

**University of New Haven**  
**Tagliatela college of Engineering**  
**Data Science, Computer Engineering and Computer**  
**Science**



**Credit Card Approval Prediction**

**MACHINE LEARNING**

DSCI-6003-02

Date: 04/24/2022

By,  
Boddu Upender

Supervisor,  
Muhammad Aminul Islam

# Introduction

Machine Learning is an application that provides computers the ability to learn and improve from the experience without being explicitly programmed. Machine Learning is classified into different types. They are as follows:

## 1) Supervised Learning:

Supervised learning is the task of machine learning that maps the input to an output based on input-output patterns to the system.

Supervised learning is classified into two types of regression and classification.

**A) Regression:** Regression is classified under the same branch of supervised learning. In machine learning regression algorithm try's to estimate the mapping function( $f$ ) from the variable ( $x$ ) to numerical or continuous output variable( $y$ ).

**Example:** To predict the prices of the houses, that is the regression task because price of the houses will be the continuous output.

**B) Classification:** Classification is classified under the same branch of supervised learning. In machine learning classification algorithm try's to calculate the mapping function( $f$ ) from the input ( $x$ ) to discrete or categorial output variables( $y$ ).

**Example:** To predict the prices of the houses "sell more or less than the recommended retail price." Here, the houses will be classified whether their prices fall into two discrete categories: above or below the said price.

## **2)Unsupervised Learning:**

Unsupervised learning is a self-learning technique in which system has to discover the features of the input population by its own and no prior set of categories are used.

Linear Regression is the example of Regression.

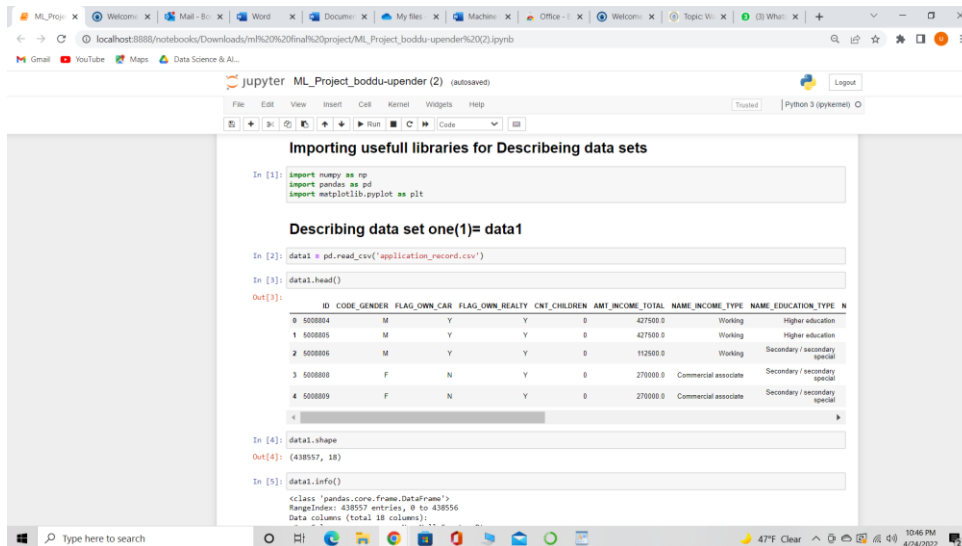
This project aims at developing two supervised Machine Learning Model- Linear Regression from scratch using Python and NumPy. The developed code has various parameters. From the given models Bias-Variance Trade-off and Prediction the best model can be selected from the given data. This project enables the users to use the parameters and select the best model in Linear Regression.

## **Motivation**

I have selected this project because it is one of the main aspects of life. It is important to know if your credit card gets approved or not well before in time to plan life and other situations. It is very challenging to predict this information by humans as there are several dependent variables. My aim is to Build a machine learning model which is used predict whether an applicant is a good customer or a bad customer. So many people apply for credit card, and we need to understand which applicant is good enough to get the approval. A machine can predict the approval if well trained. In this project several methods, models, and parameters are tested to get highest prediction accuracy. My aim is to Build a machine learning model which is used predict whether an applicant is a good customer or a bad customer. So many people apply for credit card, and we need to understand which applicant is good enough to get the approval

## Describing Data Set:

I have downloaded the data from Kaggle. We know that the shape of the data set 4 lakhs 38000 rows and 18 columns. All the columns are either integer type float type or object type. For both test data and train data checked for outliers & null values now the data is balanced and cleaned.



The screenshot shows a Jupyter Notebook interface with the following content:

```
Importing usefull libraries for Describing data sets

In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

Describing data set one(1)= data1

In [2]: data1 = pd.read_csv('application_record.csv')

In [3]: data1.head()

Out[3]:
```

ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCOME_TYPE	NAME_EDUCATION_TYPE	N
0	5000804	M	Y	Y	0	427500.0	Working	Higher education
1	5000805	M	Y	Y	0	427500.0	Working	Higher education
2	5000806	M	Y	Y	0	112500.0	Working	Secondary / secondary special
3	5000808	F	N	Y	0	270000.0	Commercial associate	Secondary / secondary special
4	5000809	F	N	Y	0	270000.0	Commercial associate	Secondary / secondary special

```


In [4]: data1.shape

Out[4]: (438557, 18)

In [5]: data1.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 438557 entries, 0 to 438556
Data columns (total 18 columns):
```

- **ID-** It shows the customer number.
- **CODE\_GENDER:** It shows the gender of the customer.
- **FLAG\_OWN\_CAR:** It shows whether the customer owns a car or not.
- **FLAG\_OWN\_REALTY:** It shows whether the customer owns a property or not.
- **CNT\_CHILDREN:** It shows how many children does customer have.
- **AMT\_INCOME\_TOTAL:** it shows the yearly income of the customer.
- **NAME\_INCOME\_TYPE:** It shows what type of income does the customer get.

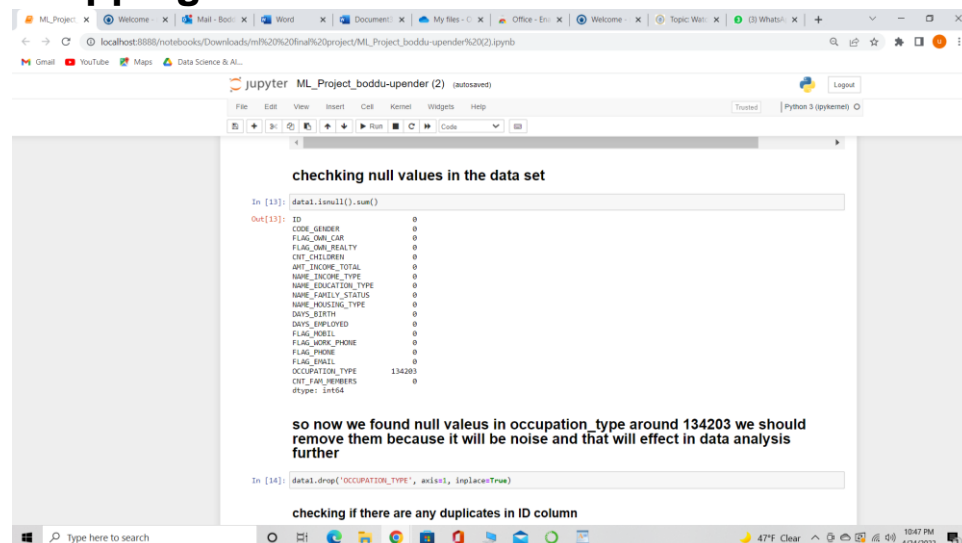
**NAME\_EDUCATION\_TYPE:** It shows the qualification of the customer.

- **MARITALSTATUS:** It shows whether the customer is single/married/widow/divorcee.
- **NAME\_HOUSING\_TYPE:** It shows the way of living of the customer.
- **DAYS\_BIRTH:** It shows the date of birth of the customer.
- **DAYS-EMPLOYED:** It shows from when the customer is working.
- **FLAG\_MOBILE:** Does the customer have mobile.
- **FLAG\_WORK\_PHONE:** Does the customer have work phone.
- **FLAG\_PHONE:** Does the customer have the phone.

## Exploring Data Analysis:

After describing data set, we should find which data is useful and which data is not useful so for that we should describe the data and find null values and duplicates and we should drop them from data

## Dropping Null values -



The screenshot shows a Jupyter Notebook interface with the following content:

```
checking null values in the data set
```

```
In [13]: data.isnull().sum()
```

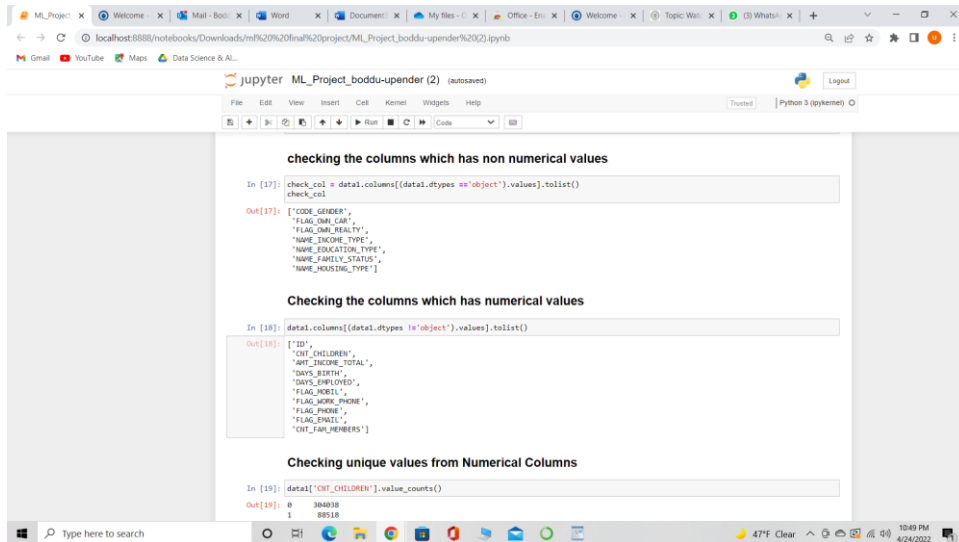
```
Out[13]: ID 0
CODE_GENDER 0
FLAG_OWN_CAR 0
FLAG_OWN_REALTY 0
CNT_CHILDREN 0
APMT_INCOME_TOTAL 0
NAME_INCOME_TYPE 0
NAME_EDUCATION_TYPE 0
NAME_FAMILY_STATUS 0
NAME_HOUSING_TYPE 0
DAYS_BIRTH 0
DAYS_EMPLOYED 0
FLAG_MOBILE 0
FLAG_WORK_PHONE 0
FLAG_PHONE 0
FLAG_EMAIL 0
OCCUPATION_TYPE 134203
CNT_FAM_MEMBERS 0
dtype: int64
```

so now we found null values in occupation\_type around 134203 we should remove them because it will be noise and that will effect in data analysis further

```
In [14]: data.drop('OCCUPATION_TYPE', axis=1, inplace=True)
```

```
checking if there are any duplicates in ID column
```

Now we should check the columns which has non numerical values, numeric values and unique values.



The screenshot shows a Jupyter Notebook with the following code and output:

```
checking the columns which has non numerical values

In [17]: check_col = data.columns[(data.dtypes == "object").values].tolist()
check_col

Out[17]: ['CODE_GENDER',
          'FLAG_Own_CAR',
          'FLAG_Own_REALTY',
          'NAME_INCOME_TYPE',
          'NAME_EDUCATION_TYPE',
          'NAME_FAMILY_STATUS',
          'NAME_HOUSING_TYPE']

Checking the columns which has numerical values

In [18]: data.columns[(data.dtypes != "object").values].tolist()

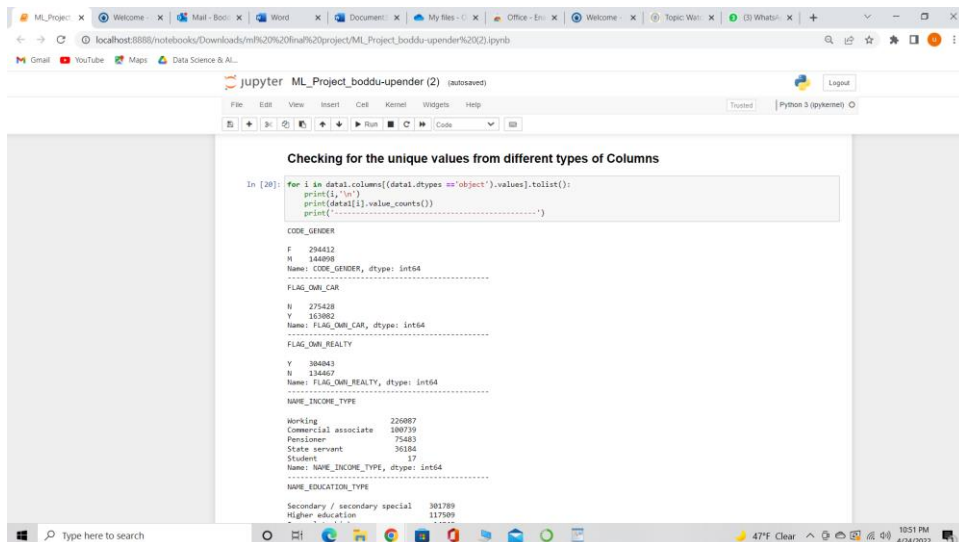
Out[18]: ['ID',
          'CNT_CHILDREN',
          'AMT_INCOME_TOTAL',
          'DAYS_BIRTH',
          'DAYS_EMPLOYED',
          'FLAG_BREEL',
          'FLAG_WORK_PHONE',
          'FLAG_PHONE',
          'FLAG_EMAIL',
          'CNT_FAM_MEMBERS']

Checking unique values from Numerical Columns

In [19]: data['CNT_CHILDREN'].value_counts()

Out[19]: 0    304878
         1     88518
```

## Checking the unique values for different types of Columns



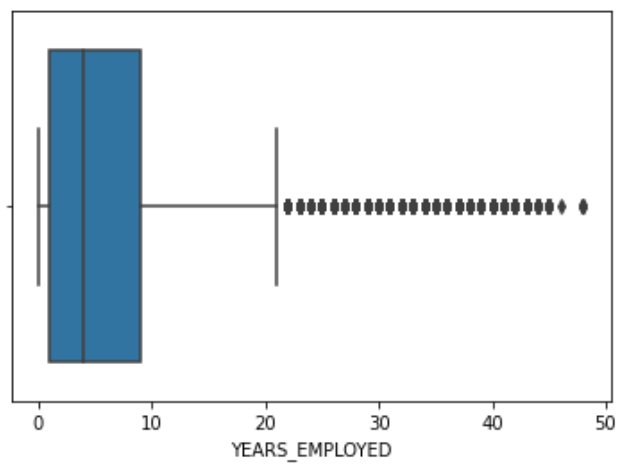
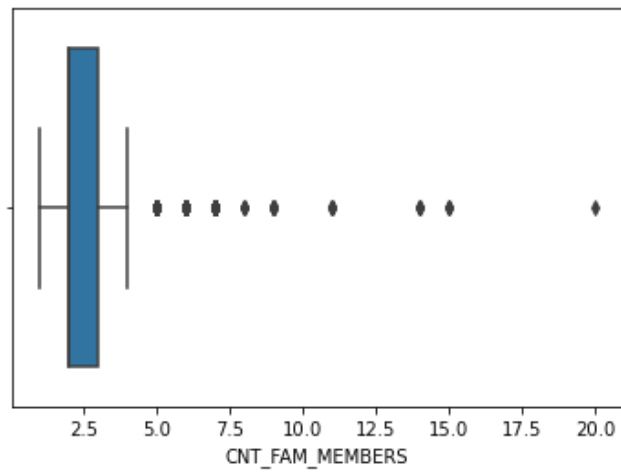
The screenshot shows a Jupyter Notebook with the following code and output:

```
Checking for the unique values from different types of Columns

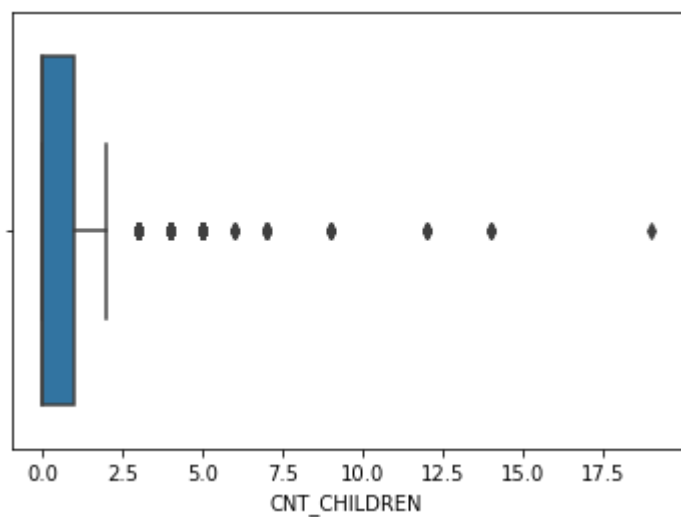
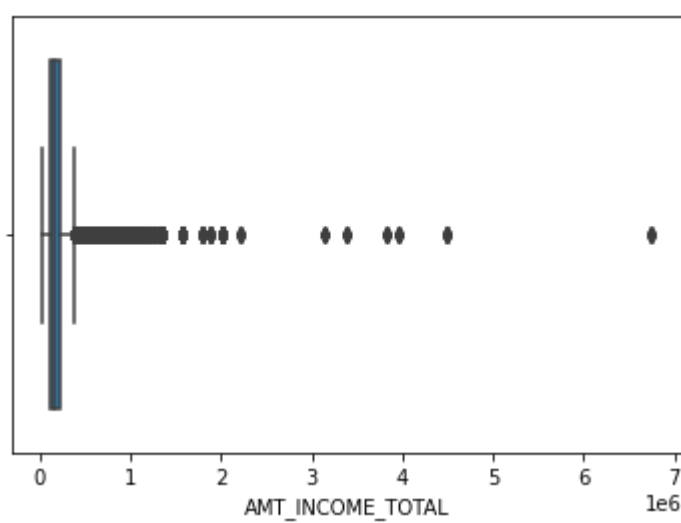
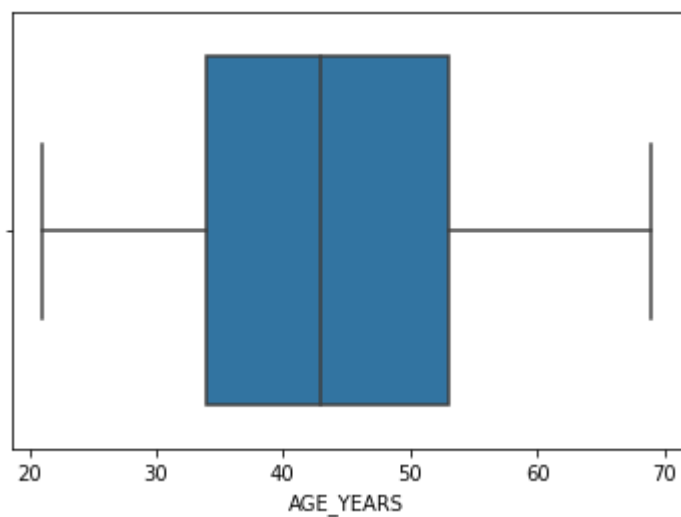
In [20]: for i in data.columns[(data.dtypes == "object").values].tolist():
          print(i, "\n")
          print(data[i].value_counts())
          print("-----")

CODE_GENDER
F    294412
M    144998
Name: CODE_GENDER, dtype: int64
-----
FLAG_Own_CAR
N    275428
Y    163882
Name: FLAG_Own_CAR, dtype: int64
-----
FLAG_Own_REALTY
Y    384843
N    134467
Name: FLAG_Own_REALTY, dtype: int64
-----
NAME_INCOME_TYPE
Working                    226887
Commercial associate      188739
Pensioner                 75483
State servant             36184
Student                   17
Name: NAME_INCOME_TYPE, dtype: int64
-----
NAME_EDUCATION_TYPE
Secondary / secondary special 301789
Higher education             117909
-----
```

- **Removing outliers**

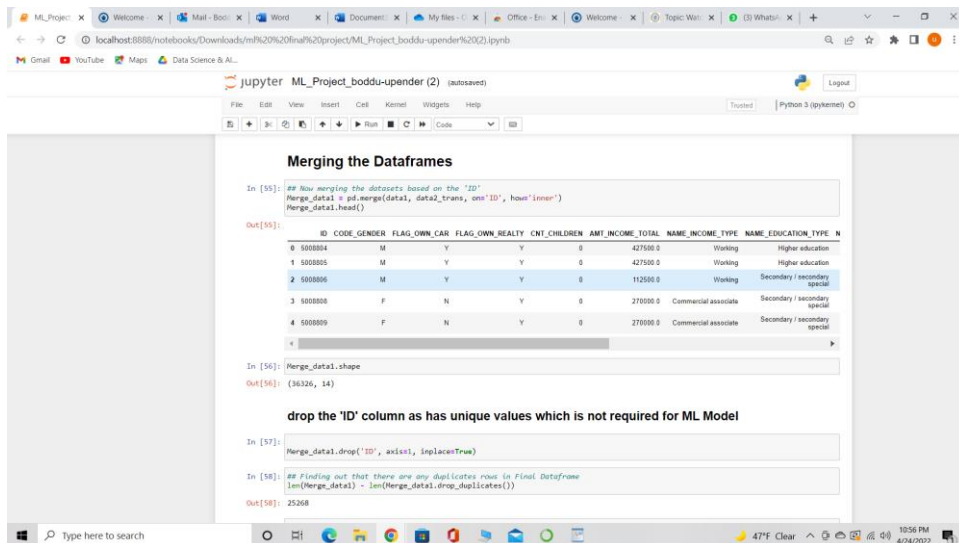






## Merging Data frames

Now we should merge the data frames by dropping nulls and noise values and only taking data which is required so that we can get required and usefull features from that data which can be used for training and testing data and building the ML models.



```
In [55]: ## Now merging the datasets based on the 'ID'
Merge_data1 = pd.merge(data1, data2_train, on='ID', how='inner')
Merge_data1.head()

Out[55]:
```

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCOME_TYPE	NAME_EDUCATION_TYPE	N
0	5008004	M	Y	Y	0	427500.0	Working	Higher education	
1	5008005	M	Y	Y	0	427500.0	Working	Higher education	
2	5008006	M	Y	Y	0	112500.0	Working	Secondary / secondary special	
3	5008008	F	N	Y	0	270000.0	Commercial associate	Secondary / secondary special	
4	5008009	F	N	Y	0	270000.0	Commercial associate	Secondary / secondary special	

```
In [56]: Merge_data1.shape
Out[56]: (36326, 14)

drop the 'ID' column as has unique values which is not required for ML Model

In [57]: Merge_data1.drop('ID', axis=1, inplace=True)

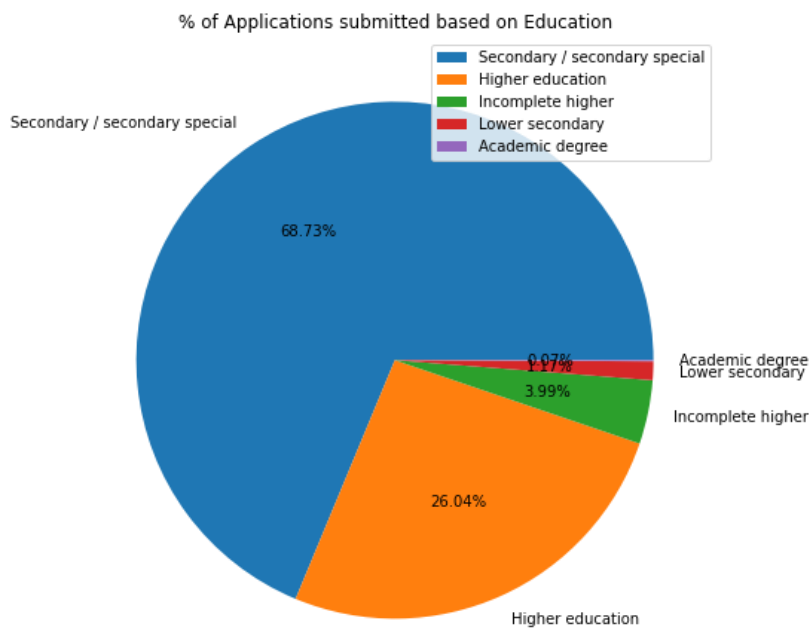
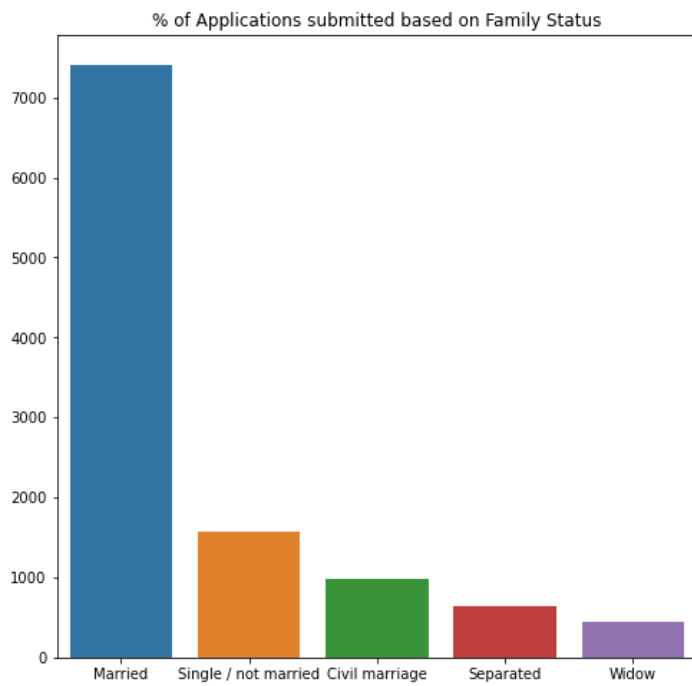
In [58]: ## Finding out that there are any duplicates rows in Final Dataframe
len(Merge_data1) - len(Merge_data1.drop_duplicates())

Out[58]: 25268
```

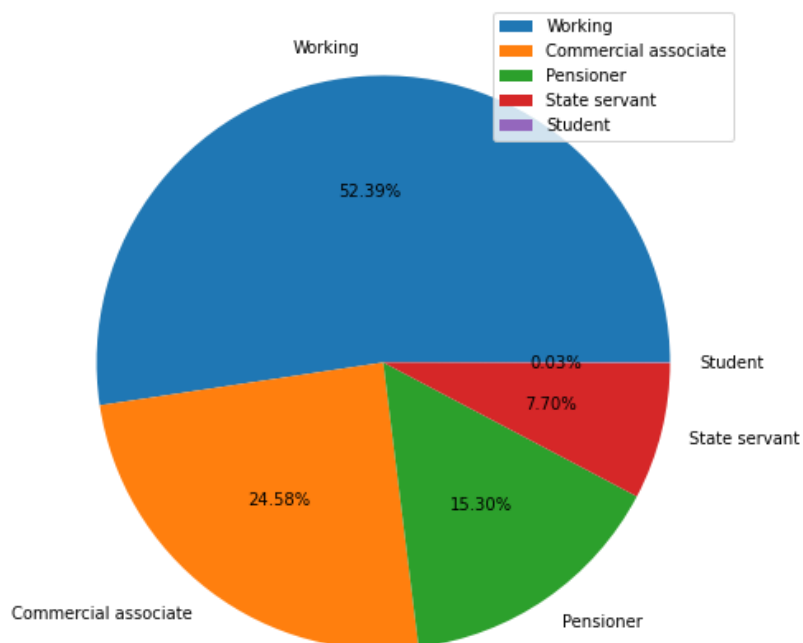
**Drop the 'ID' column as has unique values which is not required for ML Mode**

## Merge Visualization

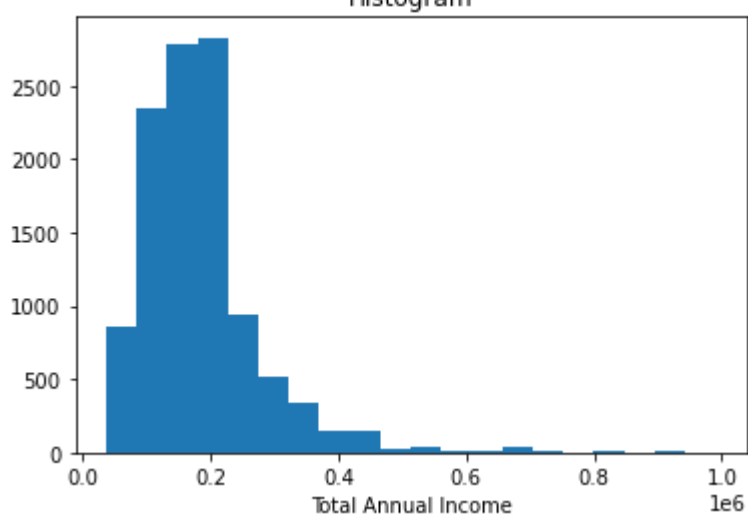
**The below heatmap shows that, there are no such column / Feature which is highly co-related with 'Status'**  
**Wherever there is 1 or light colors, there is high coorelation between the variables**



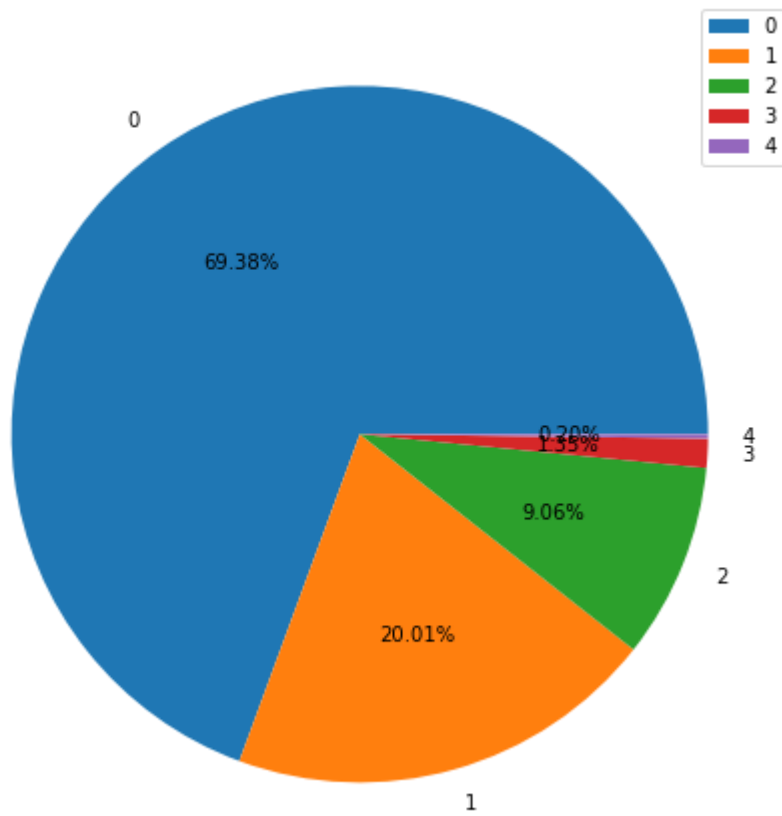
% of Applications submitted based on Income Type



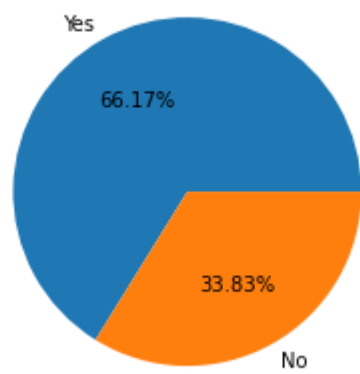
Histogram



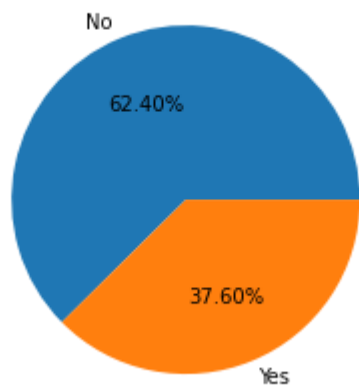
% of Applications submitted based on Children count



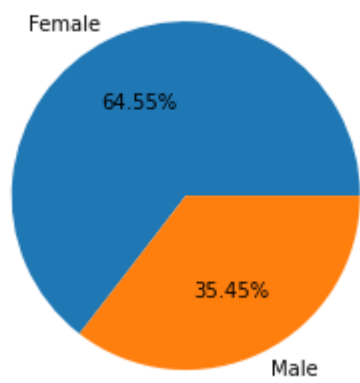
% of Applications submitted based on owning a Real estate property



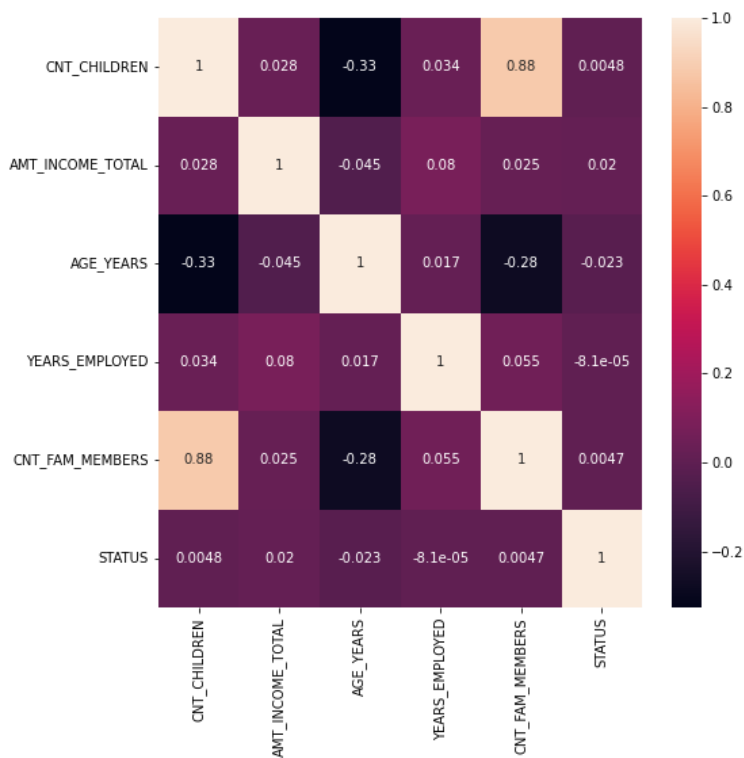
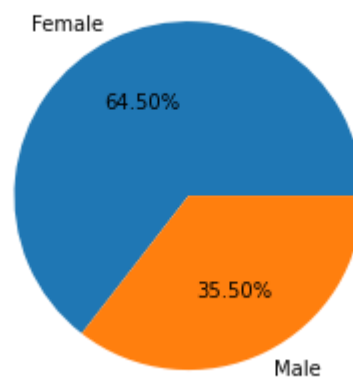
% of Applications submitted based on owning a Car



% of Applications Approved based on Gender



% of Applications submitted based on Gender

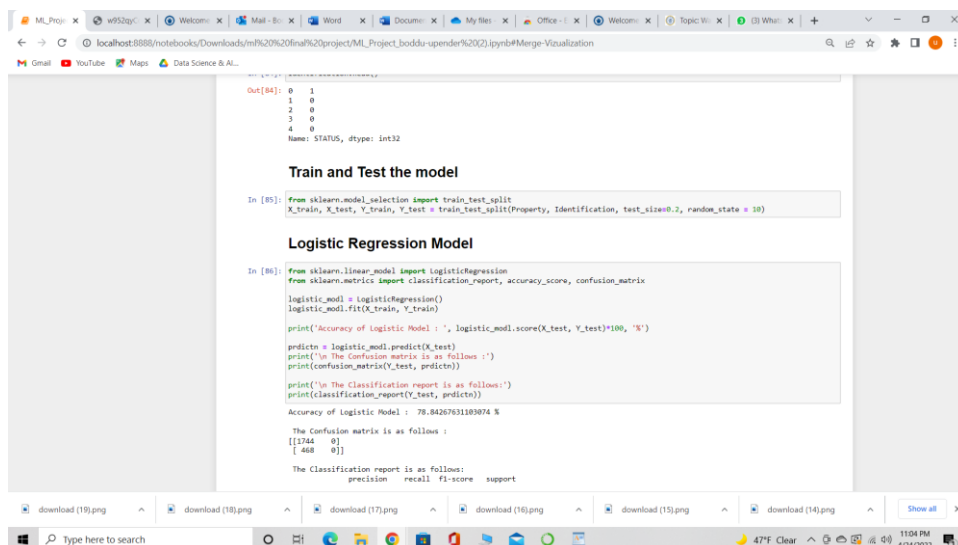


# Feature Selection

```
'CODE_GENDER',  
'FLAG_OWN_CAR',  
'FLAG_OWN_REALTY',  
'NAME_INCOME_TYPE',  
'NAME_EDUCATION_TYPE',  
'NAME_FAMILY_STATUS',  
'NAME_HOUSING_TYPE'
```

Converting all the Non-Numerical Columns into Numerical columns  
Checking for correlation between variables

Then we should train and test the final data which we got from pre-processing by selecting features which plays a crucial role.



The screenshot shows a Jupyter Notebook interface with the following content:

```
Out[84]: 0 1  
1 0  
2 0  
3 0  
4 0  
Name: STATUS, dtype: int32
```

**Train and Test the model**

```
In [85]: from sklearn.model_selection import train_test_split  
X_train, X_test, Y_train, Y_test = train_test_split(Property, Identification, test_size=0.2, random_state = 10)
```

**Logistic Regression Model**

```
In [86]: from sklearn.linear_model import LogisticRegression  
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix  
  
logistic_modl = LogisticRegression()  
logistic_modl.fit(X_train, Y_train)  
  
print('Accuracy of Logistic Model : ', logistic_modl.score(X_test, Y_test)*100, '%')  
  
predictn = logistic_modl.predict(X_test)  
print('\n The Confusion matrix is as follows :')  
print(confusion_matrix(Y_test, predictn))  
  
print('\n The Classification report is as follows:')  
print(classification_report(Y_test, predictn))  
  
Accuracy of Logistic Model : 78.84267631189874 %  
  
The Confusion matrix is as follows :  
[[1744  0]  
 [ 468  0]]  
  
The Classification report is as follows:  
precision    recall  f1-score   support  
  
0.000000      0.000000      0.000000      1744  
0.000000      0.000000      0.000000       468
```

# Modelling

In this project I am modelling 2 different types of algorithms and they are as follows:

- Logistic Regression Model.
- Decision tree classification.

# Deliverables

## Jupyter notebook presentation outlining

- Data cleaning
- Data exploration
- Implementation of machine learning algorithms
- Results of different methods
- Comparison of different methods
- Evaluation methods
- Selection of best methods

## Result

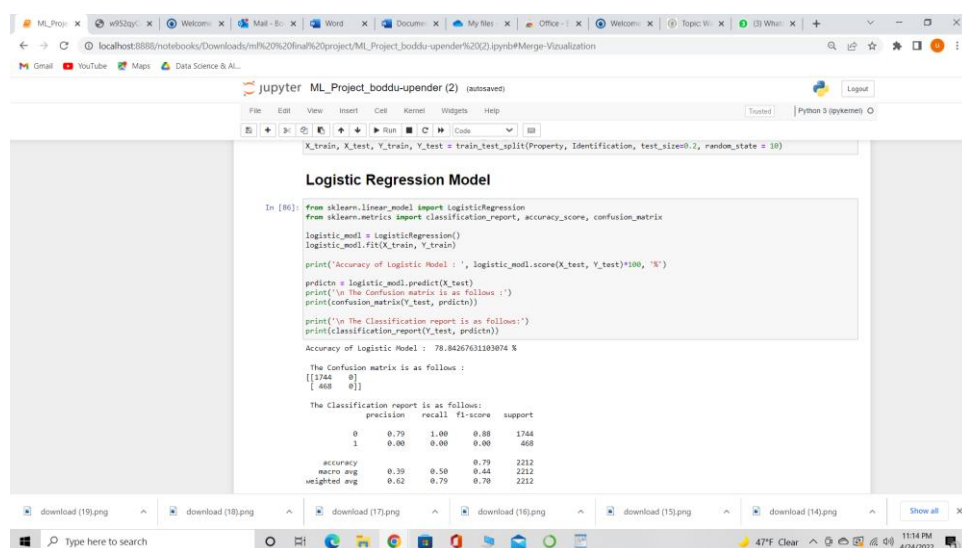
The below are the methods used to evaluate the models.

Test accuracy.

Classification report.

Confusion matrix.

- Logistic Regression Model (78.84%) have more accuracy than Decision tree classification(73.50%).



```
X_train, X_test, Y_train, Y_test = train_test_split(Property, Identification, test_size=0.2, random_state = 10)
```

### Logistic Regression Model

```
In [86]: from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix

logistic_model = LogisticRegression()
logistic_model.fit(X_train, Y_train)

print('Accuracy of Logistic Model : ', logistic_model.score(X_test, Y_test)*100, '%')

predictions = logistic_model.predict(X_test)
print('\n The Confusion matrix is as follows :')
print(confusion_matrix(Y_test, predictions))

print('\n The Classification report is as follows:')
print(classification_report(Y_test, predictions))
```

Accuracy of Logistic Model : 78.8426763180874 %

The Confusion matrix is as follows :

```
[[1744  468]
 [  468  2212]]
```

The Classification report is as follows:

	precision	recall	f1-score	support
0	0.79	1.00	0.88	1744
1	0.00	0.00	0.00	468
accuracy	0.39	0.58	0.79	2212
macro avg	0.39	0.58	0.44	2212
weighted avg	0.62	0.79	0.70	2212