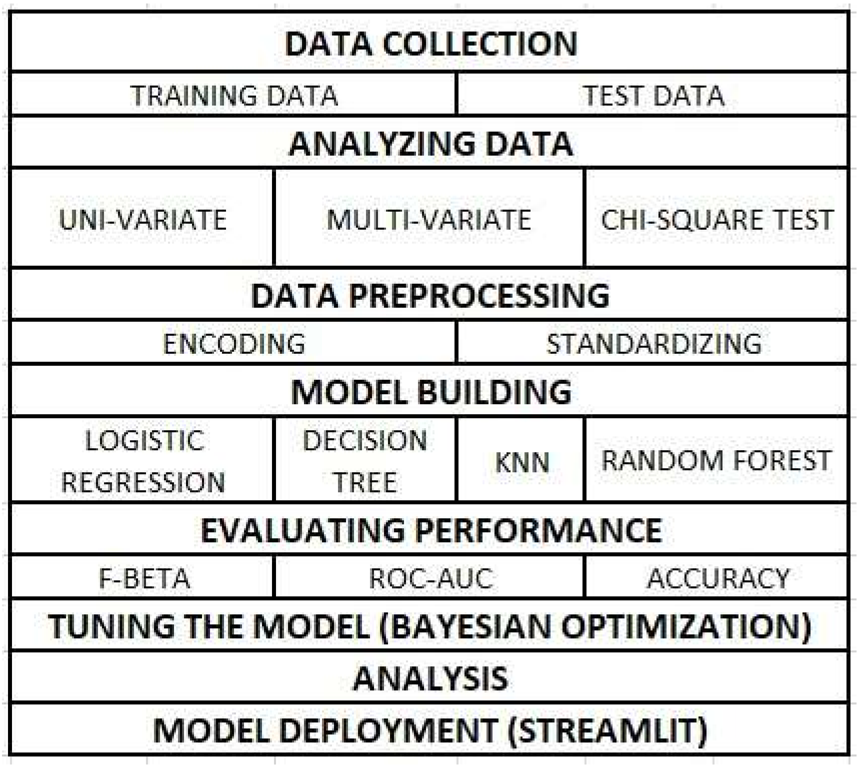


Distribution of Targeted Data

For the purpose of this study, several other models like Accuracy, Precision, Recall, Brier Score etc. were deliberated upon/. However, none of the model could be a good match for the data since the data is highly imbalanced in favour of non-defaulters.

- Accuracy for this reason will fail since the data is highly imbalanced.
- Precision and Recall were two other good options to consider as metrics for model evaluation. However, they consider only specific part of confusion metrics and not the whole of it.
- Since in model such as this, preventing false negatives and False Positives are of higher priority for a banker, F scores are much better evaluation.

METHODOLOGY



Four machine learning models have been used for the prediction of loan approvals. Below are the description of the models used:

LOGISTIC REGRESSION

given an independent variable, this is a classification procedure that utilises a logistic function to predict a binary outcome (True/False, 0/1, Yes/No). The goal of this model is to discover a link between variables and the likelihood of a specific result.

DECISION TREES

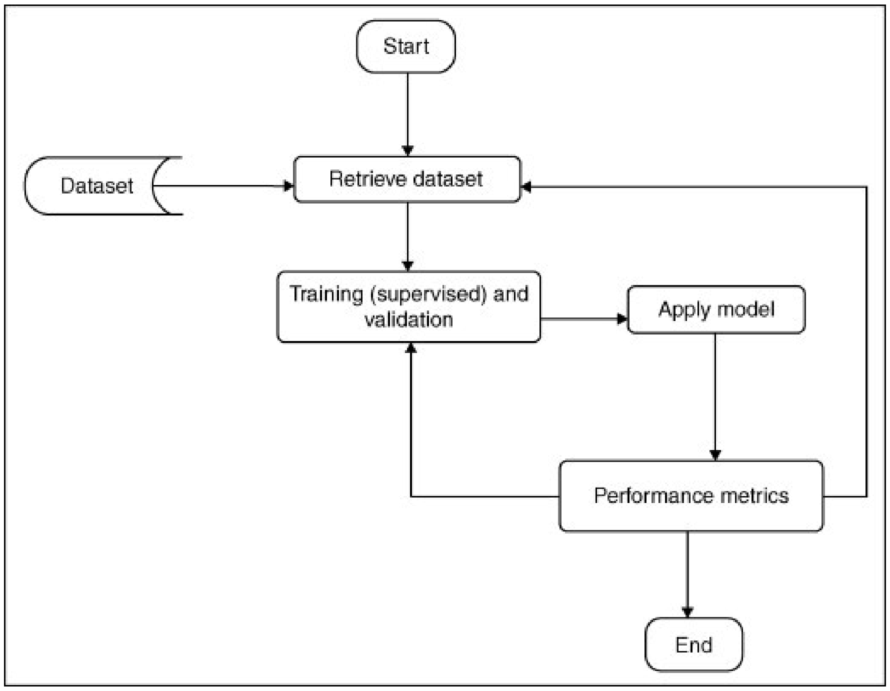
In this model, all features should be discretized such that the population may be divided into two or more homogenous groups or subsets. This model divides a node into two or more sub-nodes using a separate algorithm. The homogeneity and purity of the nodes grows in relation to the dependent variable as additional sub-nodes are created.

RANDOM FORREST

this is a tree-based ensemble model that aids in enhancing the model's accuracy. It builds a powerful forecasting model by combining a huge number of Decision trees. The median of all forecasters or the mean of all predictors is the final prediction class.

KNN

supervised learning domain and finds intense application in pattern recognition, data mining and intrusion detection. It is widely disposable in real-life scenarios since it is non-parametric, meaning, it does not make any underlying assumptions about the distribution of data



LIBRARIES USED FOR DATA ANALYSIS

The models are implemented using Python 3.7 with listed libraries:

PANDAS

Pandas is a Python package to work with structured and time series data. The data from various file formats such as csv, json, sql etc can be imported using Pandas. It is a powerful open source tool used for data analysis and data manipulation operations such as data cleaning, merging, selecting as well wrangling.

NUMPY

Numpy offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more. It aims to provide an array object that is up to 50x faster than traditional Python lists.

SEABORN & MATPLOTLIB

are a python library for building graphs to visualize data. It provides integration with pandas. This open source tool helps in defining the data by mapping the data on the informative and interactive plots. Each element of the plots gives meaningful information about the data.

SKLEARN

This python library is helpful for building machine learning and statistical models such as clustering, classification, regression etc. Though it can be used for reading, manipulating and summarizing the data as well, better libraries are there to perform these functions.

EXPLORATORY DATA ANALYSIS (EDA)

Refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary and statistics and graphical representations.

The EDA process is divided into two parts:

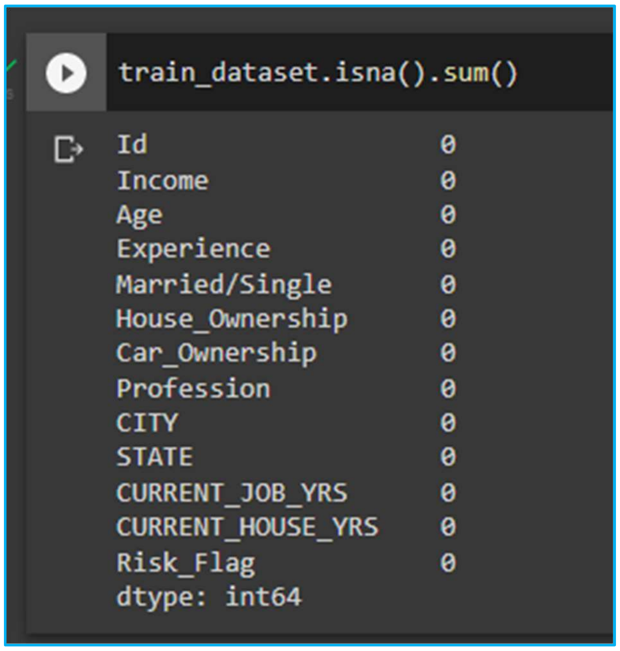
- Data Cleaning
- Data Analysis

DATA CLEANING:

During the data cleaning process, the data was checked for missing values, outliers, spaces, special characters etc.

Data initially was checked on Power BI to have a broader analysis. It was noted that some data points in features like Profession, CITY and STATE contained special characters like [] and numbers from 0-9. These characters were removed during the initial data analysis in Power BI only and the resulting the data was exported as a new csv file. All the further explorations were including checking for null and missing values, outliers etc were done on python.

NULL VALUES:



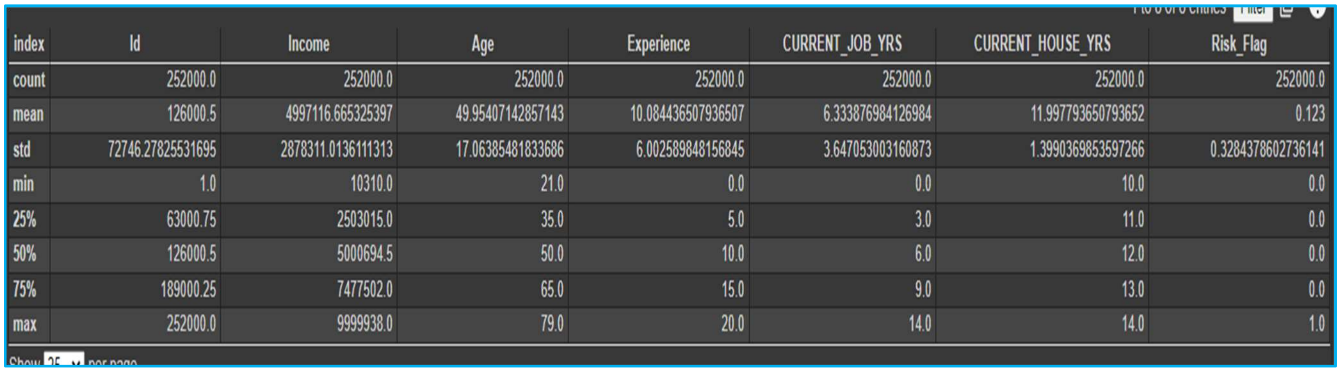
```
train_dataset.isna().sum()
```

```
Id                   0
Income               0
Age                  0
Experience           0
Married/Single       0
House_Ownership      0
Car_Ownership        0
Profession           0
CITY                 0
STATE                0
CURRENT_JOB_YRS      0
CURRENT_HOUSE_YRS    0
Risk_Flag            0
dtype: int64
```

Checking for missing values

DATA DESCRIPTION:

```
train_dataset.describe()
```

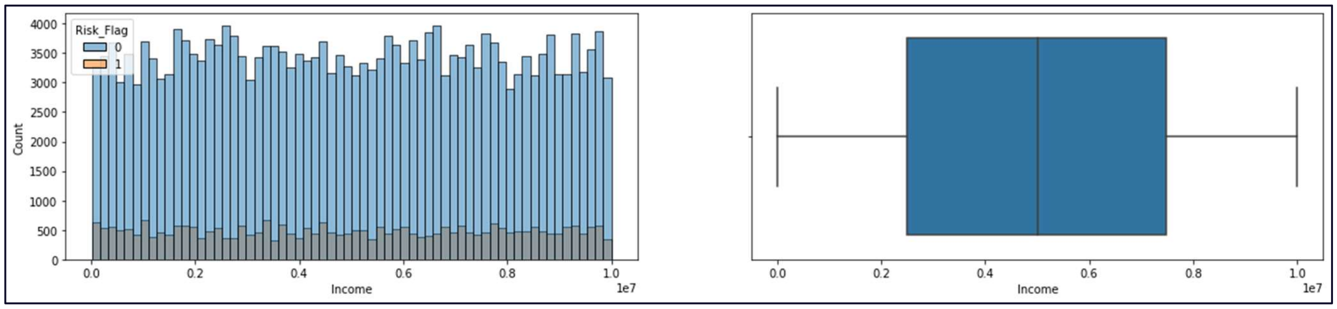


index	Id	Income	Age	Experience	CURRENT_JOB_YRS	CURRENT_HOUSE_YRS	Risk_Flag
count	252000.0	252000.0	252000.0	252000.0	252000.0	252000.0	252000.0
mean	126000.5	4997116.665325397	49.95407142857143	10.084436507936507	6.333876984126984	11.997793650793652	0.123
std	72746.27825531695	2878311.0136111313	17.06385481833686	6.002589848156845	3.647053003160873	1.3990369853597266	0.3284378602736141
min	1.0	10310.0	21.0	0.0	0.0	10.0	0.0
25%	63000.75	2503015.0	35.0	5.0	3.0	11.0	0.0
50%	126000.5	5000694.5	50.0	10.0	6.0	12.0	0.0
75%	189000.25	7477502.0	65.0	15.0	9.0	13.0	0.0
max	252000.0	9999938.0	79.0	20.0	14.0	14.0	1.0

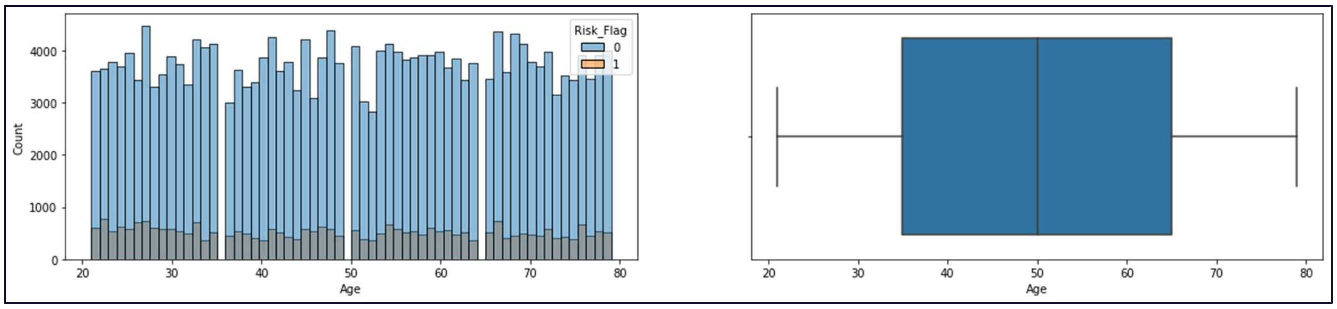
Description

GRAPHS AND BOX PLOTS:

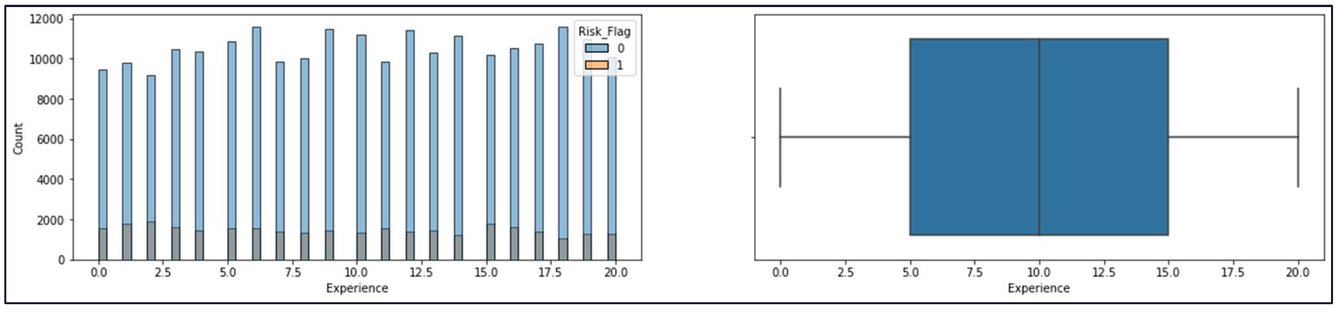
```
# checking histogram and box plot of the numerical variables for range and out
-lier detection

for i in num_col:
  fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(20,4))
  sns.histplot(data=train_dataset, x= i, hue= 'Risk_Flag', ax=ax[0])
  sns.boxplot(data=train_dataset, x= i, ax=ax[1]);
```



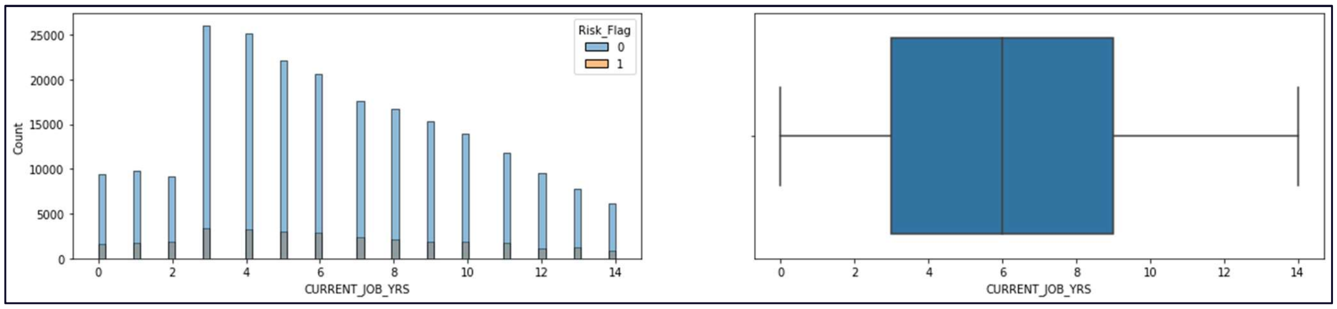
Income



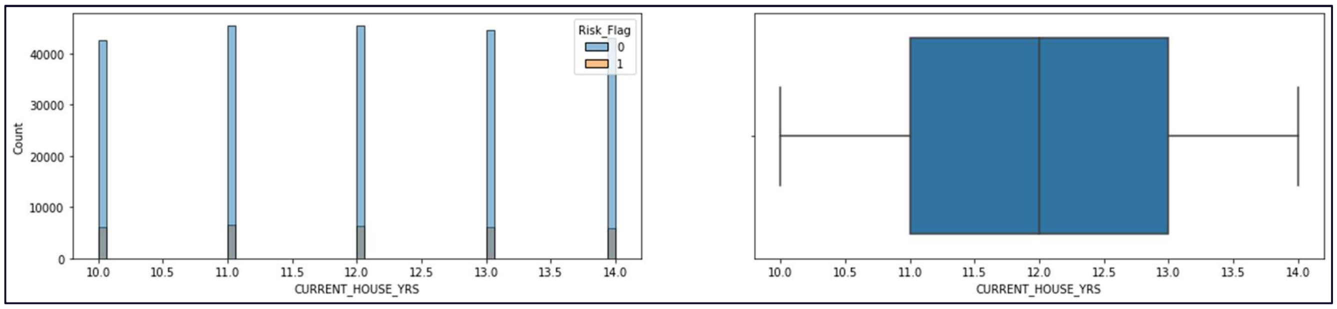
Age



Experience



Years worked in Current Job



Years in Current House

Observation:

- There are no missing values in the data with all the features containing 252k data points
- There are no outliers present
- The features have a uniform distribution
- Some data points in CITY and STATE had special characters and numbers.

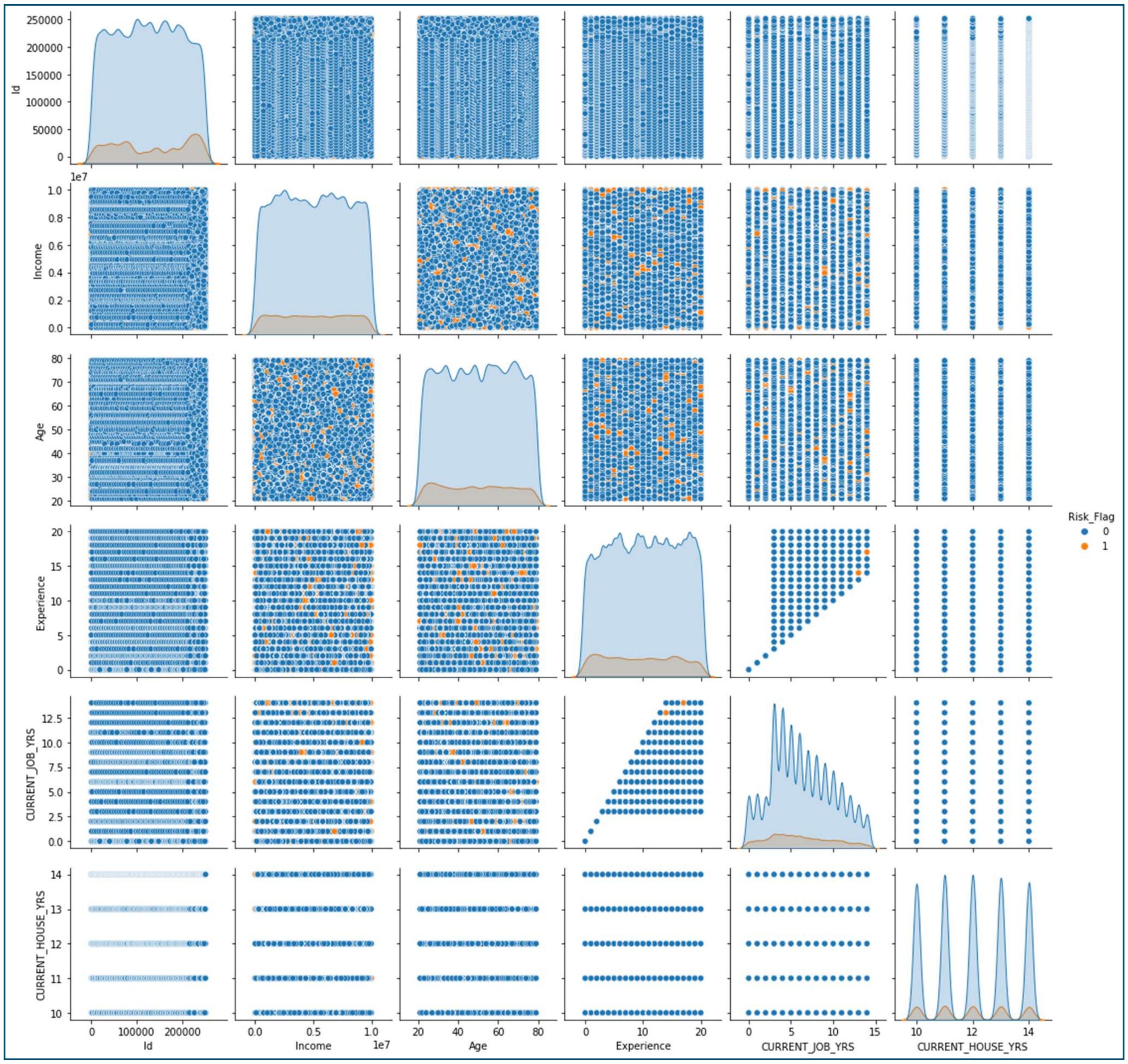
Overall, after going through the data, we have successfully cleaned the data by dealing with special characters and numbers in categorical features. Also, the data was professionally built since there were no outliers or missing values there helping us cleaning the data smoothly.

DATA ANALYSIS

PAIRPLOT

A pairs plot allows us to see both distribution of single variables and relationships between two variables.

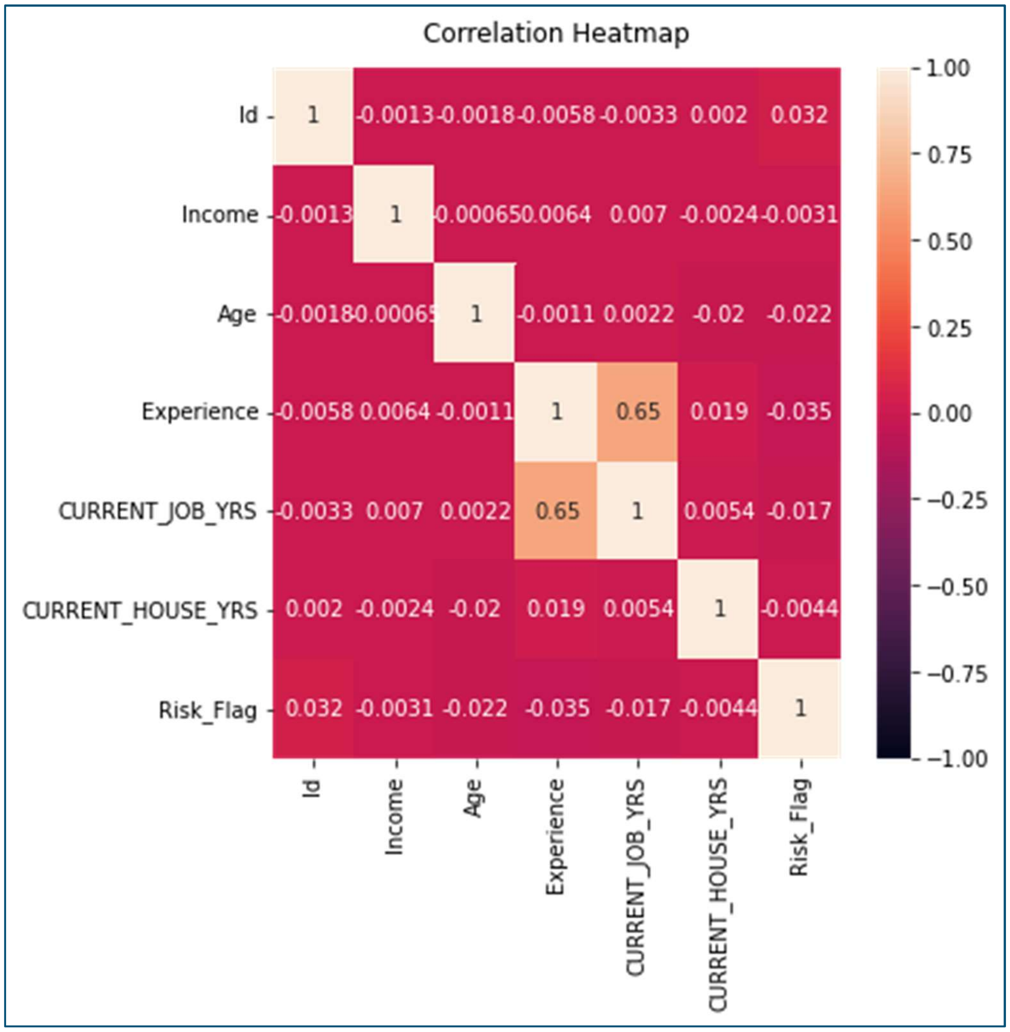
```
# Dataset pairplot wrt to Risk Flag
sns.pairplot(train_dataset, kind= 'scatter', hue='Risk_Flag')
plt.show()
```



CORRELATION MATRIX

Simple table which displays the correlation coefficients for different variables. The matrix depicts the correlation between all the possible pairs of values in a table and higher the value means higher the correlation between the features.

```
# correlation Matrix
plt.figure(figsize=(6, 6))
heatmap = sns.heatmap(train_dataset.corr(), vmin=-1, vmax=1, annot=True)
heatmap.set_title('Correlation Heatmap', fontdict={'fontsize':12}, pad=12)
```



- No clear relationship between features can be noted and can be verified in correlation matrix
- Some relationship is noticed between experience and current job years
- We can deduce that features are not deeply inter-related

CATEGORICAL VARIABLES

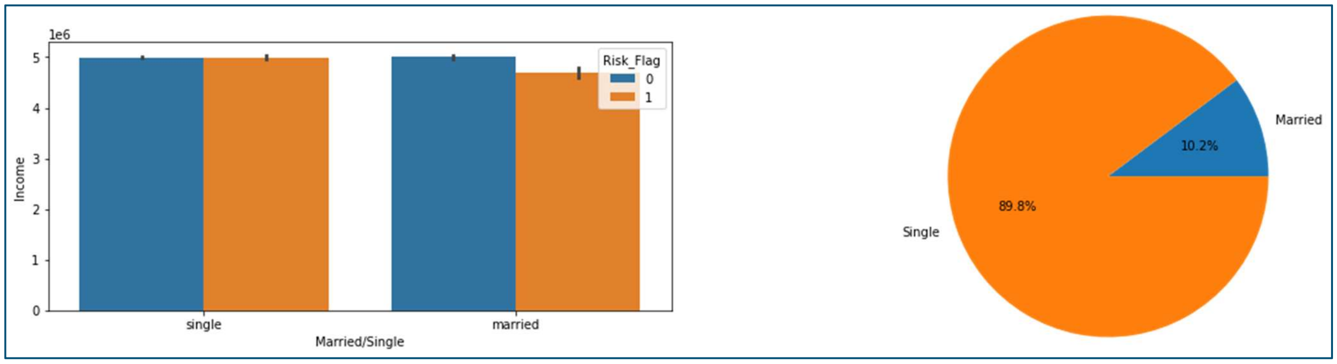
Checking the effect of categorical variables like Marital Status, Car Ownership, House Ownership on risk flag with the help of bar graph and over all distribution of variables with the help of pie chart

Bar Chart: bar chart is used when you want to show a distribution of data points or perform a comparison of metric values across different subgroups of your data. From a bar chart, we can see which groups are highest or most common, and how other groups compare against the others.
In this project, we will be using bar plot to understand different sub groups in a category and how much they account for default cases.

Pie Chart: expresses a part-to-whole relationship in your data.
We will be using pie chart to see the distribution of sub groups in a category and how much a feature account for in the whole of category.

```
# Effect of Marital Status, Car & House Ownership on Risk Flag

# marital status and Income
mylabels = ["Married", "Single"]
fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(20,4))
sns.barplot(y= "Income", x= 'Married/Single', hue= 'Risk_Flag', data = train_d
ataset, ax=ax[0])
pie= train_dataset.groupby('Married/Single')['Married/Single'].count()
plt.pie(pie, radius=1.5, autopct='%1.1f%%', labels = mylabels)

# car ownership and Income
car_labels = ["No", "Yes"]
fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(20,4))
sns.barplot(y= "Income", x= 'Car_Ownership', hue= 'Risk_Flag', data = train_da
taset, ax=ax[0])
pie= train_dataset.groupby('Car_Ownership')['Car_Ownership'].count()
plt.pie(pie, radius=1.5, autopct='%1.1f%%', labels = car_labels)

# house ownership and Income
house_labels = ['no rent, no own', 'owned', 'rent']
fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(20,4))
sns.barplot(y= "Income", x= 'House_Ownership', hue= 'Risk_Flag', data = train_
dataset, ax=ax[0])
pie= train_dataset.groupby('House_Ownership')['House_Ownership'].count()
plt.pie(pie, radius=1.5, autopct='%1.1f%%', labels = house_labels)
```

Married/Single

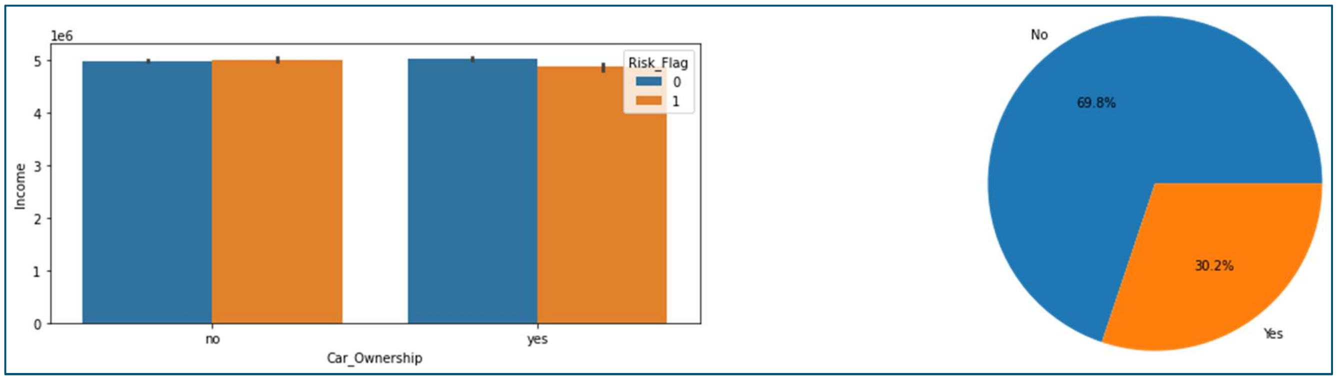


Marital status

As can be seen in the above graph,

- Single people constitute the majority of our data with 89.8% while married are only 10.2%.
- Average income of Single and Married are almost same for non-defaulting category
- Married people who have defaulted have a lower average salary in comparison to their counterparts who have not defaulted.

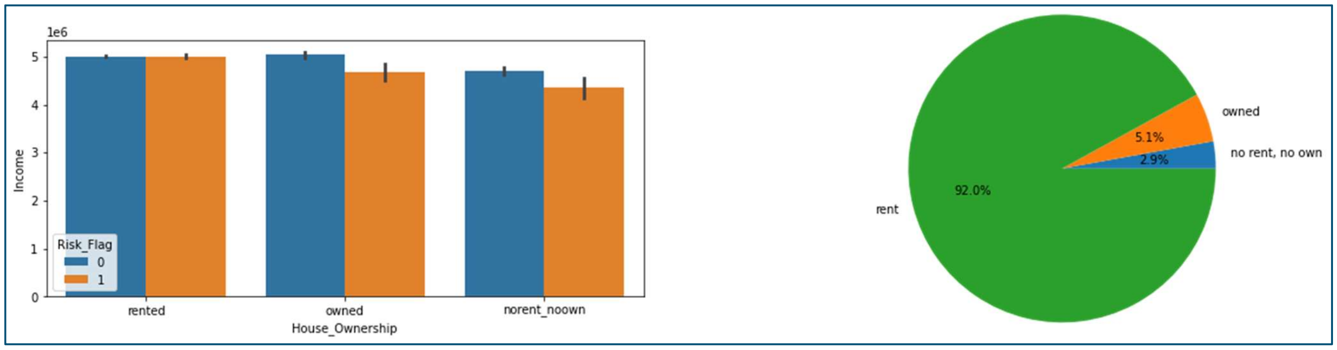
Car Ownership



Car Ownership

- 69.8% of the population does not own a car while 30.2% owns it.
- For the population that does not own a car, average income is almost same for defaulting and non-defaulting categories.
- For the population that own a car, average income is slightly lower for defaulting category than in comparison to defaulting category

House Ownership



House Ownership

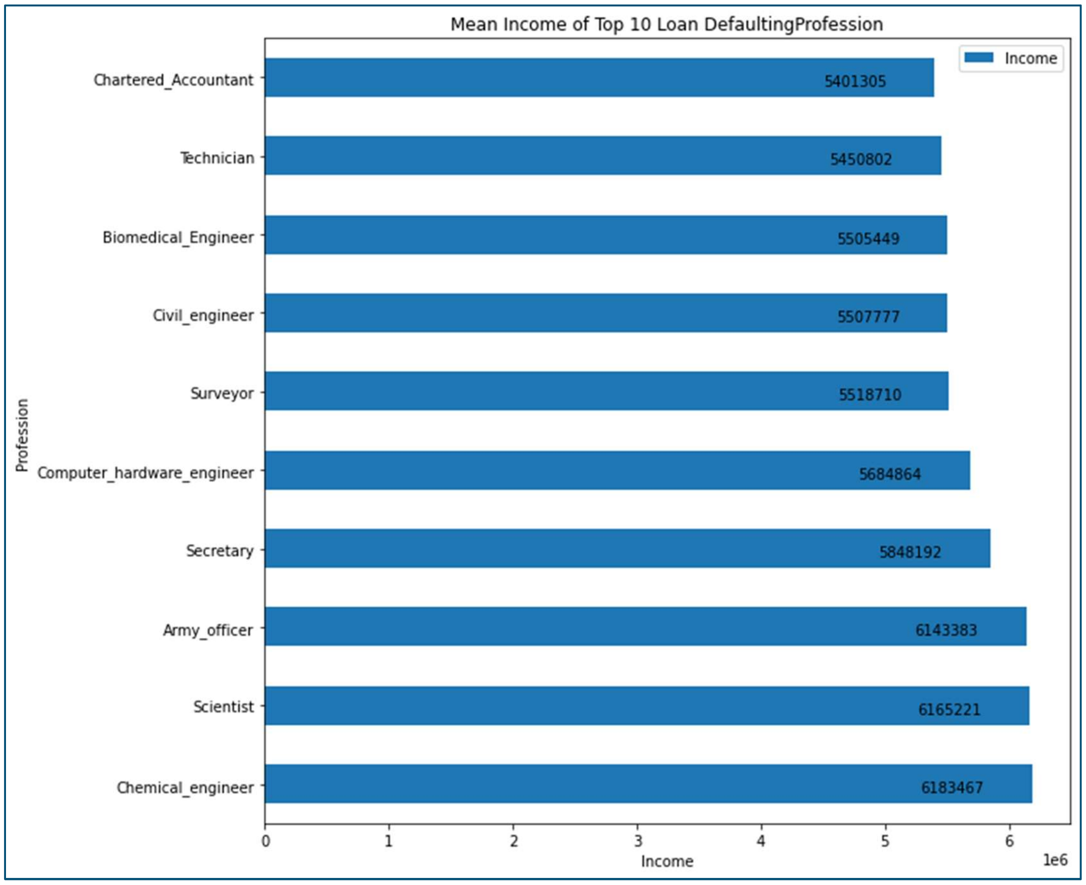
- Majority of the population rents a house at 92% while 5.1% owns it and 2.9% neither owns nor rents.
- For the population that rents a house, average income is almost same for defaulting and non-defaulting categories.
- For the population that owns a house or neither owns non rents, average income is slightly lower for defaulting category than in comparison to defaulting category

INCOME DISTRIBUTION IN TOP DEFAULTING CATEGORIES

Goal is find out average income of top defaulting categories

```
# Income of top 10 loan defaulting professions (https://www.kaggle.com/shahidm
andal/loan-prediction-based-on-customer-behaviour)
def cat_income_analysis(data):
  data_cat = data.loc[data['Risk_Flag'] == 1].groupby([i]).mean()[['Income']].
sort_values(by='Income',ascending=False)
  data_cat.sort_values(by='Income',ascending=False)[:10].plot(kind='barh',figs
ize=(10,10))

  plt.title('Mean Income of Top 10 Loan Defaulting'+ i)
  plt.xlabel('Income')
  plt.ylabel(i)

  for index,value in enumerate(data_cat['Income'][:10]):
      plt.text(value-900000,index-0.1,str(int(value)))

  plt.legend(loc='best')
  plt.show()
```

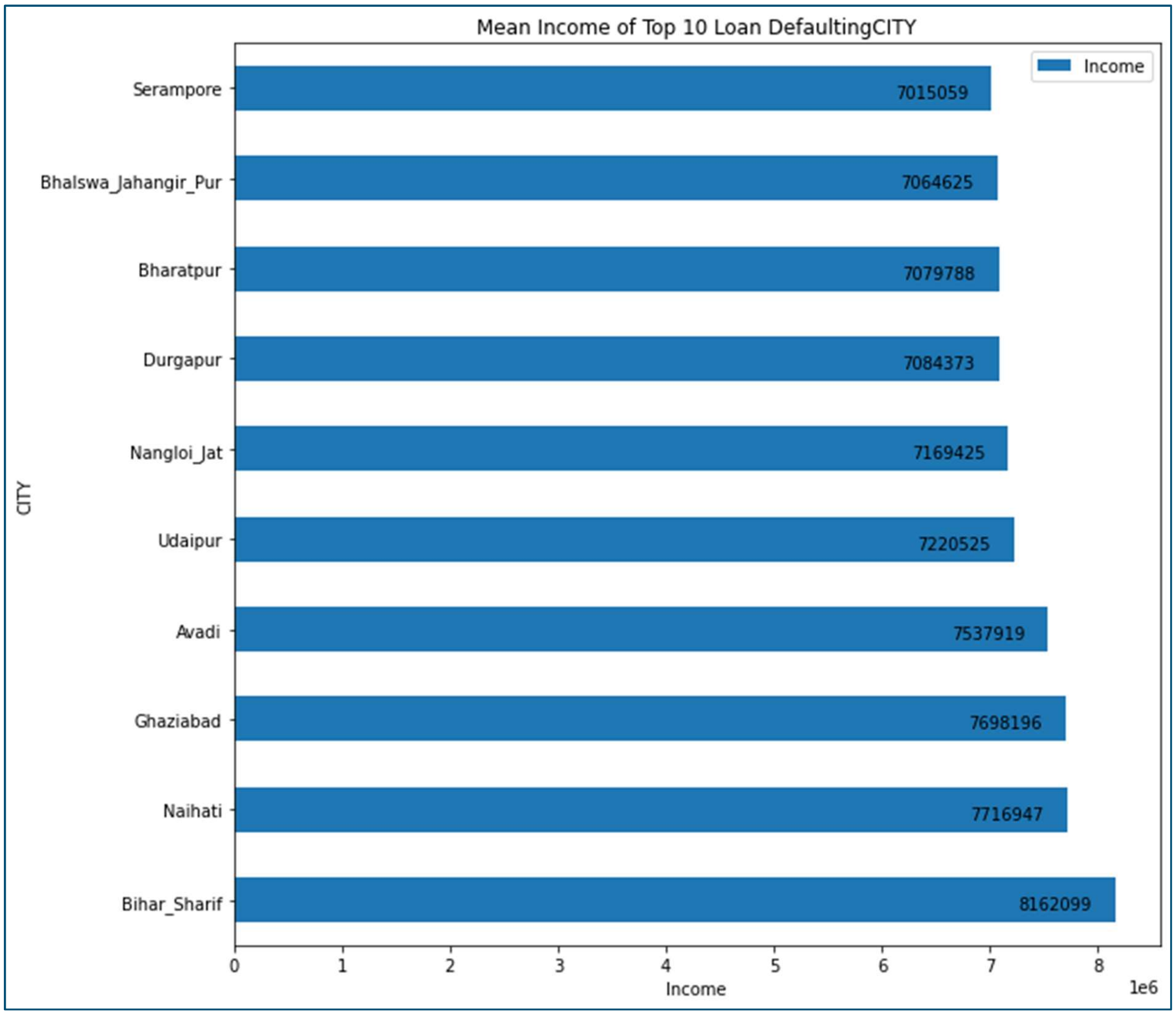
```
for i in cat_col[3:]:
  cat_income_analysis(train_dataset)
  print('-----------------------------------------------')
```



Avg Income of top 10 defaulting profession

The chart represents the top loan defaulting professions and their average income at the time of default.

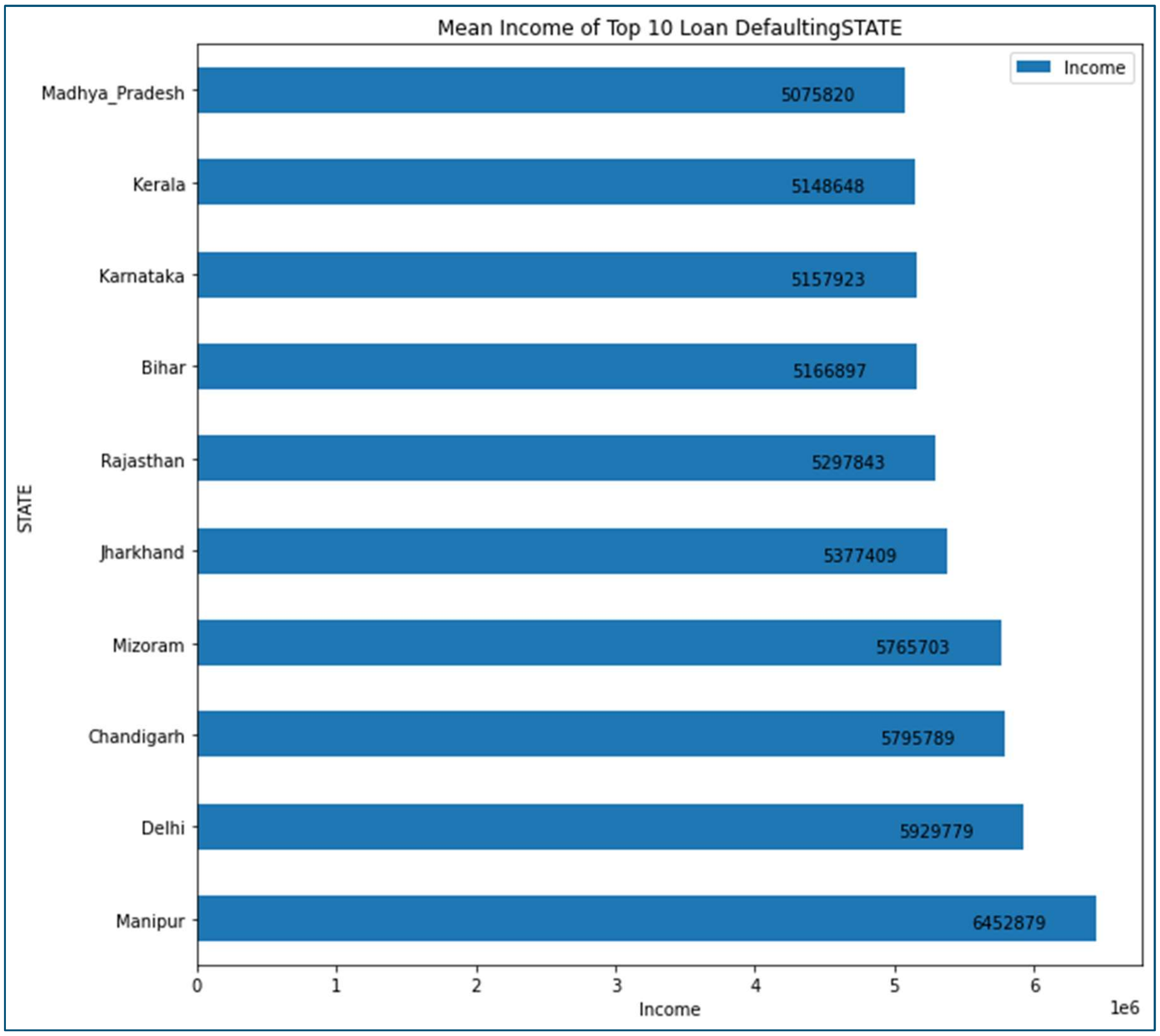
- Chartered Accountants have a high default chance at an income of 5401305 or lower.
- Chemical Engineers have a high default chance at an income of 6183467 or lower.
- Mean income ranges from 54 lakhs to 61 lakhs in defaulting professions



Avg Income of top 10 defaulting city

The chart represents the top loan defaulting cities and their average income at the time of default.

- Borrowers from Serampore have a high default chance at an income of 7064625 or lower.
- Borrowers from Bihar Sharif have a high default chance at an income of 8162099 or lower.
- Mean income ranges from 70 lakhs to 81 lakhs in defaulting cities.



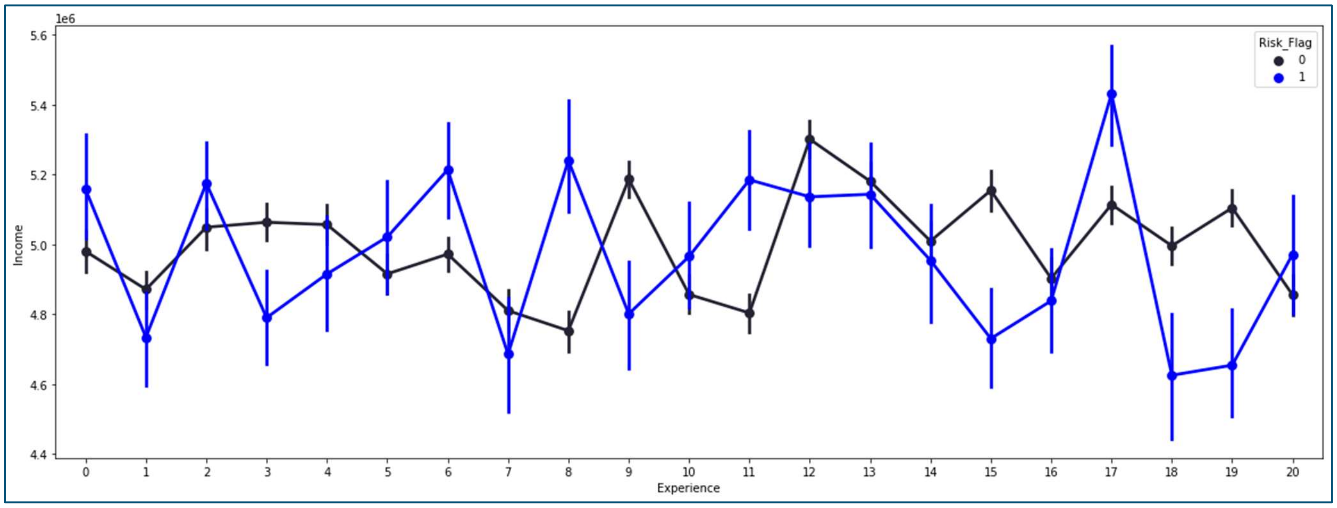
Avg Income of top 10 defaulting state

The chart represents the top loan defaulting states and their average income at the time of default.

- Borrowers from Madhya Pradesh have a high default chance at an income of 5075820 or lower.
- Borrowers from Manipur have a high default chance at an income of 6452879 or lower.
- Mean income ranges from 50 lakhs to 64 lakhs in defaulting cities.

INCOME AND EXPERIENCE

```
plt.figure(figsize=(20,7))
sns.pointplot(data=train_dataset,x='Experience',y='Income', hue= 'Risk_Flag',c
olor='blue')
#C1.set_xticklabels(C1.get_xticklabels(), rotation=45, horizontalalignment='ri
ght')
```



Income Experience Graph

The above graph represents point plot between Income and Experience. Pointplot is used to show point estimates and confidence intervals of the features. It represents an estimate of central tendency for a numeric variable by the position of scatter plot points and provides some indication of the uncertainty around that estimate using error bars.

The above graph shows the income in case of default and in case of full payment of loan at different experience levels. As can be seen from the graph above, on an average mean income in case of default of loan has been lower. It has occurred in 12 years of the 20 years data while in 8 years, the average income in case of default has been higher than the average income in case of full payment.

No proper estimation is noticed in experience and income as both remain consistent over the complete working life span of an individual.

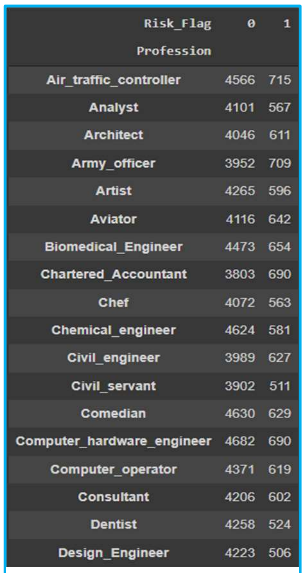
CHI-SQUARE TEST

Chi-square is a statistical test used to examine the differences between categorical variables from a random sample in order to judge goodness of fit between expected and observed results.

```
# code source: https://www.geeksforgeeks.org/python-pearsons-chi-square-test/
# Crosstable : https://www.geeksforgeeks.org/contingency-table-in-python/

def chi_sqr(data):
  stat, p, dof, expected = chi2_contingency(data)

  # interpret p-value
  alpha = 0.05
  print("p value is " + str(p))
  if p <= alpha:
      print('Dependent (reject H0)')
  else:
      print('Independent (H0 holds true)')
```

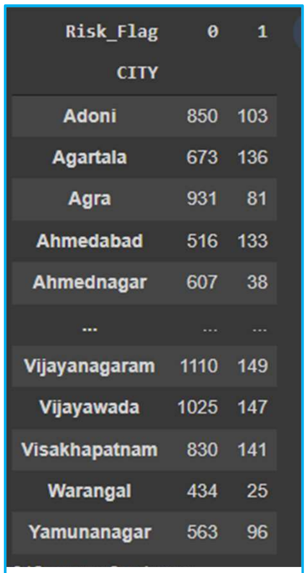
Creating Crosstables

```
# creating crosstables for chi square test

Profession_risk_flag = pd.crosstab(train_dataset["Profession"], train_dataset["Risk_Fla
g"])
CITY_risk_flag = pd.crosstab(train_dataset["CITY"], train_dataset["Risk_Flag"])
STATE_risk_flag = pd.crosstab(train_dataset["STATE"], train_dataset["Risk_Flag"])
```



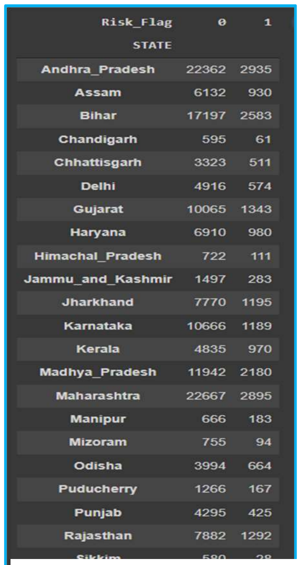
Risk_Flag	0	1
Profession		
Air_traffic_controller	4566	715
Analyst	4101	567
Architect	4046	611
Army_officer	3952	709
Artist	4265	596
Aviator	4116	642
Biomedical_Engineer	4473	654
Chartered_Accountant	3803	690
Chef	4072	563
Chemical_engineer	4624	581
Civil_engineer	3989	627
Civil_servant	3902	511
Comedian	4630	629
Computer_hardware_engineer	4682	690
Computer_operator	4371	619
Consultant	4206	602
Dentist	4258	524
Design_Engineer	4223	506

Profession crosstable



Risk_Flag	0	1
CITY		
Adoni	850	103
Agartala	673	136
Agra	931	81
Ahmedabad	516	133
Ahmednagar	607	38
...	...	...
Vijayanagaram	1110	149
Vijayawada	1025	147
Visakhapatnam	830	141
Warangal	434	25
Yamunanagar	563	96

City Crosstable



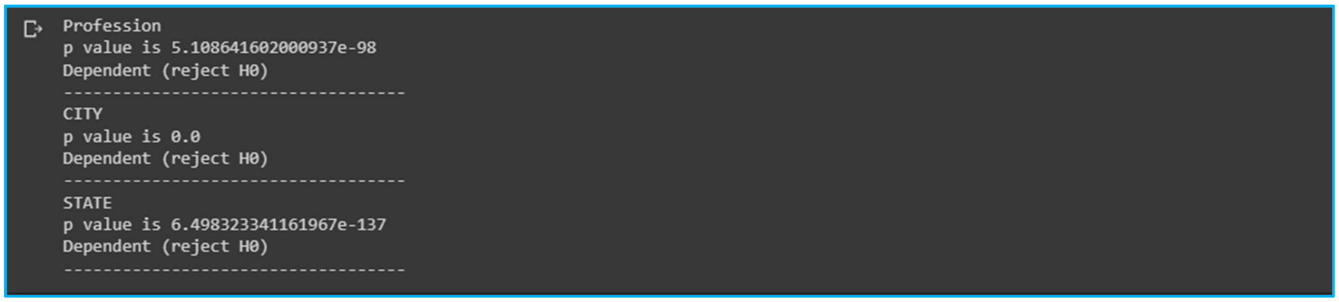
Risk_Flag	0	1
STATE		
Andhra_Pradesh	22362	2935
Assam	6132	930
Bihar	17197	2583
Chandigarh	595	61
Chhattisgarh	3323	511
Delhi	4916	574
Gujarat	10065	1343
Haryana	6910	980
Himachal_Pradesh	722	111
Jammu_and_Kashmir	1497	283
Jharkhand	7770	1195
Karnataka	10666	1189
Kerala	4835	970
Madhya_Pradesh	11942	2180
Maharashtra	22667	2895
Manipur	666	183
Mizoram	755	94
Odisha	3994	664
Puducherry	1266	167
Punjab	4295	425
Rajasthan	7882	1292
Sikkim	580	28

State Crosstable

We would be conducting a chi square test between high cardinality categorical variables like CITY, Profession and STATE with the risk flag in order to ascertain if there is any fit. If dependence is not ascertained during the testing process, we would drop that feature from the model training process.

Hypothesis Testing

```
# CHI-SQUARE TESTING FOR CATEGORICAL VARIABLES WRT TO RISK FLAG
# H0 : there is no relation between the variable (Profession, City, State) and
 the Risk_Flag (Independent Variable)
# H1 : There is relation between the vaiable and the Risk Flag

cat_chi_sqr = [Profession_risk_flag, CITY_risk_flag, STATE_risk_flag]

for i in range(len(cat_chi_sqr)):
  print(cat_col[3:][i])
  chi_sqr(cat_chi_sqr[i])
  print('-----------------------------------')
```



```
Profession
p value is 5.108641602000937e-98
Dependent (reject H0)
-----------------------------------
CITY
p value is 0.0
Dependent (reject H0)
-----------------------------------
STATE
p value is 6.498323341161967e-137
Dependent (reject H0)
-----------------------------------
```

In the test above, we have taken the 3 features with high cardinality and tested them with risk flag to check if risk flag was dependent on the 3 features. A cross table was formed for all the 3 features and with .05 significance level, chi-square test was run on all the 3 features.

As can be seen from the test above, the risk flag is dependent on all the 3 categorical features and hence we cannot drop the features in the pre-processing stage.

OBSERVATIONS

- There are no missing values in the data with all the features containing 252k data points
- There are no outliers present
- The features have a uniform distribution
- Some data points in CITY and STATE had special characters and numbers.
- CITY, STATE and Profession have high cardinality
- The data was heavily imbalanced in favour of non-defaulters. Some form of sampling technique could be used to address the issue
- No clear relationship between features can be noted and can be verified in correlation matrix
- Some relationship is noticed between experience and current job years
- We can deduce that features are not deeply inter-related
- Married people who have defaulted have a lower average salary in comparison to their counterparts who have not defaulted.
- For the population that own a car, average income is slightly lower for defaulting category than in comparison to defaulting category
- For the population that owns a house or neither owns non rents, average income is slightly lower for defaulting category than in comparison to defaulting category
- Mean income ranges from 70 lakhs to 81 lakhs in defaulting cities.
- Mean income ranges from 54 lakhs to 61 lakhs in defaulting professions
- The risk flag is dependent on all the 3 categorical features and hence we cannot drop the features in the pre-processing stage

DATA PREPROCESSING

process that takes raw data and transforms it into a format that can be understood and analysed by computers and machine learning.

Preprocessing Steps:

- Copying the complete database
- Converting categorical features to numerical variables with the help of one hot encoding
- Scaling numerical features with the help of min-max scaler

```
#converting categorical to numerical and scaling numerical data
def data_preprocess(data):
  data = pd.get_dummies(data, columns= cat_col)
  scaler = MinMaxScaler()
  scaler.fit(data[num_col])
  data[num_col]=scaler.transform(data[num_col])
  return data
```

```
# preprocessing data
train_data = data_preprocess(train_data)
test_data = data_preprocess(test_data)

train_data.shape, test_data.shape, target.shape
```

```
((252000, 407), (28000, 407), (252000,))
```

After pre-processing the data, the resulting database was of the shape (252000,407) for the train data and (28000,407) for the test data.

Alternate Pre-processing Tecchniques:

- For reducing the dimensionality of the data, alternate encoding techniques like Target encoding could have been used
- Top 10 occurring categorical variables with high cardinality could have been shortlisted and one hot encoded.
- Standard Scaler could have been used instead of min max scaler

MODELLING

Process of predicting something useful from the datasets with the help machine learning algorithms.

Splitting Data

We would be splitting the train data into train and validation data with a ratio of 75:25 and then store them into new train test and validation variables

```
# train-test validation split

from sklearn.model_selection import train_test_split

x_train ,x_val , y_train ,y_val =train_test_split(train_data, target, test_siz
e=0.25, random_state=42)
x_test = test_data
```

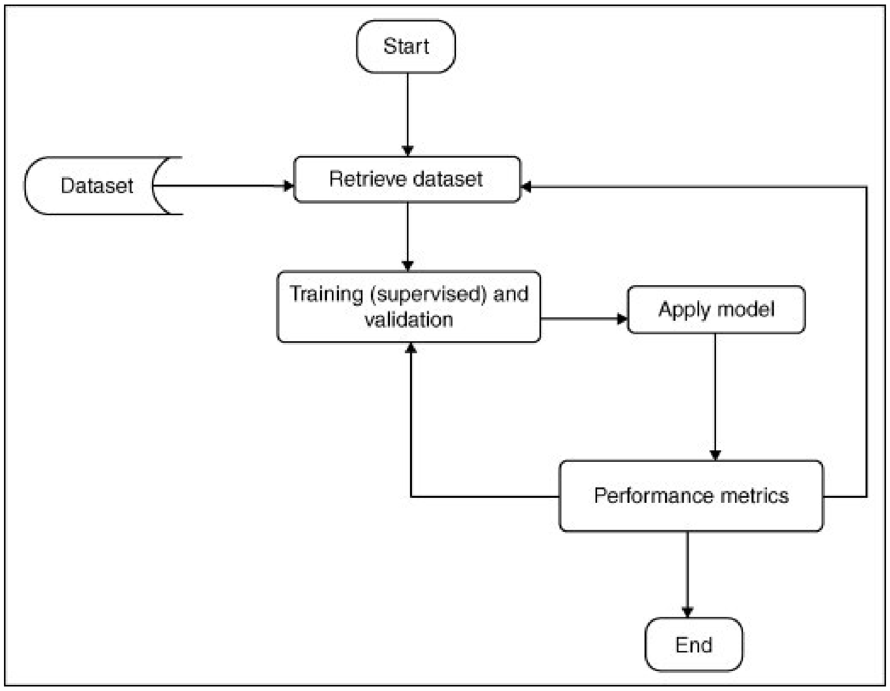
Setting Metrics Evaluation

```
# f beta with more emphasis on false negative
def f_beta(y_true, y_pred):
    return fbeta_score(y_true, y_pred, beta=2)

def mod_score(y_true, y_pred):
  return 'F_beta: ', f_beta(y_true, y_pred), ', Roc_Auc_Score: ',  roc_auc_sco
re(y_true, y_pred)
```

As discussed previously we will be using f beta score and roc auc score for our model . A function with the name *mod_score* is made that taken in the input as original target value and the predicted target value and returns the metric score.

BASELINE MODEL

Baseline model is a method that uses heuristics, simple summary statistics, randomness, or machine learning to create predictions for a dataset. You can use these predictions to measure the baseline's performance (e.g., accuracy)– this metric will then become what you compare any other machine learning algorithm against.



Modelling Process

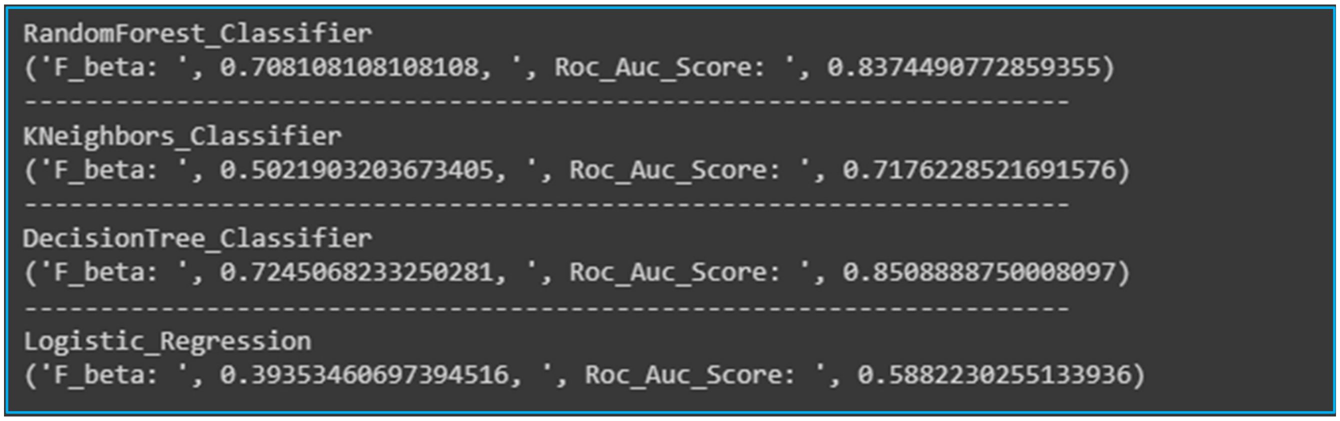
Furthermore the baseline model output serve as one of the input for the process discussed. In this design, the “retrieve dataset” process will retrieve the the dataset at and pass it over to the “training and validation” process where x-validation used and the model applied for training.

The most important component of this model are the reference classifiers used for each loop from the “performance metric” to “training and validation.” Also, the “performance metric” loop back to “retrieve dataset” after every complete rotation of obtaining performance metrics until all the three datasets have been passed through the model. In order to successfully carry out the training and testing process, some parameters are used to achieve the best result

For the validating and testing data, we would be creating a function *mod_pred_1* that would take training and test data as its input and pass them through all the four machine learning models one by one and finally giving evaluating the model on the basis of metric selected and giving the model score.

```
# Baseline Models
# testing with validation data

def mod_pred_1(xtrain, ytrain, xval, yval):
  model= [RandomForestClassifier(class_weight='balanced'), KNeighborsCl
assifier(), DecisionTreeClassifier(class_weight='balanced'), LogisticRe
gression(class_weight='balanced', solver='lbfgs')]
  mod_name= ['RandomForest_Classifier', 'KNeighbors_Classifier', 'Decis
ionTree_Classifier', 'Logistic_Regression']
  for x in range(4):
    print(mod_name[x])
    clf = model[x].fit(xtrain, ytrain)
    y_pred = clf.predict(xval)
    scr = mod_score(yval, y_pred)
    print(scr)
    print('-----------------------------------------------------------')
```

```
mod_pred_1(x_train, y_train, x_val, y_val)
```



```
RandomForest_Classifier
('F_beta: ', 0.708108108108108, ', Roc_Auc_Score: ', 0.8374490772859355)
----------------------------------------------------------------------
KNeighbors_Classifier
('F_beta: ', 0.5021903203673405, ', Roc_Auc_Score: ', 0.7176228521691576)
----------------------------------------------------------------------
DecisionTree_Classifier
('F_beta: ', 0.7245068233250281, ', Roc_Auc_Score: ', 0.8508888750008097)
----------------------------------------------------------------------
Logistic_Regression
('F_beta: ', 0.39353460697394516, ', Roc_Auc_Score: ', 0.5882230255133936)
```

As can be seen from the scores above:

- Decision tree provided the best fbeta score at 70.8% and Roc score at 85%.
- Logistic Regession was the worst performing model

HANDLING IMBALANCED DATA

Usually the data we gather is heavily imbalanced. The training samples are not equally distributed across the target classes. For instance, in current case of loan classification problem, it is effortless to get the ‘no default’ data, in contrast to, ‘default details. As a result, the model is more biased to the class which has a large number of training instances which degrades the model’s prediction power.

It also results in an increase in Type II errors, in the case of a typical binary classification problem.

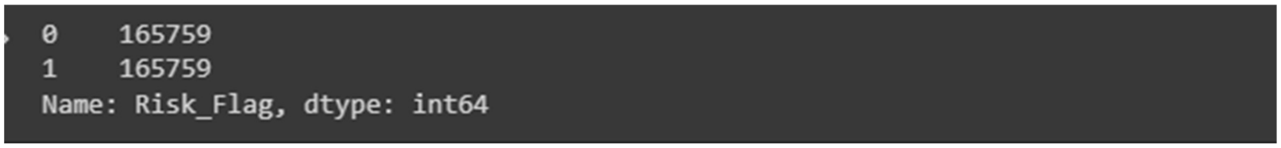
The main two methods that are used to tackle the class imbalance is up sampling/oversampling and down sampling/under sampling. The sampling process is applied only to the training set and no changes are made to the validation and testing data

Fo the purpose of this study, we have upsampled out data so as to reduce the imbalance in the risk flag data.

UPSAMPLING DATA

```
# upsampling
# upsampling y train
from imblearn.over_sampling import SMOTE

sm = SMOTE(random_state = 42)
x_train_1, y_train_1 = sm.fit_resample(x_train, y_train)
```

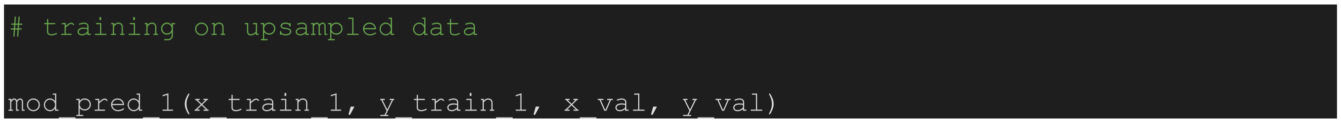
```
y_train_1.value_counts()
```



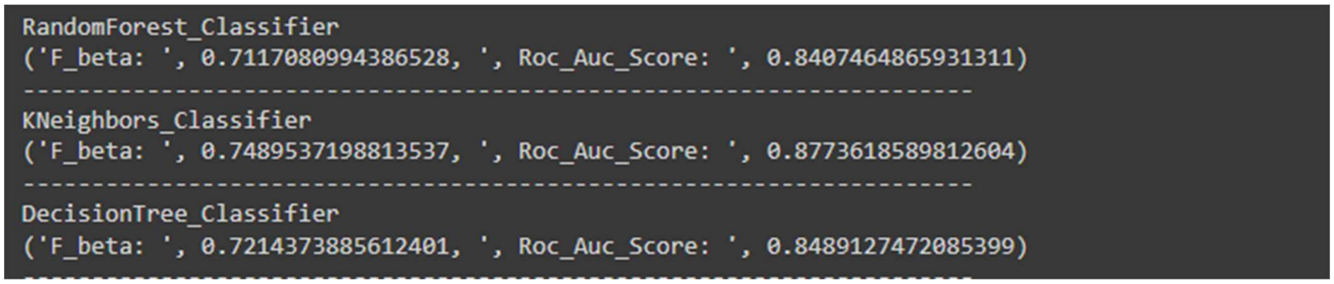
```
0    165759
1    165759
Name: Risk_Flag, dtype: int64
```

After upsampling the data, both default and no default numbers were matched at 165759 each.

SCORE ON UPSAMPLED DATA



```
# training on upsampled data

mod_pred_1(x_train_1, y_train_1, x_val, y_val)
```



```
RandomForest_Classifier
('F_beta: ', 0.7117080994386528, ', Roc_Auc_Score: ', 0.8407464865931311)
----------------------------------------------------------------------
KNeighbors_Classifier
('F_beta: ', 0.7489537198813537, ', Roc_Auc_Score: ', 0.8773618589812604)
----------------------------------------------------------------------
DecisionTree_Classifier
('F_beta: ', 0.7214373885612401, ', Roc_Auc_Score: ', 0.8489127472085399)
----------------------------------------------------------------------
```

As we can notice from the scores above:

- Scores have increased after upsampling the data
- Increase in score in case of decision tree is minimal as is the case in random forest as well
- The score has drastically increased in Kneighbors and has become the best performing model.

We have not considered down-sampling because it is a mechanism that reduces the count of training samples falling under the majority class. As it helps to even up the counts of target categories by removing the collected data, we tend to lose so much valuable information which is not advisable in most machine learning models.

HYPER-PARAMETER TUNING

A Machine Learning model is defined as a mathematical model with a number of parameters that need to be learned from the data. By training a model with existing data, we are able to fit the model parameters. However, there is another kind of parameters, known as Hyperparameters, that cannot be directly learned from the regular training process. They are usually fixed before the actual training process begins. These parameters express important properties of the model such as its complexity or how fast it should learn.

Since we have noticed that K Neighbors was the best performing model, we would be tuning the hyper parameters of K Neighbors and try and get a better score from the data.

Parameters that we would be tuning are

- Leaf Size
- N Neighbors: Number of neighbors to use by default
- P: Power parameter for the Minkowski metric

```
from sklearn.model_selection import GridSearchCV

#List Hyperparameters that we want to tune.
leaf_size = list(range(25,50,2))
n_neighbors = list(range(1,10,2))
p=[1,2]
#Convert to dictionary
hyperparameters = dict(leaf_size=leaf_size, n_neighbors=n_neighbors, p=p)
#Create new KNN object
knn_2 = KNeighborsClassifier()
#Use GridSearch
clf = GridSearchCV(knn_2, hyperparameters, cv=2)
#Fit the model
best_model = clf.fit(x_train_1, y_train_1)


#Print The value of best Hyperparameters
print('Best leaf_size:', best_model.best_estimator_.get_params()['leaf_size'])
print('Best p:', best_model.best_estimator_.get_params()['p'])
print('Best n_neighbors:', best_model.best_estimator_.get_params()['n_neighbor
s'])
```

```
# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000, num = 10
)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(10, 110, num = 11)]
max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4]
# Method of selecting samples for training each tree
bootstrap = [True, False]
# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}
print(random_grid)
```

```
{'n_estimators': [200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000],
'max_features': ['auto', 'sqrt'], 'max_depth': [10, 20, 30, 40, 50, 60, 70,
80, 90, 100, 110, None], 'min_samples_split': [2, 5, 10], 'min_samples_leaf':
[1, 2, 4], 'bootstrap': [True, False]}
```

```
model = RandomForestClassifier()
rf_random = RandomizedSearchCV(estimator = model, param_distributions = random
_grid, cv = 3, verbose=True, random_state=42)
rf_random.fit(x_train, y_train)
best_random = rf_random.best_estimator_
best_random.fit(x_train, y_train)
print("Fitting done")
print(rf_random.best_params_)
y_pred = best_random.predict(x_val)
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
Fitting done
{'n_estimators': 1000, 'min_samples_split': 10, 'min_samples_leaf': 1,
'max_features': 'auto', 'max_depth': 80, 'bootstrap': False}
```

```
# Random Forest Classifier

rf_clf_1 = RandomForestClassifier(class_weight='balanced', n_estimators= 1000,
 min_samples_split= 10, min_samples_leaf= 1, max_features= 'auto', max_depth=
80, bootstrap= False)
rf_clf_1 = rf_clf_1.fit(x_train, y_train)
rf_pred_5 = rf_clf_1.predict(x_val)
```

```
print(mod_score(y_val, rf_pred_5))
```

```
F_beta:  0.7178247021351892 Roc_Auc_Score:  0.8439858078628607 Accuracy:
0.8914444444444445
```

As can be seen from the scores, there was major impact of tuning the model on the scores. While the scores increased slightly for K neighbors it were the same for decision tree and Random Forest.

Hence we can conclude this model with a F beta score of 74.8% and an ROC score of 87.73 which can give us a good understanding of whether to give loan to a particular borrower or to do further risk benefit analysis.

REFERENCES

1. Ideal Loans and Problem Loans: Causes of Problem Loans
2. Redefining loan monitoring and early warning signal detection through an integrated solution
3. Detecting loan defaults at an early stage using models of machine intelligence
4. A Survey on Ensemble Model For Loan Prediction
5. An Ensemble Model of Multiple Classifiers for Time Series Prediction
6. Loan Credibility Prediction System Based on Decision Tree Algorithm
7. Weight-Selected Attribute Bagging for Credit Scoring
8. Loan default prediction using decision trees and random forest: A comparative study
9. A Survey of Machine Learning in Credit Risk
10. A study on predicting loan default based on the random forest algorithm
11. Business Analytics using Random Forest Trees for Credit Risk Prediction: A Comparison Study
12. Loan Prediction Model Using Logistic Regression, Decision Tree, Random Forest