REPORT ON ELIGIBILITY OF LOAN PREDICTION

BY

SALISU IBRAHIM

DATA SCIENTIST APPLICANT INTERN

1. **Introduction**
   In this report, I'm going to predictions customers are eligible for the loan and check whether what are the missing criteria to know why customer not getting loan to make their own house.

Problems we are to trying to solve

- we are trying to know exact issue for loan application
- to find the ways for the applicant to apply the loan

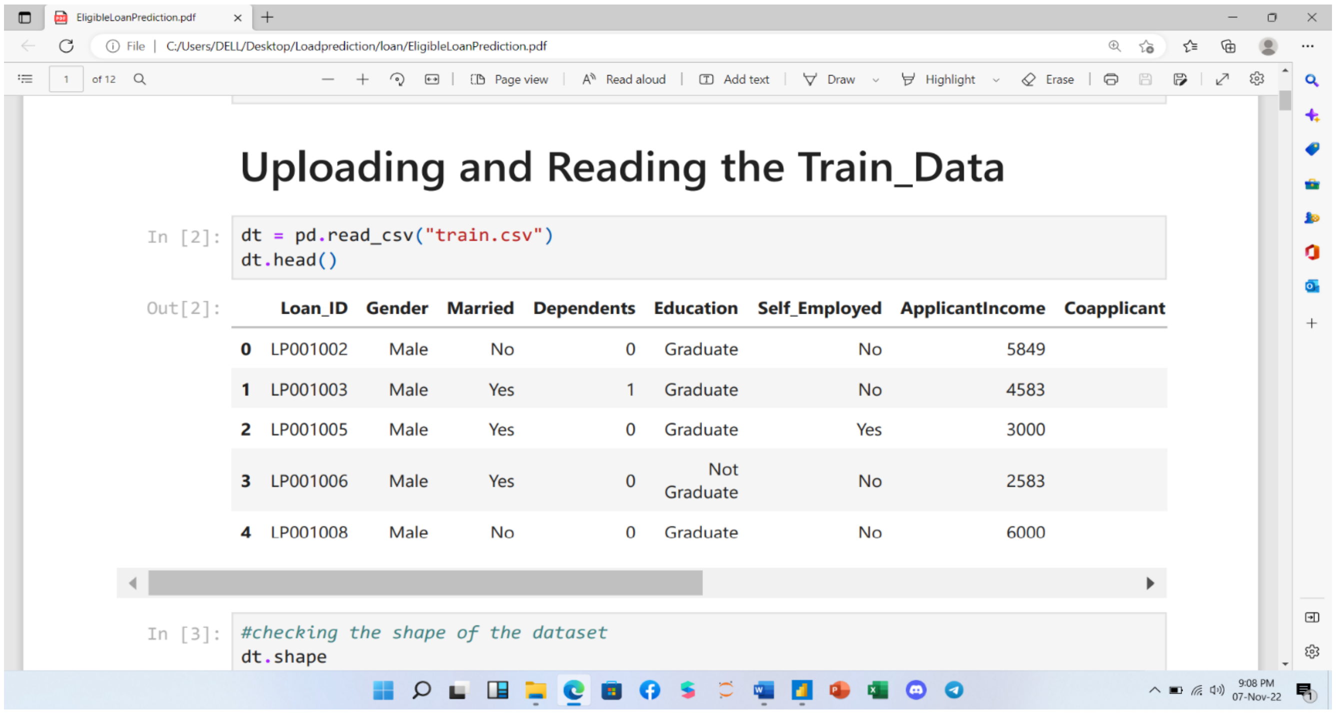
Main Goal

- To create an easier way for the applicant
- Find other and relevant information needed to the applicant before trying to apply for the loan
- Develop a model that can predict the eligibility of the application loan.

About the dataset

The dataset we have consisted of records of different applicant ID, region, loan term, loan status etc. the dataset is so useful which help by understanding the structure of the applicant, this will give us more highlight to make an analysis base on the given information on the dataset to use machine learning algorithm for developing a model to predict the eligibility of the applicant loan.

Below is how the dataset look like,



Uploading and Reading the Train_Data

```
In [2]: dt = pd.read_csv("train.csv")
        dt.head()
```

Out[2]:

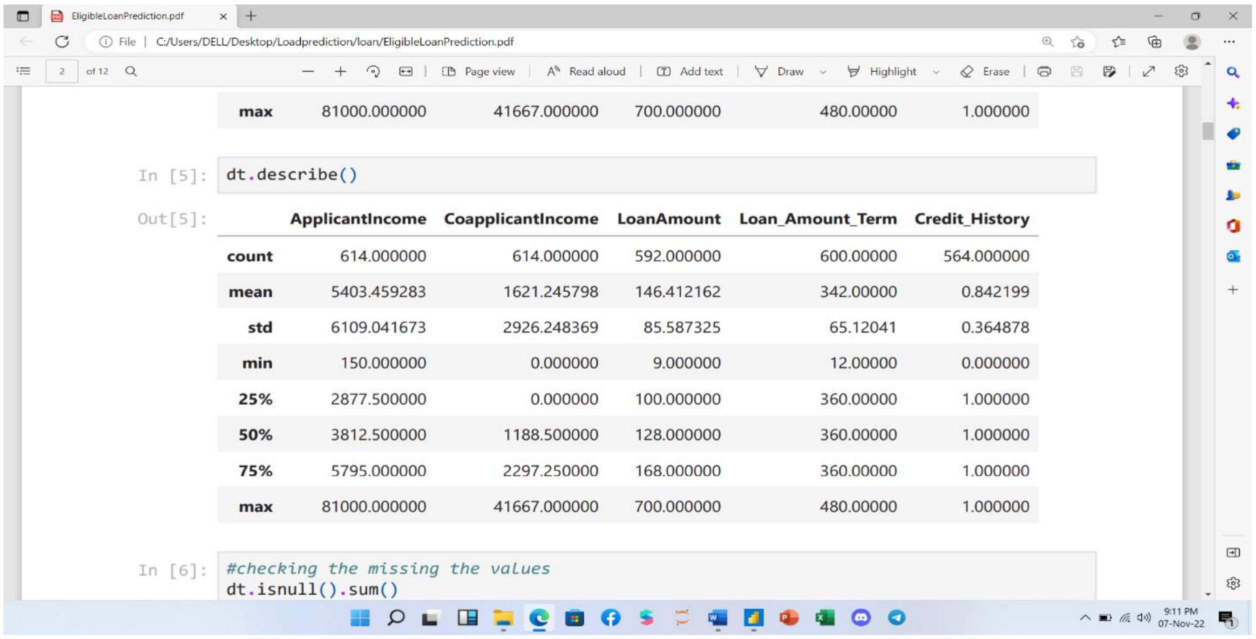
	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplicant
0	LP001002	Male	No	0	Graduate	No	5849	
1	LP001003	Male	Yes	1	Graduate	No	4583	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	
4	LP001008	Male	No	0	Graduate	No	6000	

```
In [3]: #checking the shape of the dataset
        dt.shape
```

2. EDA and Business Implication

Exploratory data analysis (EDA) where we explore our data and grab insights from it. EDA helps us in getting knowledge in form of various plots and diagrams where we can easily understand the data and its features

Analysis of the Data



	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

```
In [5]: dt.describe()
```

Out[5]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

```
In [6]: #checking the missing the values
        dt.isnull().sum()
```

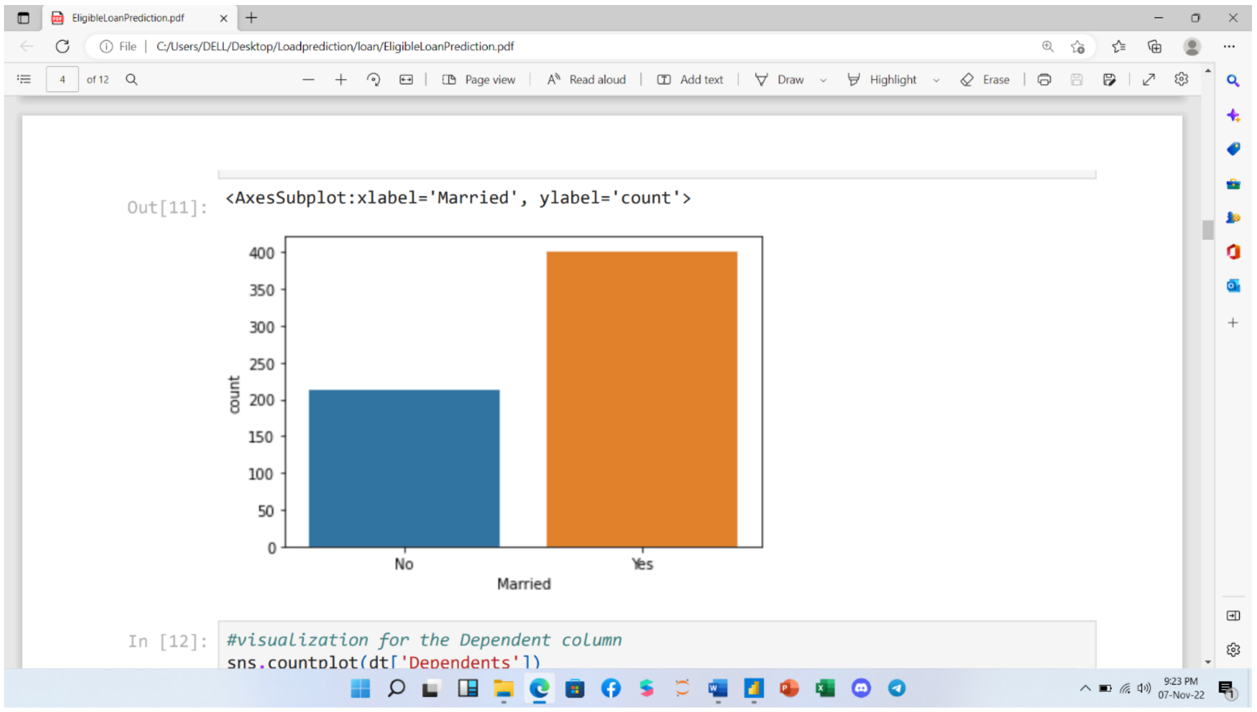
EligibleLoanPrediction.pdf
File | C:/Users/DELL/Desktop/Loadprediction/loan/EligibleLoanPrediction.pdf
2 of 12
Page view
Read aloud
Add text
Draw
Highlight
Erase
9:11 PM
07-Nov-22

Observation

Loan amount: Loan amount is higher than the credit history for the first row first column which is 593 and credit history is 564

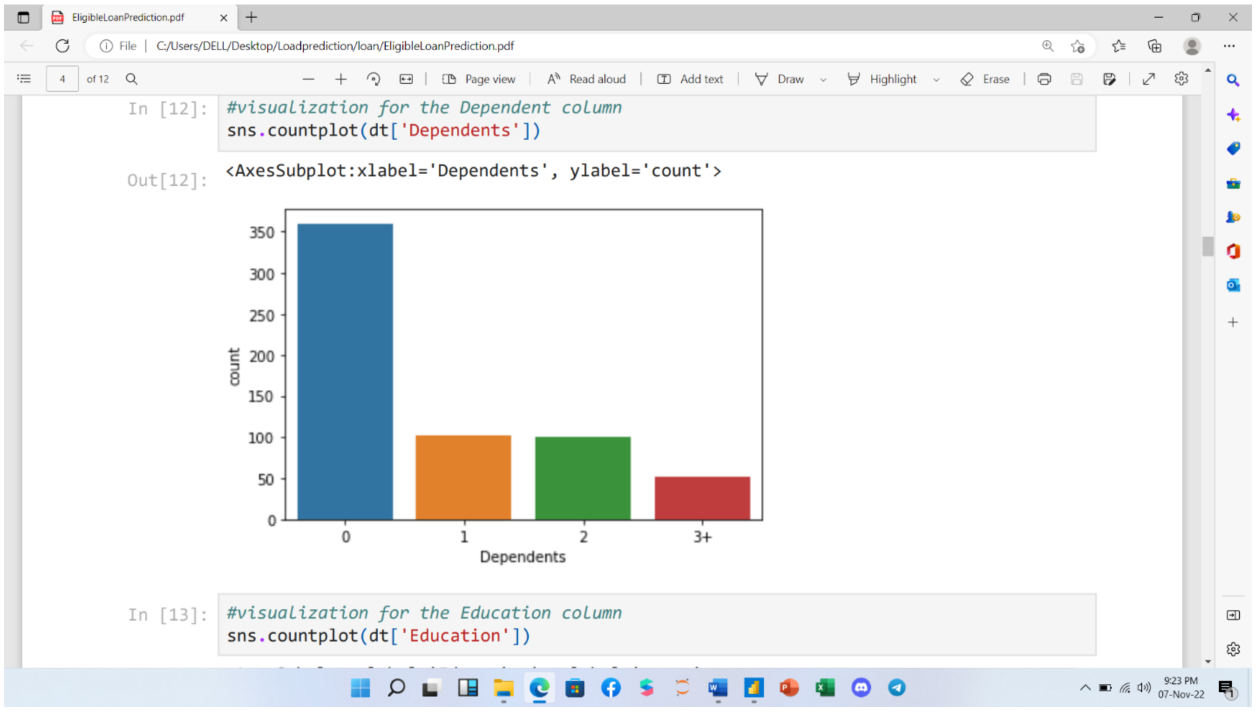
Applicant Income: Applicant income is 614 and loan amount is 592 which is eligible to get a loan

Below is the EDA visualization for the data


EligibleLoanPrediction.pdf
C:/Users/DELL/Desktop/Loadprediction/loan/EligibleLoanPrediction.pdf
4 of 12
Page view
Read aloud
Add text
Draw
Highlight
Erase
9:23 PM
07-Nov-22

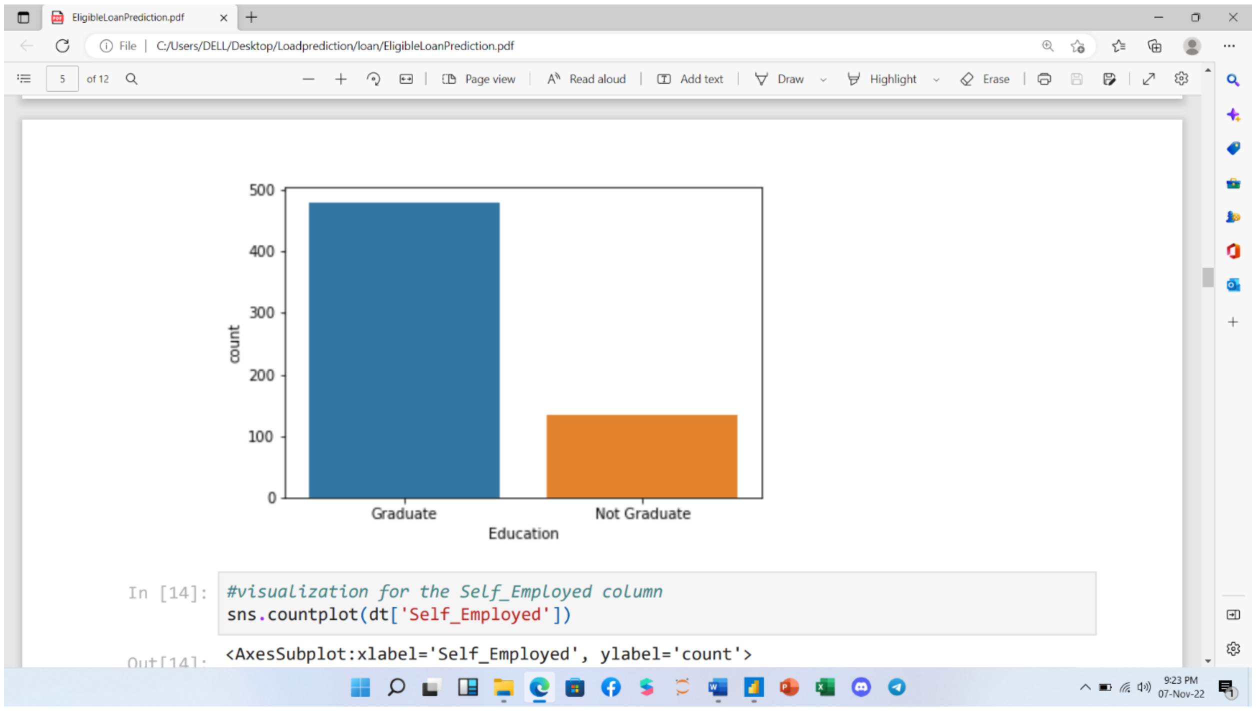
```
Out[11]: <AxesSubplot:xlabel='Married', ylabel='count'>
```

```
In [12]: #visualization for the Dependent column
         sns.countplot(dt['Dependents'])
```

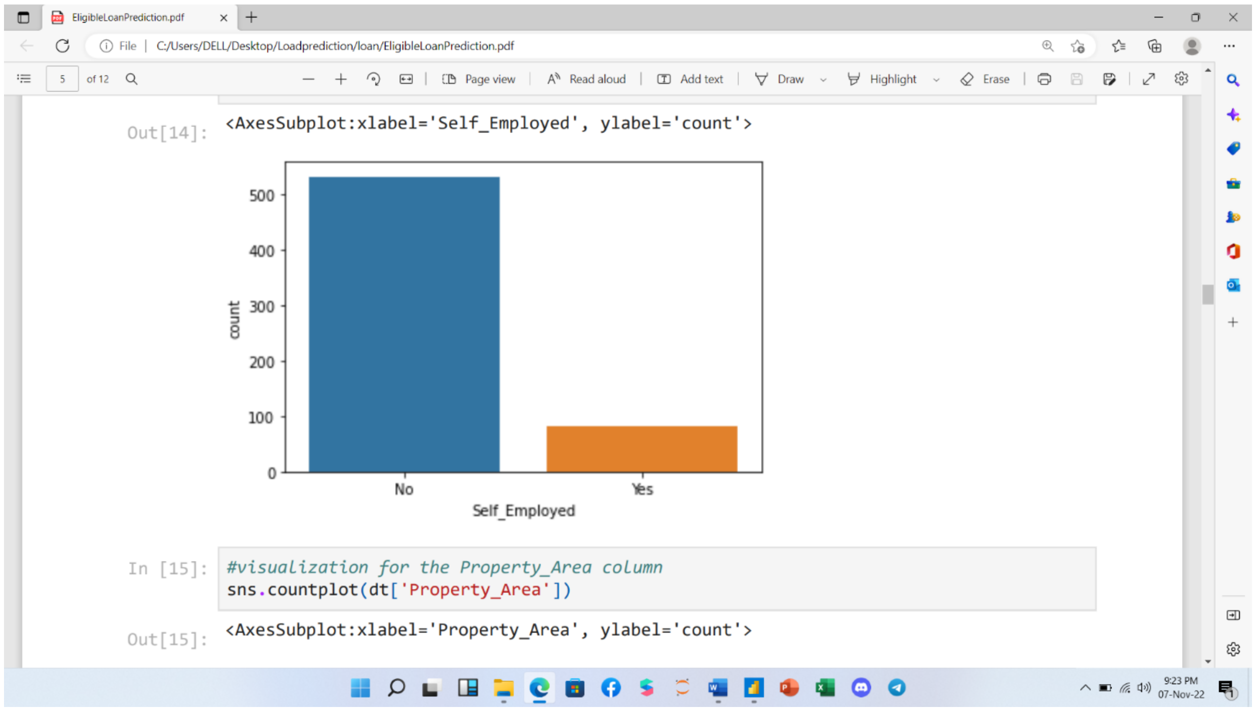

EligibleLoanPrediction.pdf
C:/Users/DELL/Desktop/Loadprediction/loan/EligibleLoanPrediction.pdf
4 of 12
Page view
Read aloud
Add text
Draw
Highlight
Erase
9:23 PM
07-Nov-22

```
In [12]: #visualization for the Dependent column
         sns.countplot(dt['Dependents'])

Out[12]: <AxesSubplot:xlabel='Dependents', ylabel='count'>
```

```
In [13]: #visualization for the Education column
         sns.countplot(dt['Education'])
```

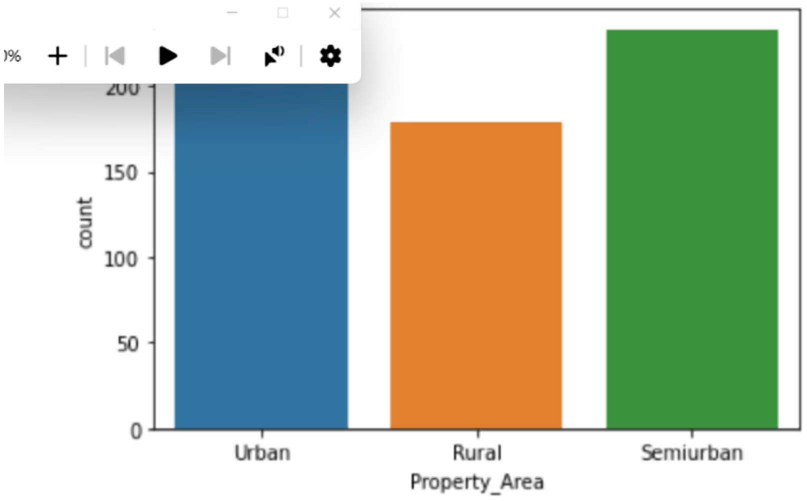


```
In [14]: #visualization for the Self_Employed column
         sns.countplot(dt['Self_Employed'])

Out[14]: <AxesSubplot:xlabel='Self_Employed', ylabel='count'>
```

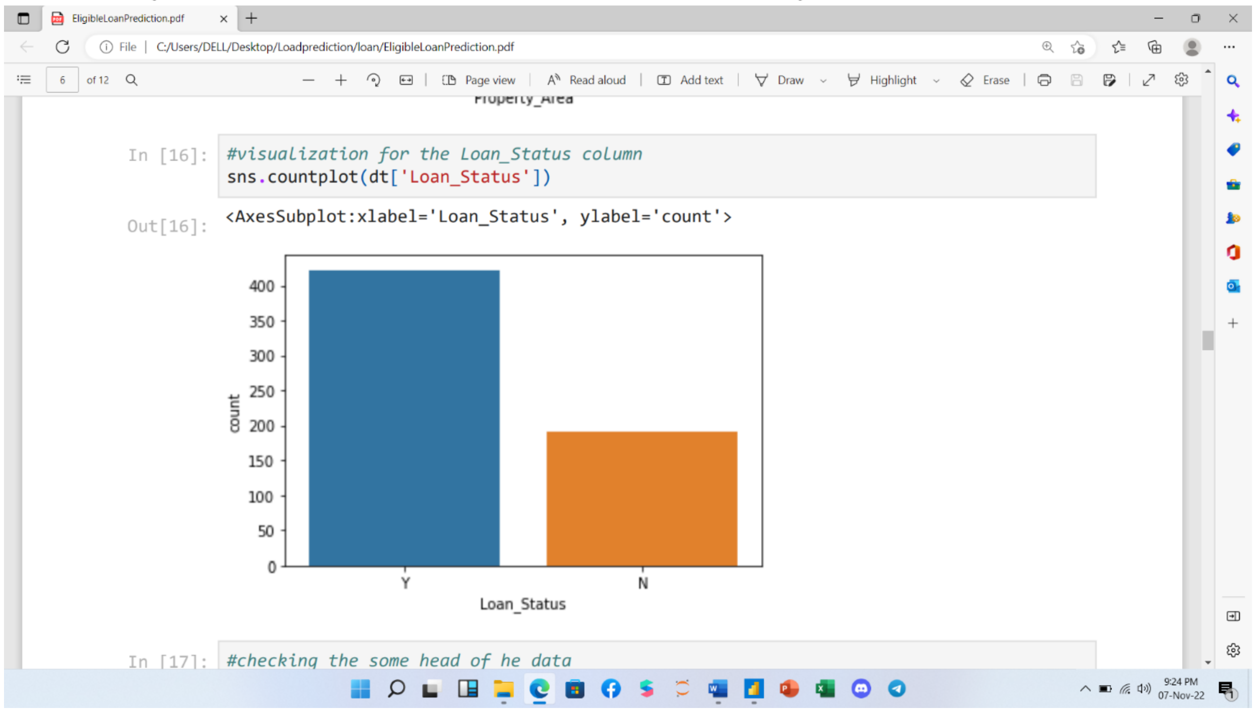


```
Out[14]: <AxesSubplot:xlabel='Self_Employed', ylabel='count'>
```

```
In [15]: #visualization for the Property_Area column
         sns.countplot(dt['Property_Area'])

Out[15]: <AxesSubplot:xlabel='Property_Area', ylabel='count'>
```



```
16]: #visualization for the Loan_Status column
     sns.countplot(dt['Loan_Status'])
16]: <AxesSubplot:xlabel='Loan_Status', ylabel='count'>
```

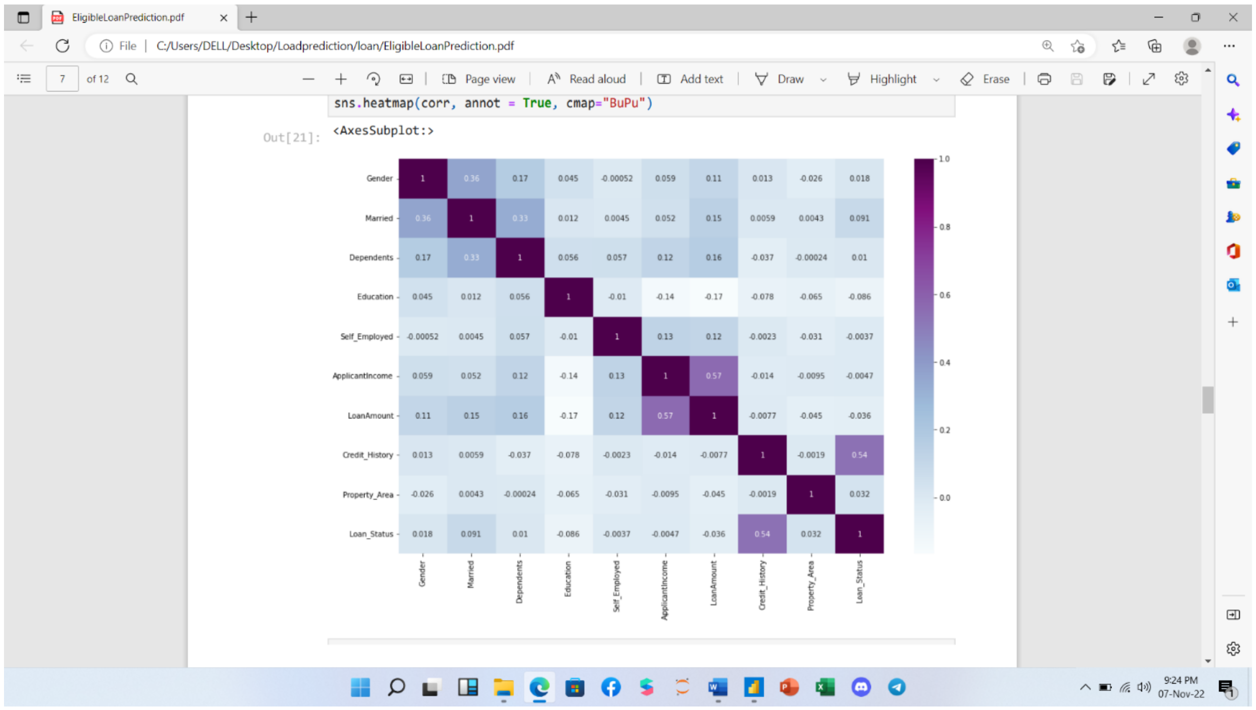


EligibleLoanPrediction.pdf

C:/Users/DELL/Desktop/Loadprediction/loan/EligibleLoanPrediction.pdf

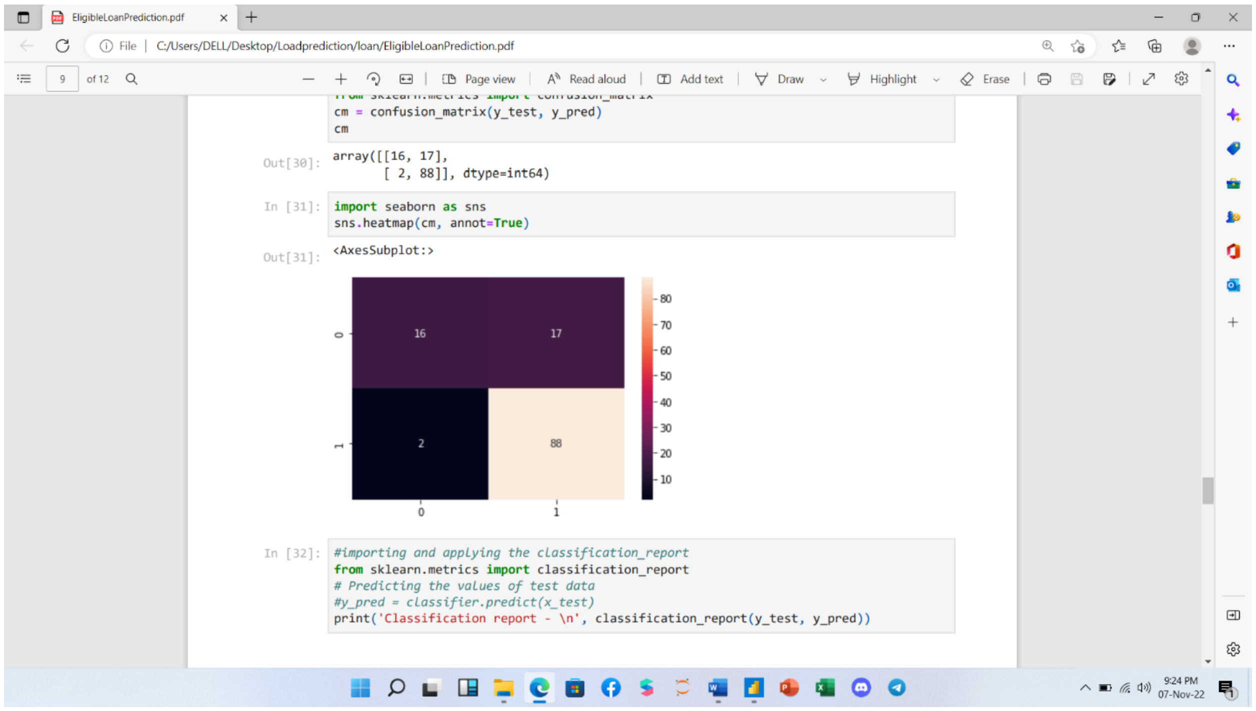
```
In [16]: #visualization for the Loan_Status column
         sns.countplot(dt['Loan_Status'])
Out[16]: <AxesSubplot:xlabel='Loan_Status', ylabel='count'>
```

```
In [17]: #checking the some head of he data
```


EligibleLoanPrediction.pdf
C:/Users/DELL/Desktop/Loadprediction/loan/EligibleLoanPrediction.pdf
7 of 12
Page view
Read aloud
Add text
Draw
Highlight
Erase
Gender 1 0.36 0.17 0.045 -0.00052 0.059 0.11 0.013 -0.026 0.018
Married 0.36 1 0.33 0.012 0.0045 0.052 0.15 0.0059 0.0043 0.091
Dependents 0.17 0.33 1 0.056 0.057 0.12 0.16 -0.037 -0.00024 0.01
Education 0.045 0.012 0.056 1 -0.01 -0.14 -0.17 -0.078 -0.065 -0.086
Self_Employed -0.00052 0.0045 0.057 -0.01 1 0.13 0.12 -0.0023 -0.031 -0.0037
ApplicantIncome 0.059 0.052 0.12 -0.14 0.13 1 0.57 -0.014 -0.0095 -0.0047
LoanAmount 0.11 0.15 0.16 -0.17 0.12 0.57 1 -0.0077 -0.045 -0.036
Credit_History 0.013 0.0059 -0.037 -0.078 -0.0023 -0.014 -0.0077 1 -0.0019 0.54
Property_Area -0.026 0.0043 -0.00024 -0.065 -0.031 -0.0095 -0.045 -0.0019 1 0.032
Loan_Status 0.018 0.091 0.01 -0.086 -0.0037 -0.0047 -0.036 0.54 0.032 1
Gender
Married
Dependents
Education
Self_Employed
ApplicantIncome
LoanAmount
Credit_History
Property_Area
Loan_Status
1.0
0.8
0.6
0.4
0.2
0.0
9:24 PM
07-Nov-22

```
sns.heatmap(corr, annot = True, cmap="BuPu")
```

```
Out[21]: <AxesSubplot:>
```



```
cm = confusion_matrix(y_test, y_pred)
cm
```

```
Out[30]: array([[16, 17],
                [ 2, 88]], dtype=int64)
```

```
In [31]: import seaborn as sns
         sns.heatmap(cm, annot=True)
```

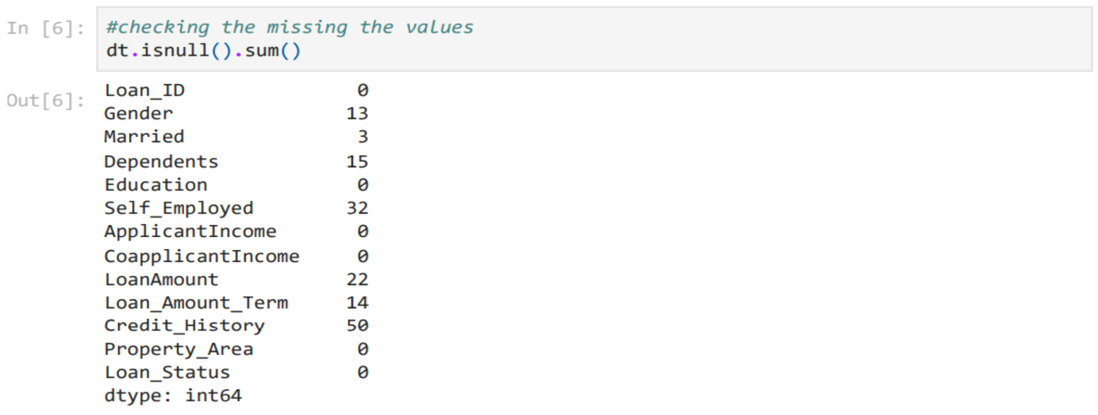
```
Out[31]: <AxesSubplot:>
```

```
In [32]: #importing and applying the classification_report
         from sklearn.metrics import classification_report
         # Predicting the values of test data
         #y_pred = classifier.predict(x_test)
         print('Classification report - \n', classification_report(y_test, y_pred))
```

3. Data Cleaning and Pre-Processing

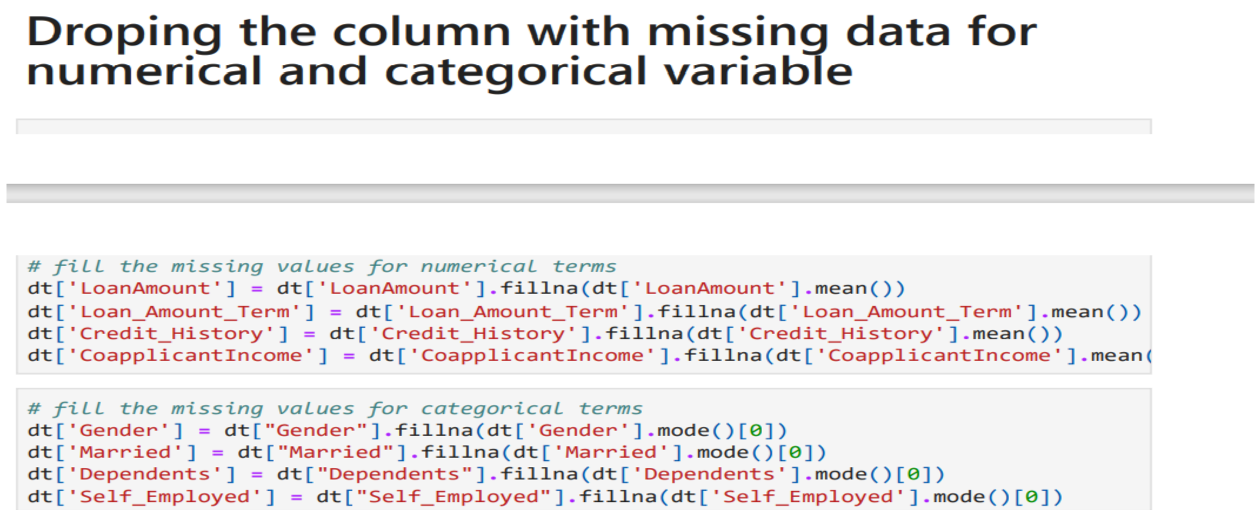
Data cleaning is the most important part of any data analysis which will help us to identify the area where we have the missing values in order to replace or remove unwanted column from our data so that to have good analysis and model prediction.

The method/approach used for identifying and treating the missing values and outlier treatment:-



```
In [6]: #checking the missing the values
        dt.isnull().sum()

Out[6]: Loan_ID              0
        Gender              13
        Married              3
        Dependents          15
        Education            0
        Self_Employed       32
        ApplicantIncome      0
        CoapplicantIncome    0
        LoanAmount          22
        Loan_Amount_Term    14
        Credit_History      50
        Property_Area        0
        Loan_Status          0
        dtype: int64
```

To identify any missing values in our data set we have used **Pandas** pre built function **isnull()** to detect any missing values in our datasets. From the picture we can see that the column credit history has the highest number of missing values which is (50) followed by self employed with 32 missing values and married column with lowest missing values. The next step is how to handle those missing values either to manipulate it or even delete the column if we don’t want it.



Droping the column with missing data for numerical and categorical variable

```
# fill the missing values for numerical terms
dt['LoanAmount'] = dt['LoanAmount'].fillna(dt['LoanAmount'].mean())
dt['Loan_Amount_Term'] = dt['Loan_Amount_Term'].fillna(dt['Loan_Amount_Term'].mean())
dt['Credit_History'] = dt['Credit_History'].fillna(dt['Credit_History'].mean())
dt['CoapplicantIncome'] = dt['CoapplicantIncome'].fillna(dt['CoapplicantIncome'].mean(
```

```
# fill the missing values for categorical terms
dt['Gender'] = dt["Gender"].fillna(dt['Gender'].mode()[0])
dt['Married'] = dt["Married"].fillna(dt['Married'].mode()[0])
dt['Dependents'] = dt["Dependents"].fillna(dt['Dependents'].mode()[0])
dt['Self_Employed'] = dt["Self_Employed"].fillna(dt['Self_Employed'].mode()[0])
```

Above are the picture for the step we followed for filling the missing values from our data.

4. Modeling Building

Model Selection and Why?

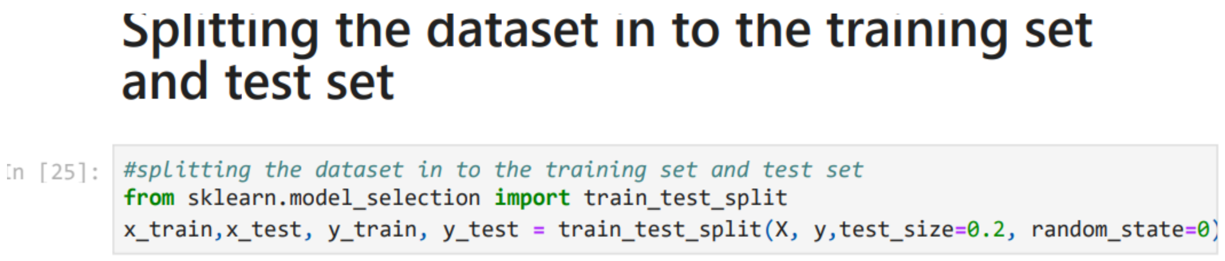
After cleaning and processing the data then comes the modeling part which includes building Machine Learning models, let's first understand in brief what Machine Learning is?

Machine Learning is a technique that analyzes past data and tries to extract meaningful insights and patterns from them which can be further used to perform predictions in future. For example, classifying whether a tumor is benign or malignant, predicting stock prices, etc. One such application which we're using right here is predicting house prices. Before making predictions first we need to build a model and train it using past data.

First, we need to separate the dataset into two parts: features (property attributes) and labels (prices) which is the required format for any model to be trained on.

Then the data needs to be split into 3 sets

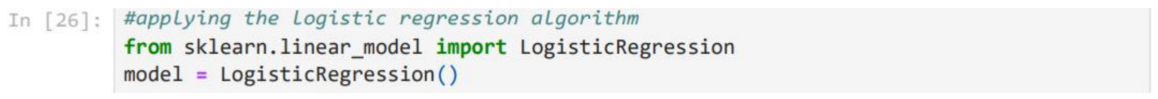
1. Training set - This will be the part of the dataset which the model will be using to train itself, the size should be at least 60-70% of the total data we've.
2. Validation set - This set is used for validating our model's performance for a different set of hyperparameters. After taking out the train set, the remaining set can be split into validation and test set.
3. Testing set - To evaluate how the model is performing on the unseen data on which the model will be doing future predictions on, test set is used. It helps to understand how much error is there between actual and predicted values.



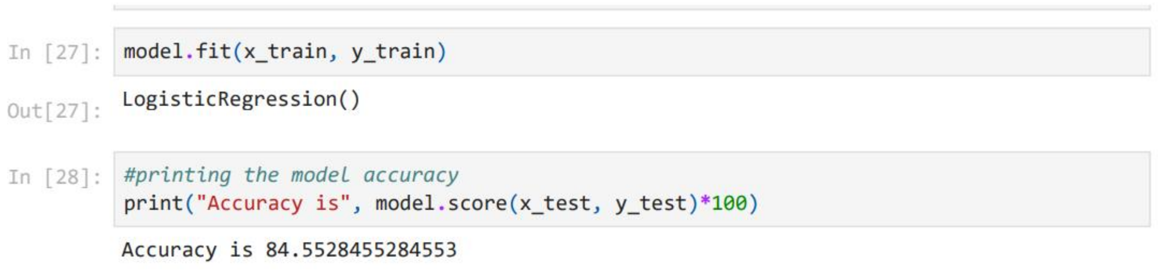
Splitting the dataset in to the training set and test set

```
In [25]: #splitting the dataset in to the training set and test set
         from sklearn.model_selection import train_test_split
         x_train,x_test, y_train, y_test = train_test_split(X, y,test_size=0.2, random_state=0)
```

Then we're to try applying different algorithm for training our model, then using the validation set we can determine which model to keep for making final predictions.

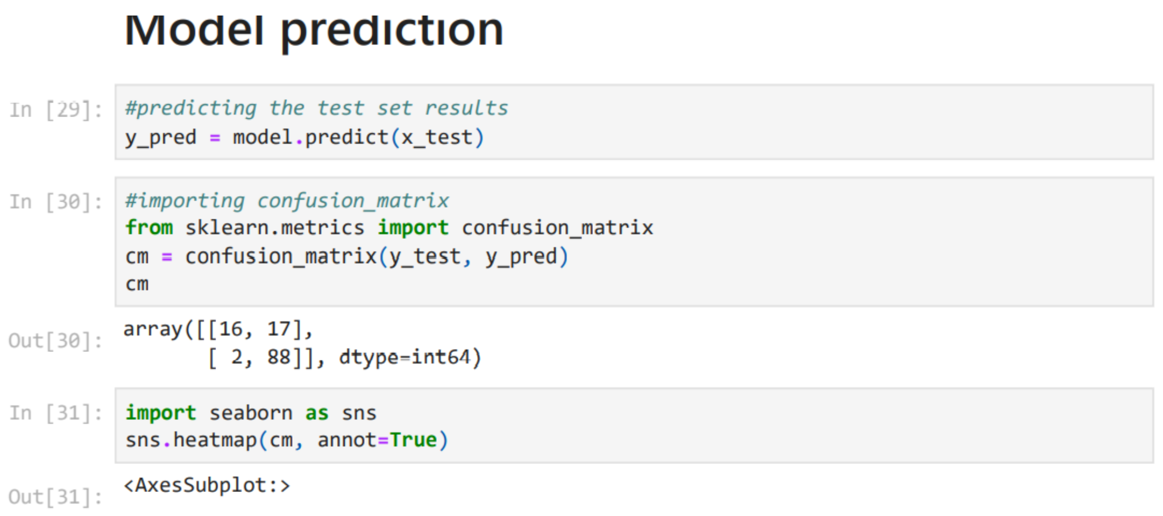


```
In [26]: #applying the logistic regression algorithm
         from sklearn.linear_model import LogisticRegression
         model = LogisticRegression()
```



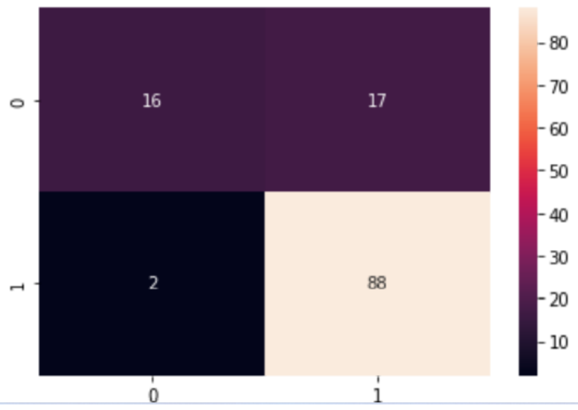
```
In [27]: model.fit(x_train, y_train)

Out[27]: LogisticRegression()

In [28]: #printing the model accuracy
         print("Accuracy is", model.score(x_test, y_test)*100)

         Accuracy is 84.5528455284553
```

Model Prediction



Model prediction

```
In [29]: #predicting the test set results
         y_pred = model.predict(x_test)

In [30]: #importing confusion_matrix
         from sklearn.metrics import confusion_matrix
         cm = confusion_matrix(y_test, y_pred)
         cm

Out[30]: array([[16, 17],
                [ 2, 88]], dtype=int64)

In [31]: import seaborn as sns
         sns.heatmap(cm, annot=True)

Out[31]: <AxesSubplot:>
```



Final Words:

- By comparison the logistic regression performs more than the 2 algorithm we used in this model
- We also understand that the graduate applicant are more request for loan than non-graduate.