Smart Lender - Applicant Credibility Prediction For Loan Approval Using Machine Learning

Category: Machine Learning

Project Description:

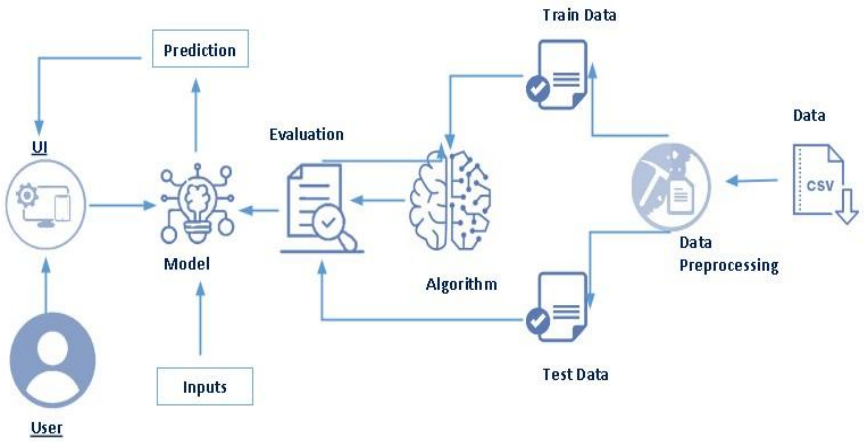
introduction:

One of the most important factors which affect our country's economy and financial condition is the credit system governed by the banks. The process of bank credit risk evaluation is recognized at banks across the globe. "As we know credit risk evaluation is very crucial, there is a variety of techniques are used for risk level calculation. In addition, credit risk is one of the main functions of the banking community.

The prediction of credit defaulters is one of the difficult tasks for any bank. But by forecasting the loan defaulters, the banks definitely may reduce their loss by reducing their non-profit assets, so that recovery of approved loans can take place without any loss and it can play as the contributing parameter of the bank statement. This makes the study of this loan approval prediction important. Machine Learning techniques are very crucial and useful in the predictionof these types of data.

We will be using classification algorithms such as Decision tree, Random forest, KNN, and xgboost. We will train and test the data with these algorithms. From this best model is selected and saved in pkl format. We will be doing flask integration and IBM deployment.

Technical Architecture:



Pre-Requisites:

To complete this project, you must require the following software, concepts, and packages

Python packages:

- Open anaconda prompt as administrator
- Type "pip install numpy" and click enter.
- Type "pip install pandas" and click enter.
- Type "pip install scikit-learn" and click enter.
- Type "pip install matplotlib" and click enter.
- Type "pip install pickle-mixin" and click enter.
- Type "pip install seaborn" and click enter.
- Type "pip install Flask" and click enter.

Prior Knowledge

You must have prior knowledge of the following topics to complete this project.

- ML Concepts

Supervised learning

Project Objectives

Write what are all the technical aspects that students would get if they complete this project.

1. Knowledge of Machine Learning Algorithms.
2. Knowledge of Python Language with Machine Learning
3. You'll be able to understand the problem to classify if it is a regression or a classification kind of problem.
4. You will be able to know how to pre-process/clean the data using different data pre-processing techniques.
5. Applying different algorithms according to the dataset and based on visualization.
6. Real-Time Analysis of Project
7. Building ease of User Interface (UI)
8. Navigation of ideas towards other projects(creativeness)
9. Knowledge of building ML models.
10. How to build web applications using the Flask framework.

Project Flow

Install Required Libraries.
Data Collection.

- Collect the dataset or Create the dataset

Data Preprocessing.

- Import the Libraries.
- Importing the dataset.
- Understanding Data Type and Summary of features.
- Take care of missing data
- Data Visualization.
- Drop the column from DataFrame & replace the missing value.

- Splitting the Dataset into Dependent and Independent variables
- Splitting Data into Train and Test.

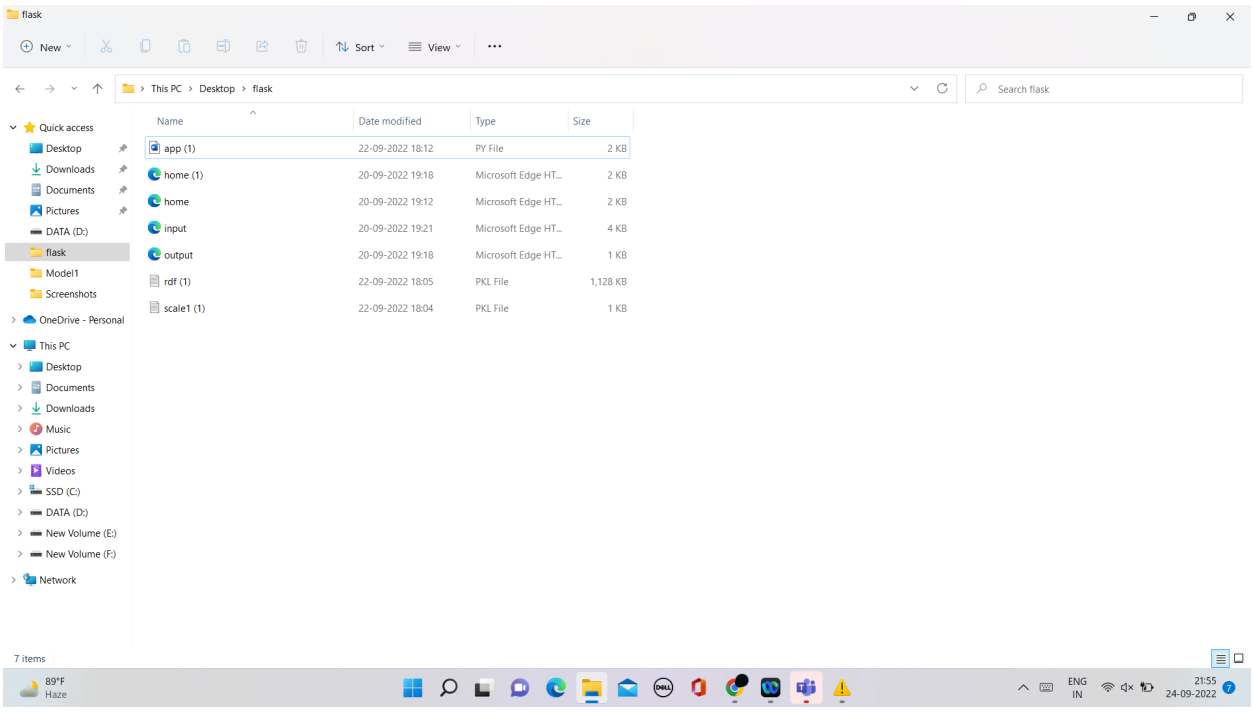
Model Building
- Training and testing the model
- Evaluation of Model
- Saving the Model

Application Building
- Create an HTML file
- Build a Python Code

Final UI
- Dashboard Of the flask app.

Project Structure

Create a Project folder that contains files as shown below



- We are building a Flask Application that needs HTML pages “home.html”, “index.html” stored in the templates folder and a python script app.py for server-side scripting
- The model is built in the notebook floods.ipynb
- We need the model which is saved and the saved model in this content is floods.save and transform.save
- The templates mainly used here are “home.html”, ”chance.html”, ”no chance.html”,

"index.html" for showcasing the UI

- The flask app is denoted as app.py
- IBM scoring endpoint consists of templates and app,py

Data Collection

ML depends heavily on data, without data, it is impossible for an "AI" to learn. It is the most crucial aspect that makes algorithm training possible. In Machine Learning projects, we need a training data set. It is the actual data set used to train the model for performing various actions.

Download The Dataset:

Download the dataset from the below link.

- You can collect datasets from different open sources like kaggle.com, data.gov, UCI machine learning repository, etc.
- Here we are using a data set which you can find in the below link and you can download it from the link: Link

Visualizing And Analyzing The Data

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualization techniques and some analyzing techniques.
Note: There is n number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

Importing The Libraries

Import the necessary libraries as shown in the image

Import the required libraries for the model to run. The first step is usually importing the libraries that will be needed in the program.

Import the necessary libraries as shown in the image

Import the required libraries for the model to run. The first step is usually importing the libraries that will be needed in the program.

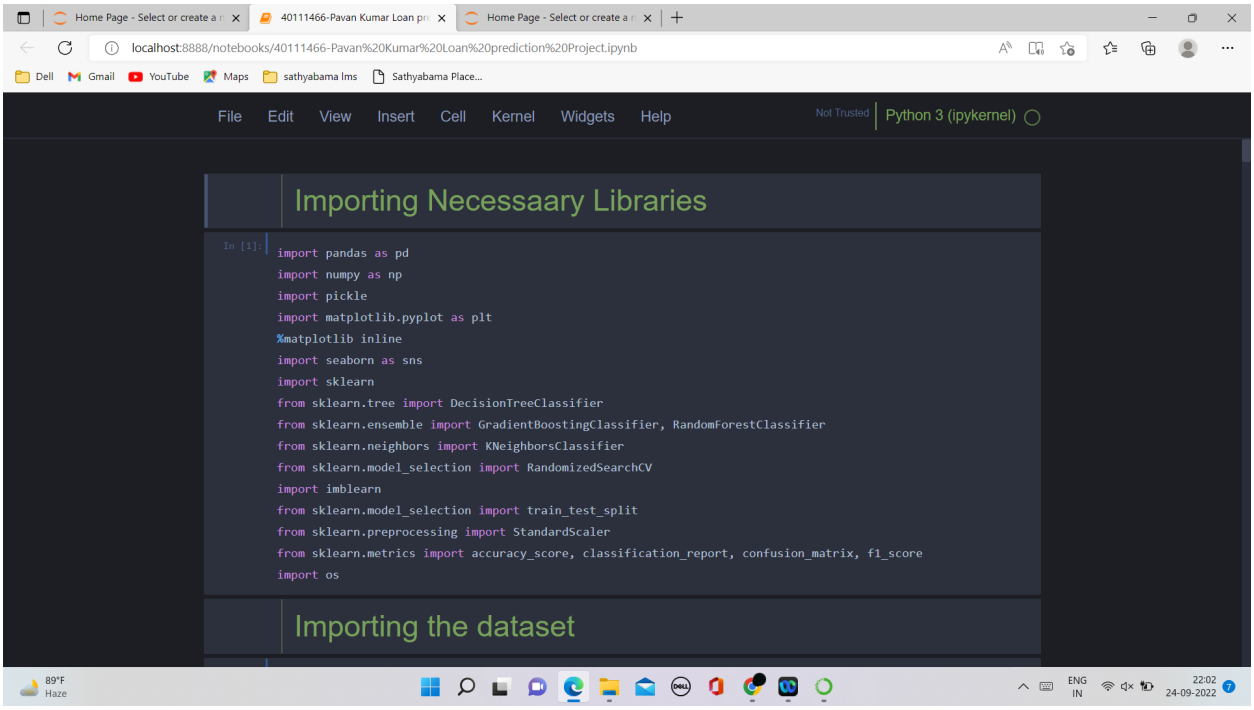
- **Numpy**- It is an open-source numerical Python library. It contains a multi-dimensional array and matrix data structures. It can be used to perform mathematical operations on arrays such as trigonometric, statistical, and algebraic routines.

- **Pandas-** It is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

- **Seaborn**- Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

- **Matplotlib**- Visualisation with python. It is a comprehensive library for creating static,animated, and interactive visualizations in Python

Sklearn – which contains all the modules required for model building

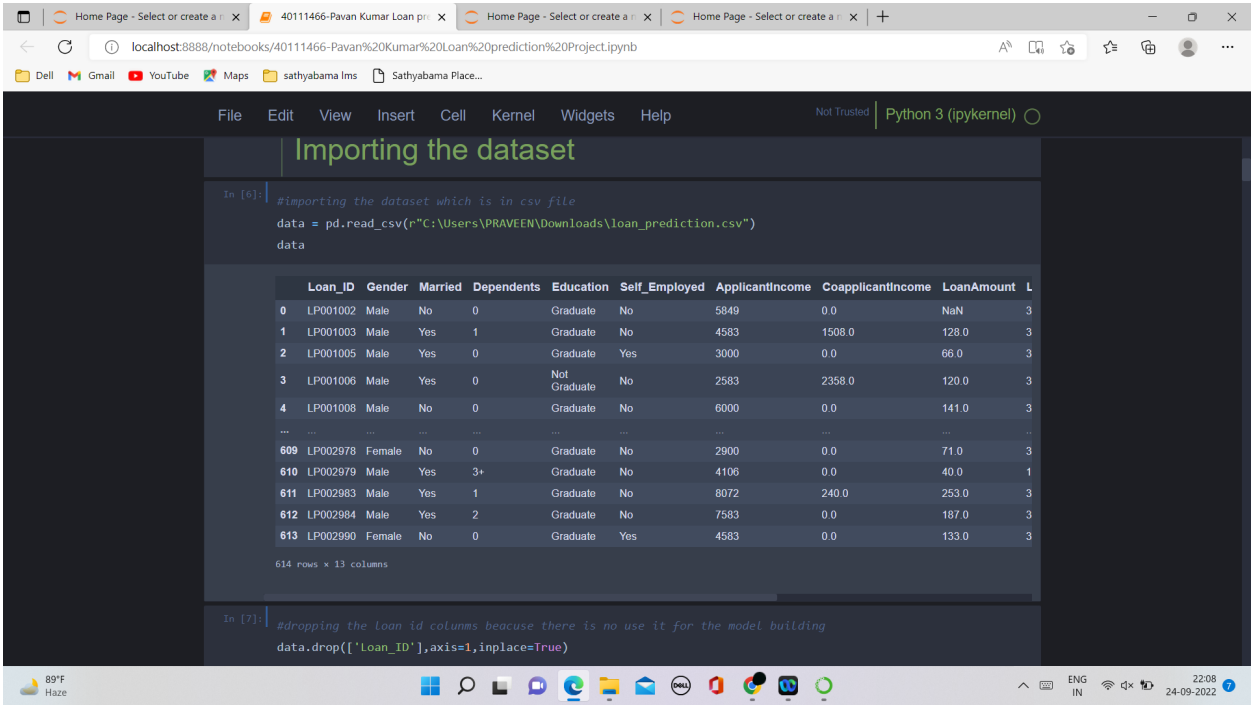


Reading The Dataset

- Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the

dataset with the help of pandas.

In pandas, we have a function called read_csv() to read the dataset. As a parameter, we have to give the directory of the CSV file



Importing The Dataset

- You might have your data in .csv files, .excel files
- Let's load a .csv data file into pandas usingread_csv() function.We will need to locate the directory of the CSV file at first (it's more efficient to keep the dataset in the same directory as your program).
- If your dataset is in some other location, Then
- **Data=pd.read_csv(r"File_location/datasetname.csv")**
- If the dataset is in the same directory of your program, you can directly read it, without giving raw as r.
- Our Dataset weatherAus.csv contains the following Columns

- Holiday - working day or holiday
- Temp- temperature of the day
- Rain and snow – whether it is raining or snowing on that day or not
- Weather = describes the weather conditions of the day
- Date and time = represents the exact date and time of the day
- Traffic volume – output column

The output column to be predicted is Traffic volume.Based on the input variables we predict the volume of the traffic.The predicted output gives them a fair idea of the count of traffic

Analyse The Data

head()method is used to return top n (5 by default) rows of a DataFrame or series.

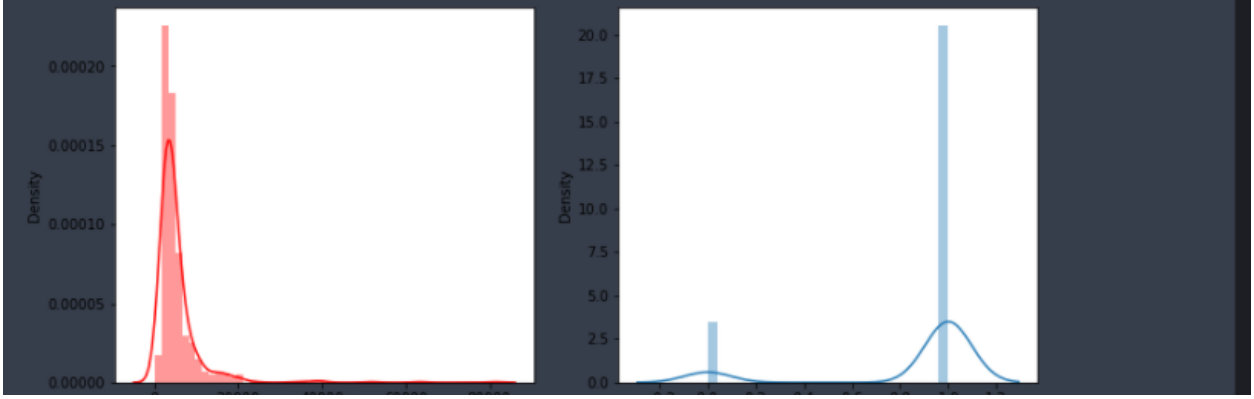
Uni-Variate Analysis

In simple words, univariate analysis is understanding the data with single feature. Here we have displayed two different graphs such as distplot and countplot.

- Seaborn package provides a wonderful function distplot. With the help of distplot, we can find the distribution of the feature. To make multiple graphs in a single plot, we use a subplot.

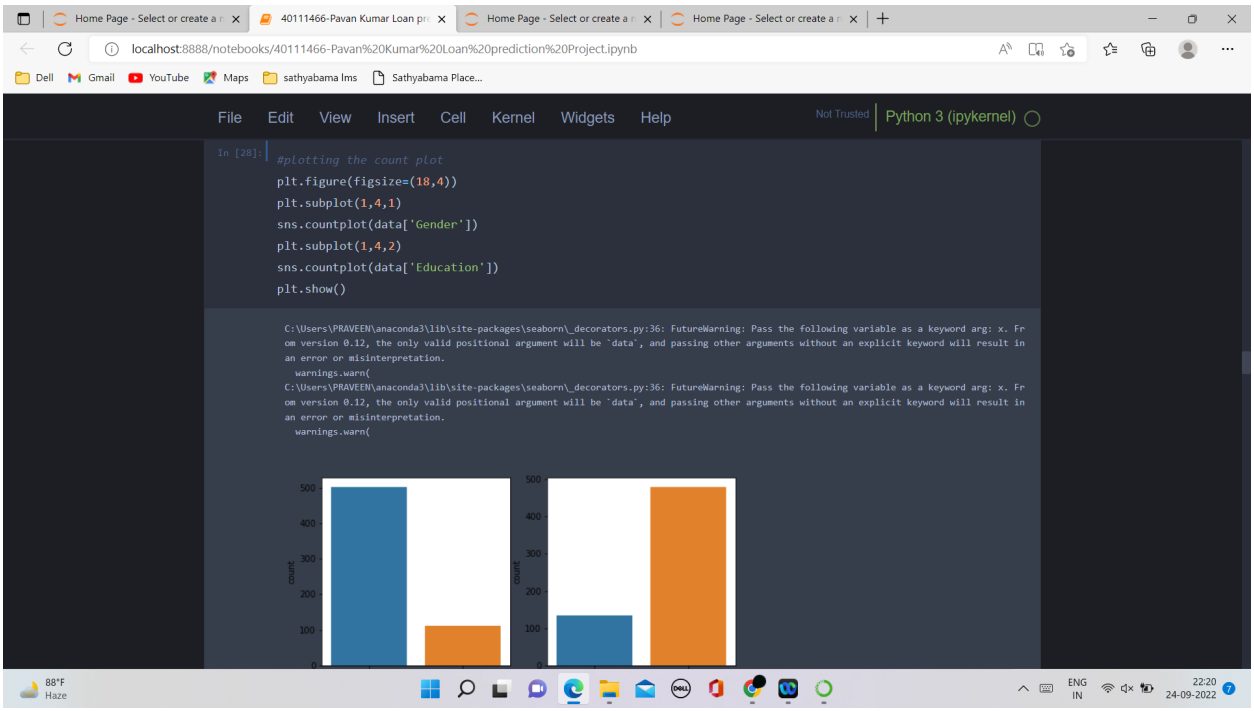
```
#plotting the using distplot
plt.figure(figsize=(12,5))
plt.subplot(121)
sns.distplot(data['ApplicantIncome'], color='r')
plt.subplot(122)
sns.distplot(data['Credit_History'])
plt.show()
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be remo
ved in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot`
(an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\HP\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be remo
ved in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot`
(an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



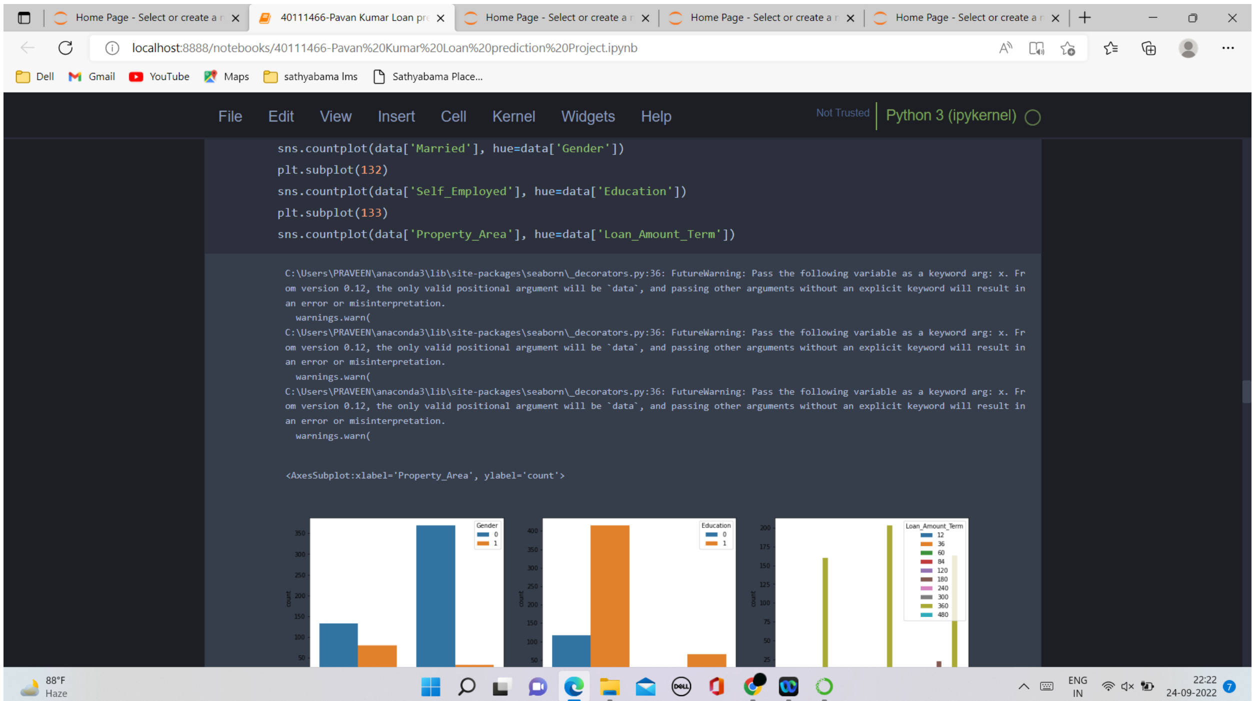
- In our dataset, we have some categorical features. With the count plot function, we are going to count the unique category in those features. We have created a dummy data frame with categorical features. With for loop and subplot, we have plotted the below graph.
- From the plot we came to know, Applicants' income is skewed towards the left side, whereas credit history is categorical with 1.0 and 0.0

Bivariate Analysis



Countplot:-

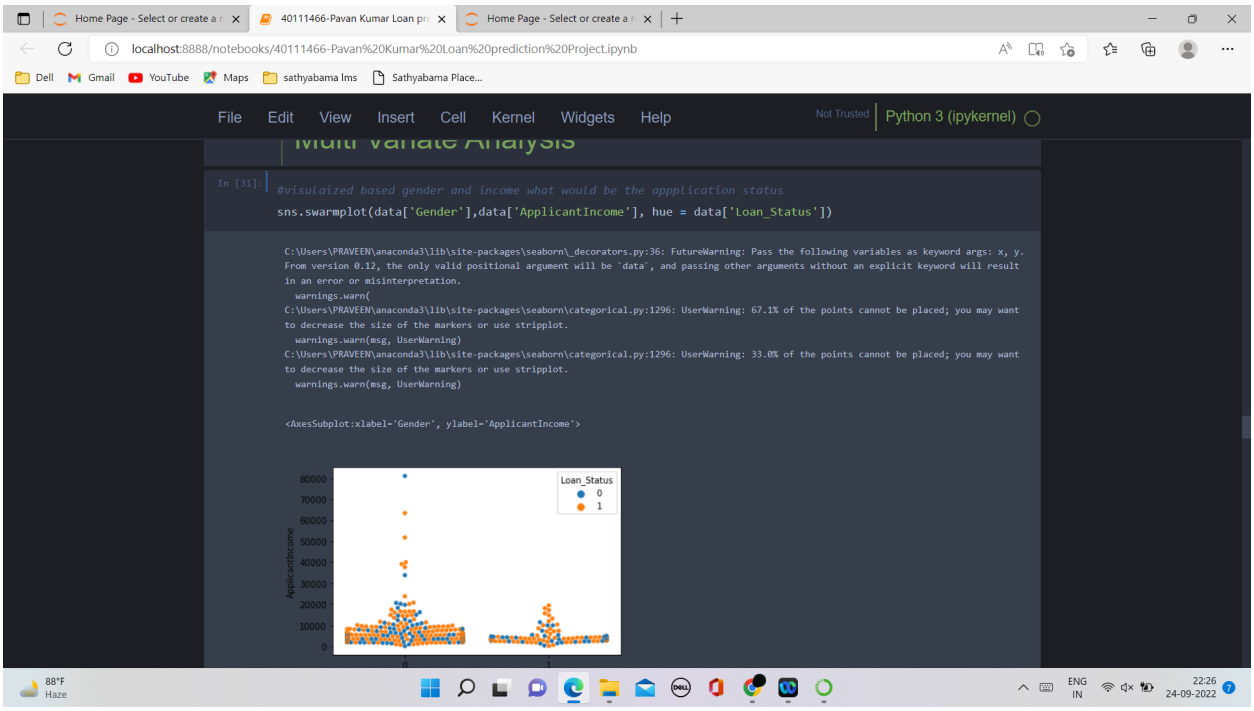
A count plot can be thought of as a histogram across a categorical, instead of a quantitative, variable. The basic API and options are identical to those for barplot() , so you can compare counts across nested variables.



rom the above graph, we can infer the analysis such as

- Segmenting the gender column and married column based on bar graphs
- Segmenting the Education and Self-employed based on bar graphs,for drawing insights such as educated people are employed.
- The loan amount term is based on the property area of a person holding

Multivariate Analysis



From the above graph we are plotting the relationship between the Gender, applicants income and loan status of the person

Descriptive Analysis

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas have a worthy function called describe. With this describe function we can understand the unique, top, and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features.

```
data.describe()
```

	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

Data Pre-Processing

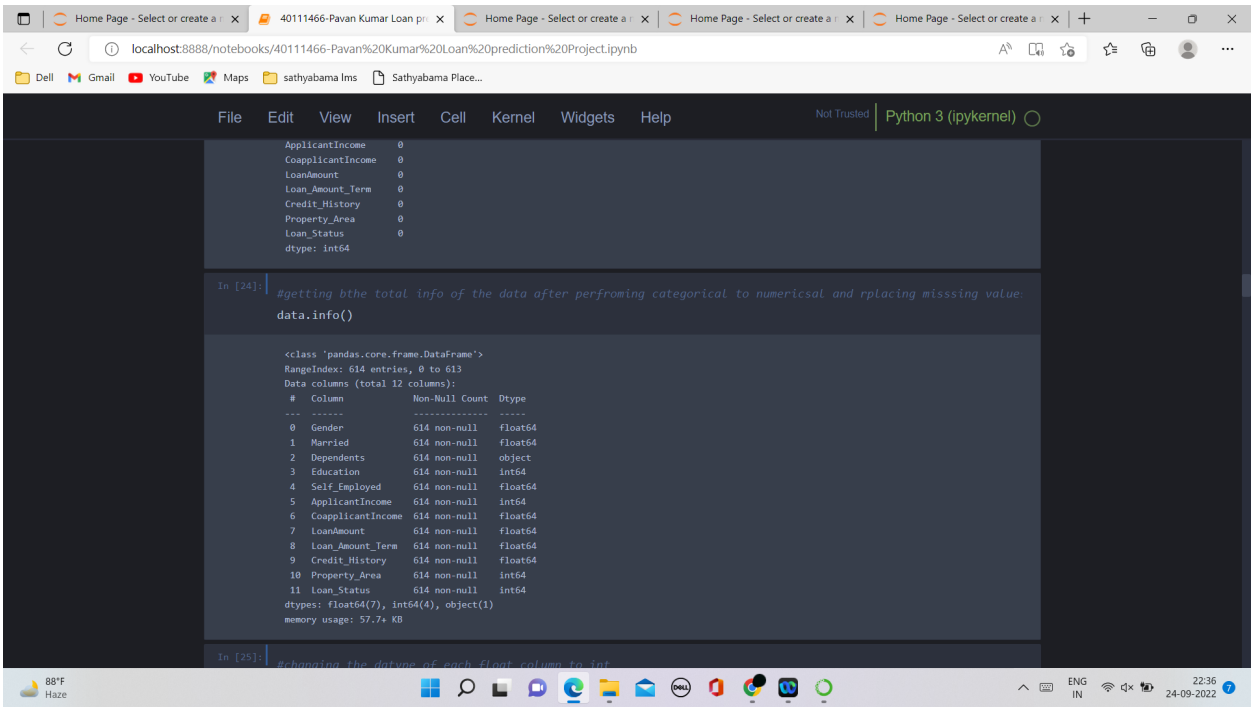
As we have understood how the data is let's pre-process the collected data.

The download data set is not suitable for training the machine learning model as it might have so much of randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

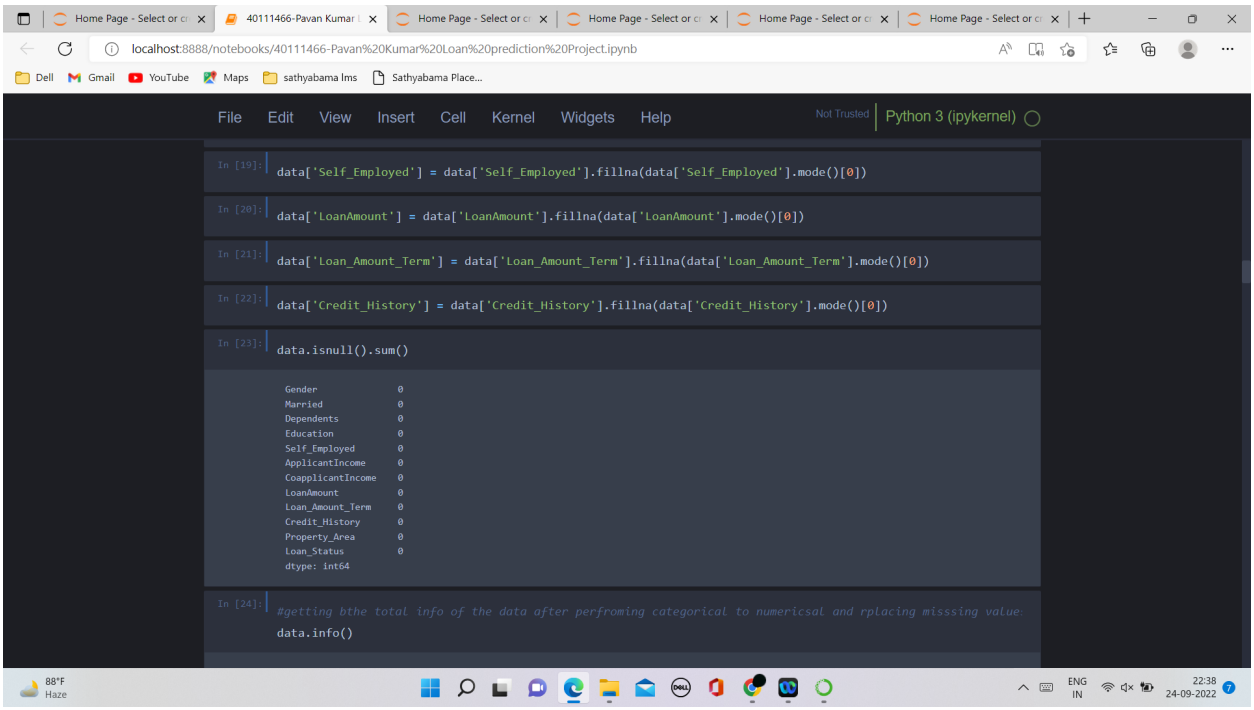
- Handling missing values
- Handling categorical data
- Handling outliers
- Scaling Techniques
- Splitting dataset into training and test set

Checking For Null Values

- Let’s find the shape of our dataset first, To find the shape of our data, df.shape method is used. To find the data type, df.info() function is used.

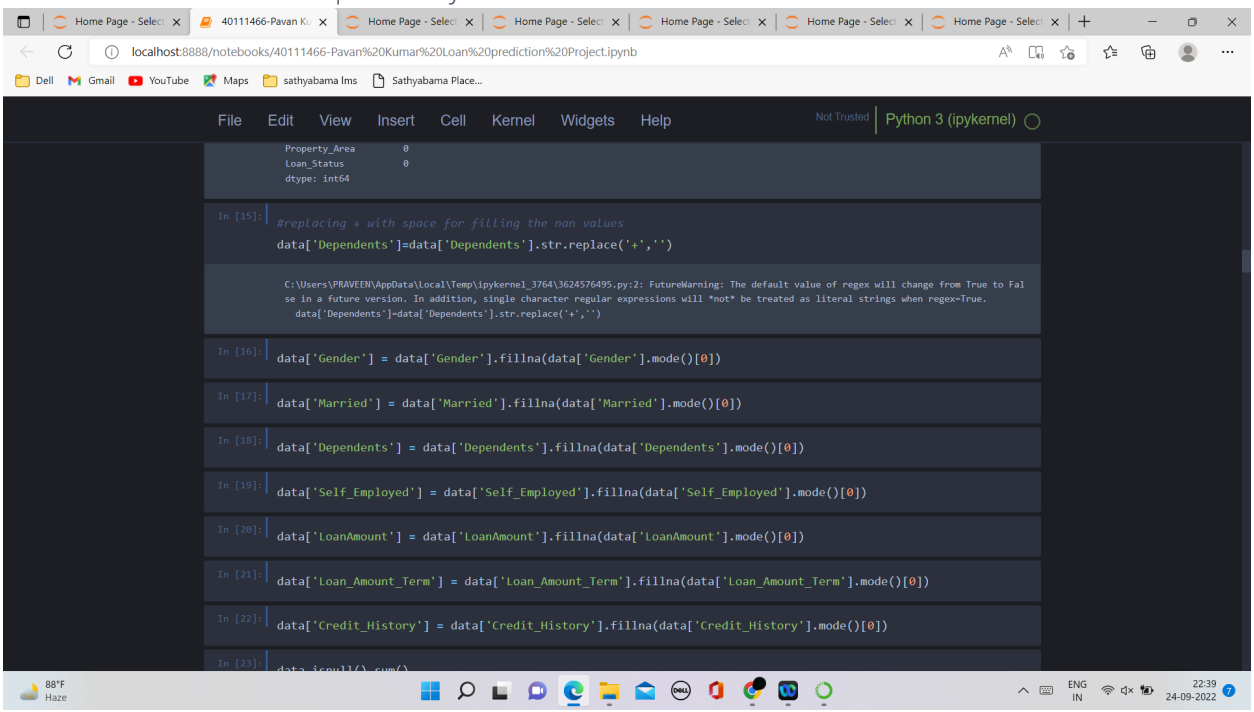


- For checking the null values, df.isnull() function is used. To sum those null values we use .sum() function to it. From the below image we found that there are no null values present in our dataset. So we can skip the handling of the missing values step.



From the above code of analysis, we can infer that columns such as gender, married, dependents, self-employed, loan amount, loan amount tern, and credit history are having the missing values, we

need to treat them in a required way.



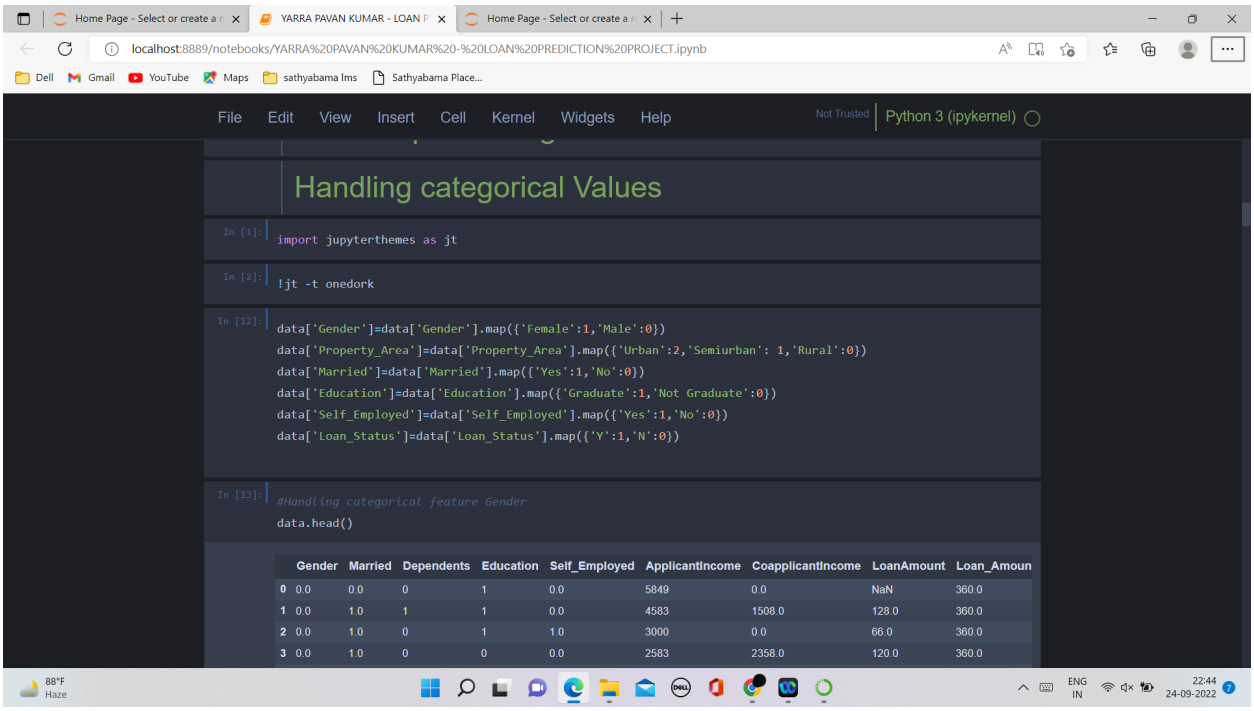
We will fill the missing values in numeric data type using the mean value of that particular column and categorical data type using the most repeated value.

Handling Categorical Values

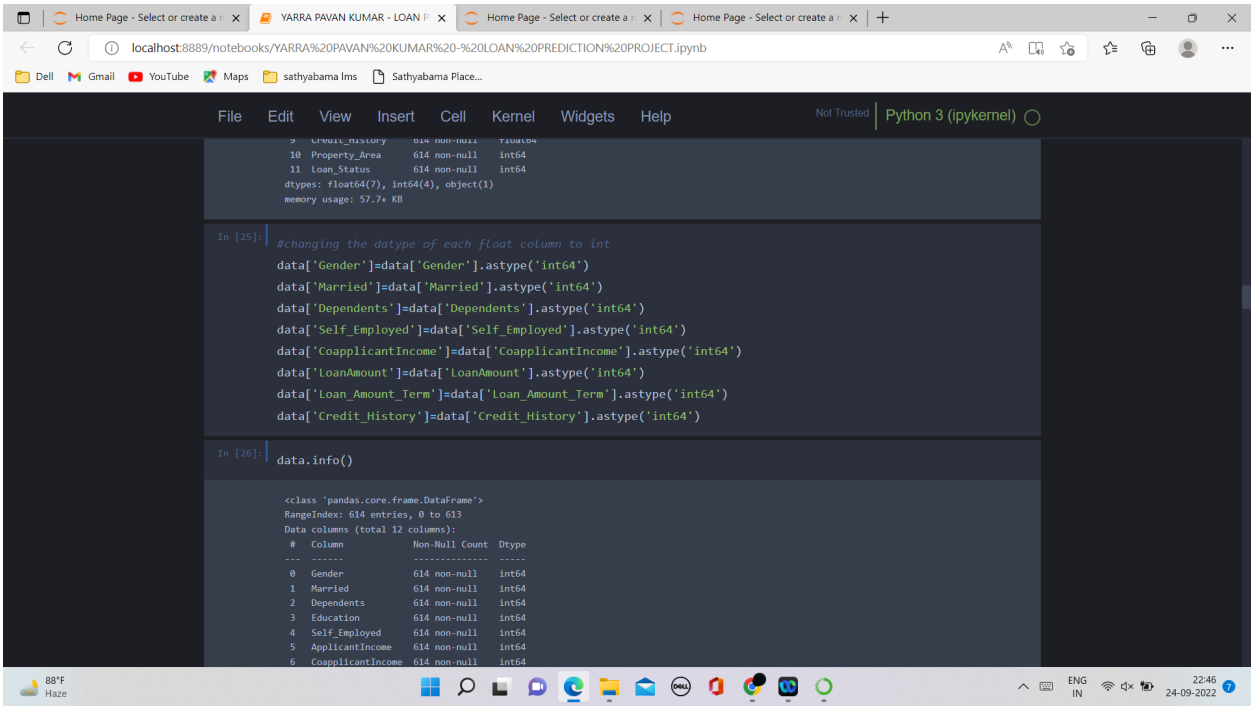
As we can see our dataset has categorical data we must convert the categorical data to integer encoding or binary encoding.

To convert the categorical features into numerical features we use encoding techniques. There are several techniques but in our project, we are using manual encoding with the help of list comprehension.

- In our project, Gender , married,dependents,self-employed,co-applicants income, loan amount , loan amount term, credit history With list comprehension encoding is done.



converting string datatype variables into integer data type

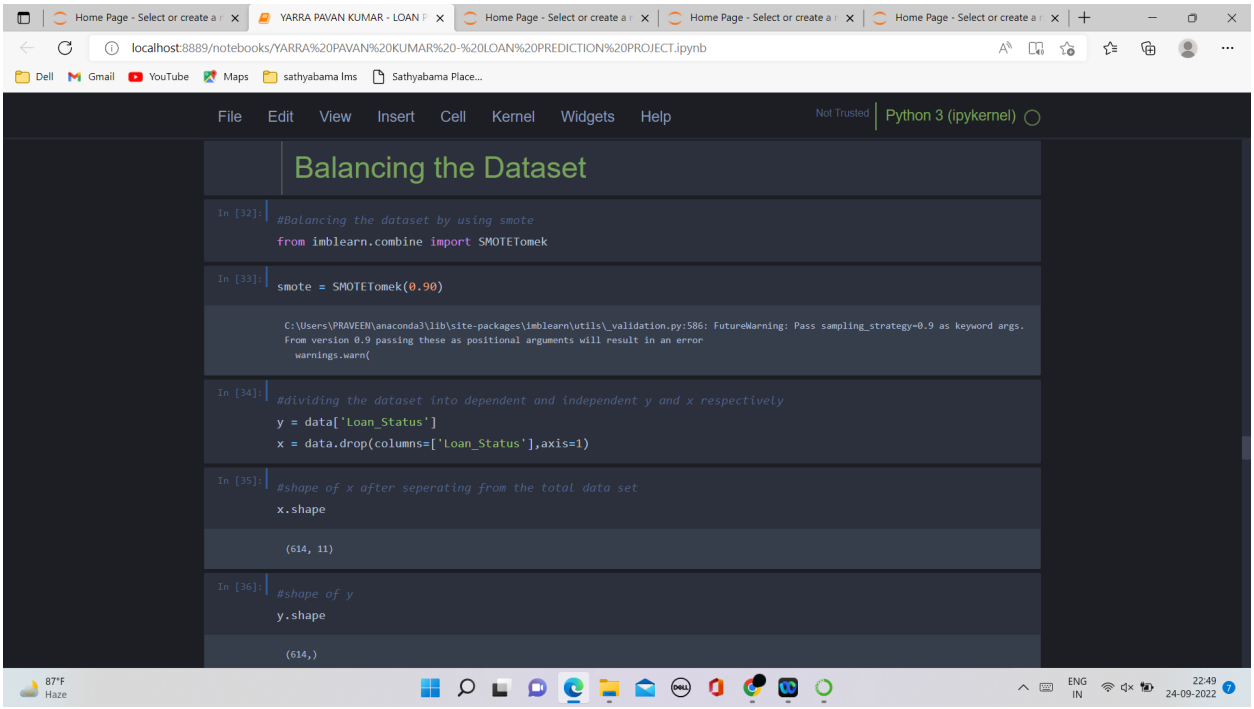


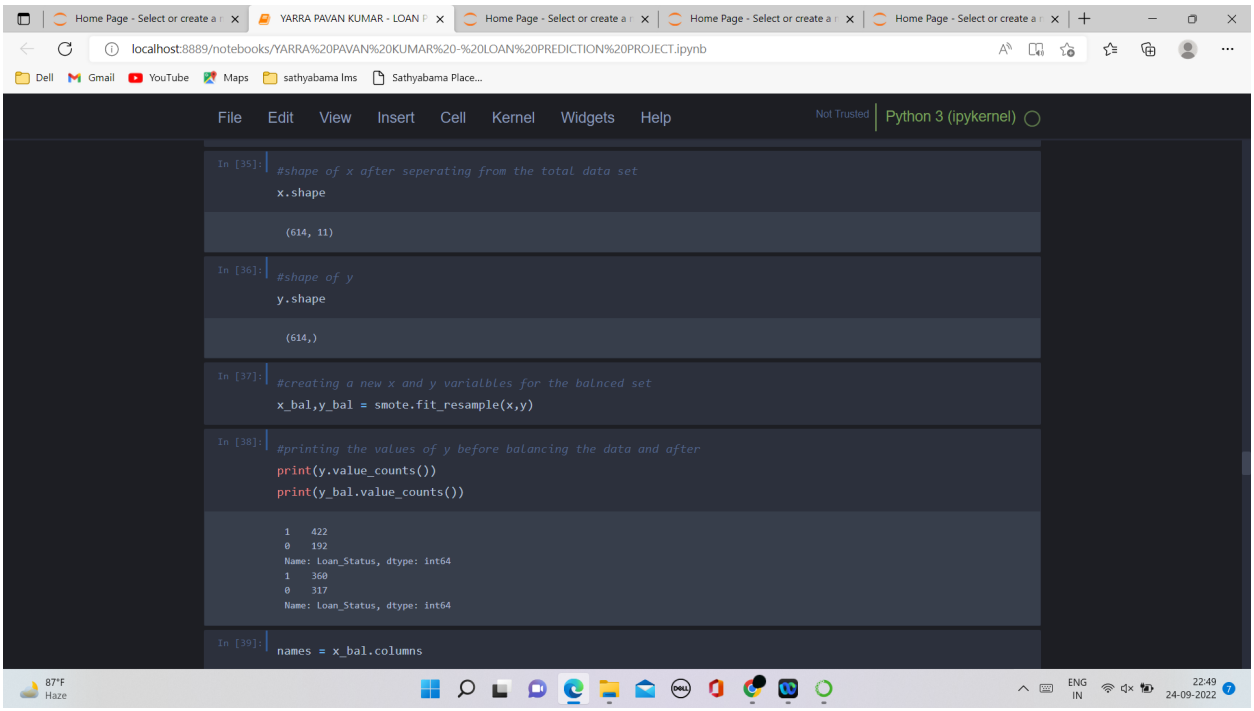
Balancing The Dataset

Data Balancing is one of the most important step, which need to be performed for classification models, because when we train our model on imbalanced dataset ,we will get biased results, which means our model is able to predict only one class element

For Balancing the data we are using SMOTE Method.

SMOTE: Synthetic minority over sampling technique, which will create new synthetic data points for under class as per the requirements given by us using KNN method.



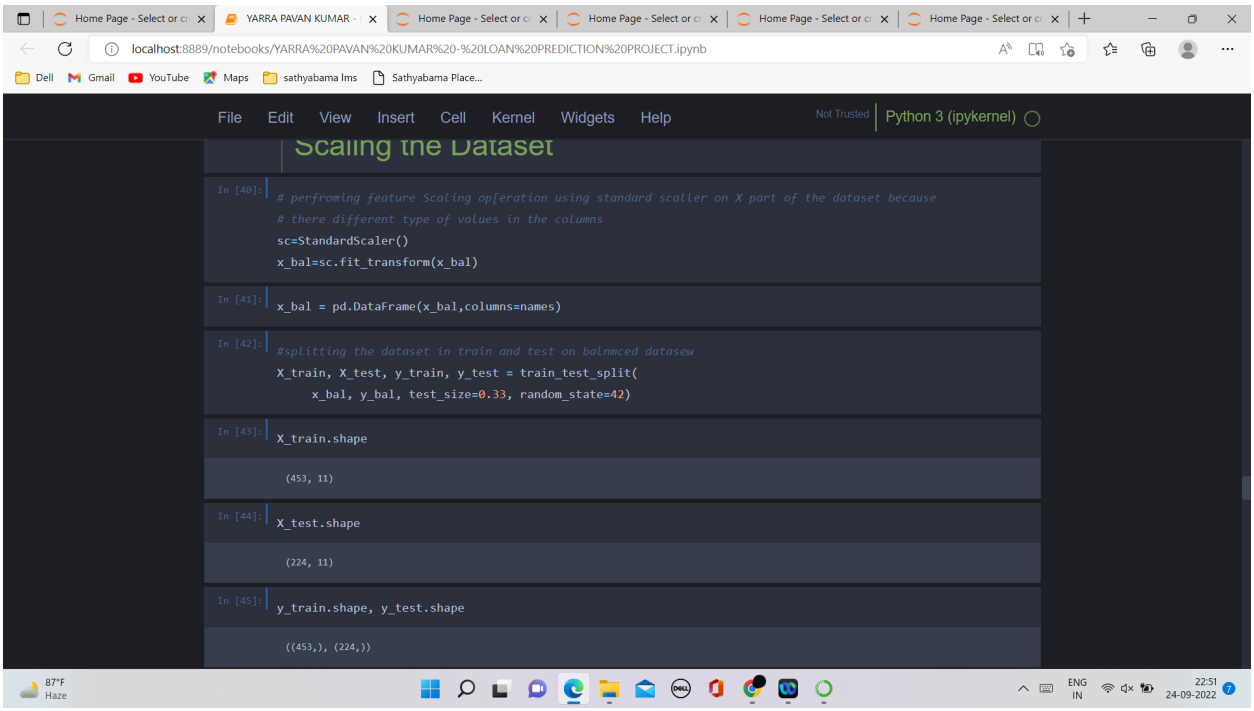


From the above picture, we can infer that previously our dataset is having 492 class 1, and 192 class items, after applying smote technique on the dataset the size has been changed for minority class.

Scaling The Data

Scaling is one the important process, we have to perform on the dataset, because of data measures in different ranges can leads to mislead in prediction

Models such as KNN, Logistic regression need scaled data, as they follow distance based method and Gradient Descent concept.



We will perform scaling only on the input values

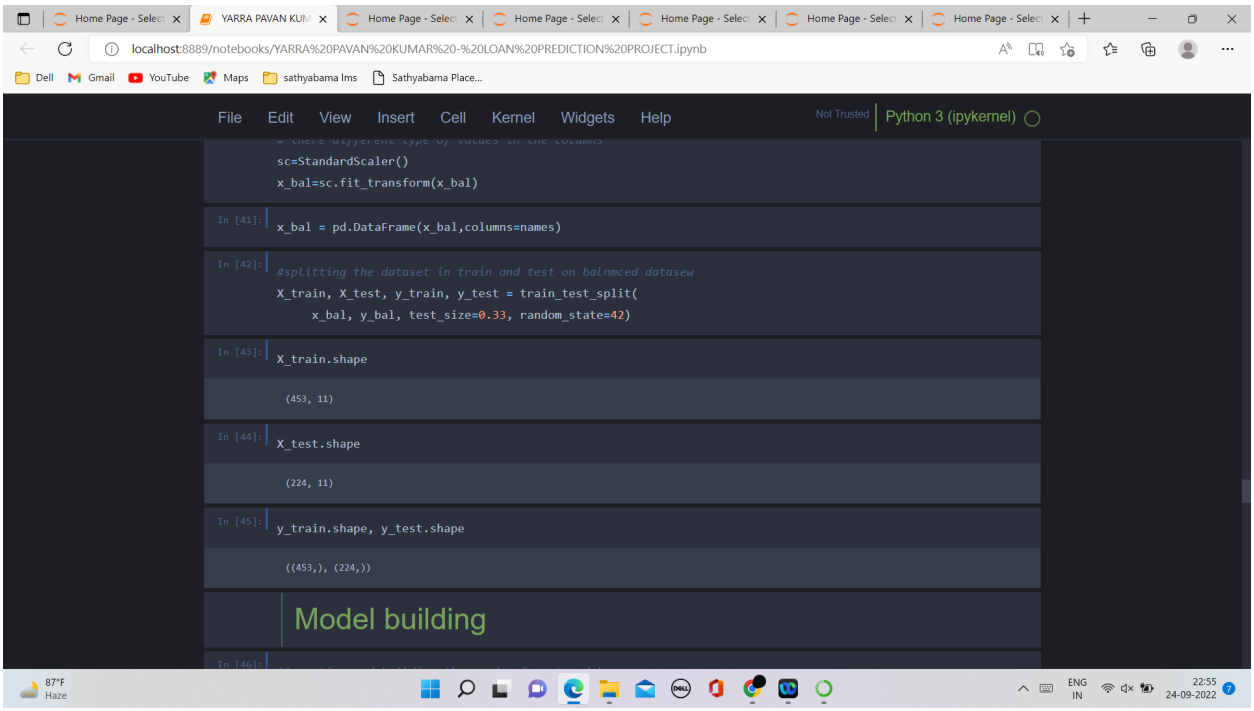
Once the dataset is scaled, it will be converted into array and we need to convert it back to dataframe.

Splitting Data Into Train And Test

Now let's split the Dataset into train and test sets

Changes: first split the dataset into x and y and then split the data set

Here x and y variables are created. On the x variable, df is passed by dropping the target variable. And on y target variable is passed. For splitting training and testing data, we are using the train_test_split() function from sklearn. As parameters, we are passing x, y, test_size, and random_state.

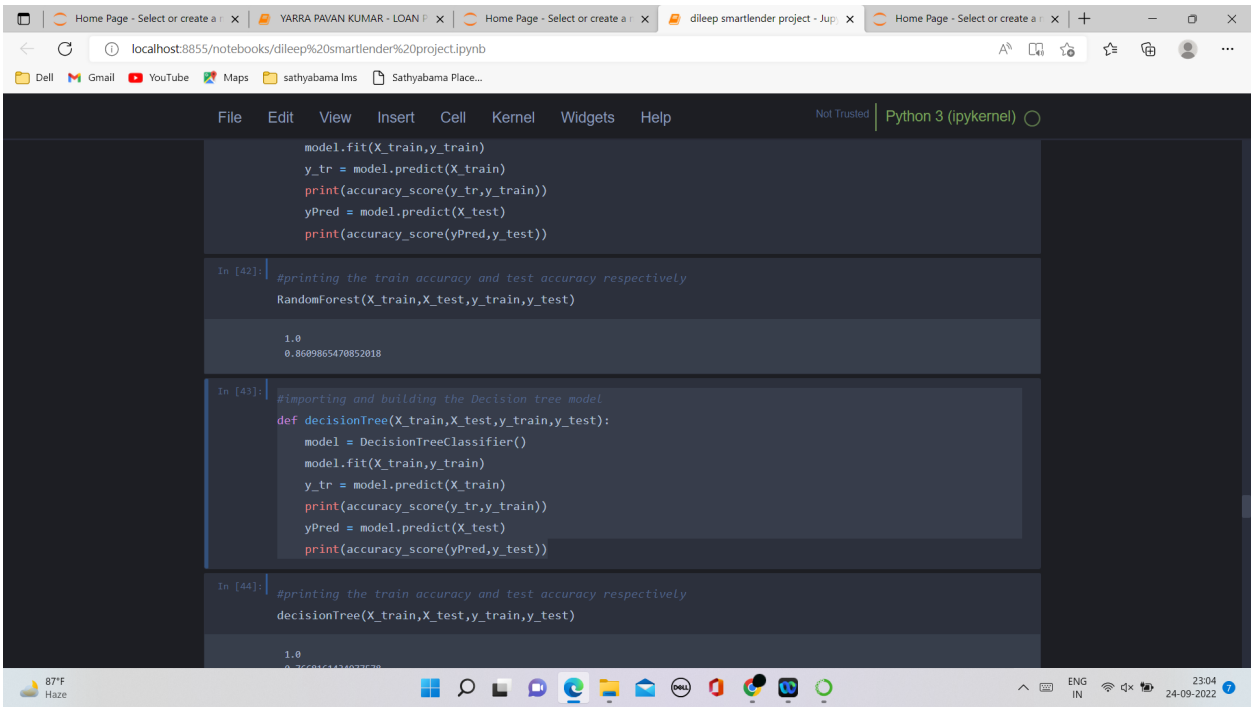


Model Building

Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. for this project, we are applying four classification algorithms. The best model is saved based on its performance.

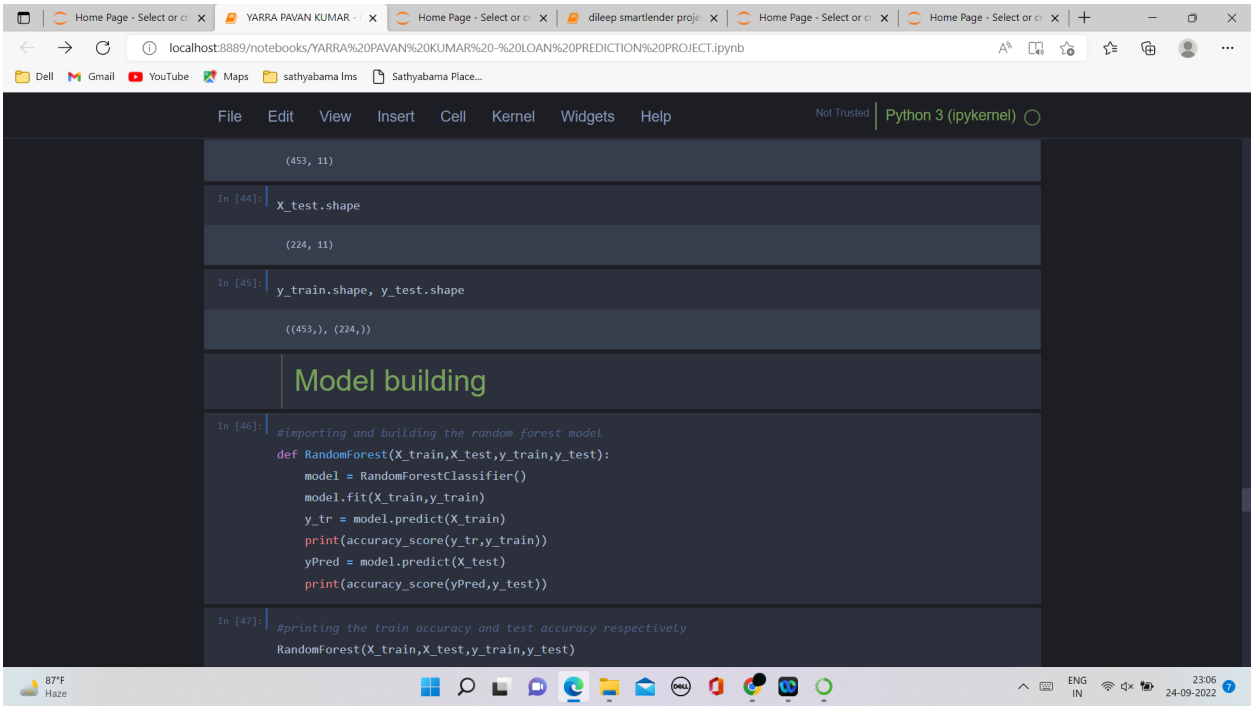
Decision Tree Model

A function named decision tree is created and train and test data are passed as the parameters. Inside the function, the DecisionTreeClassifier algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with the .predict() function and saved in the new variable. For evaluating the model, a confusion



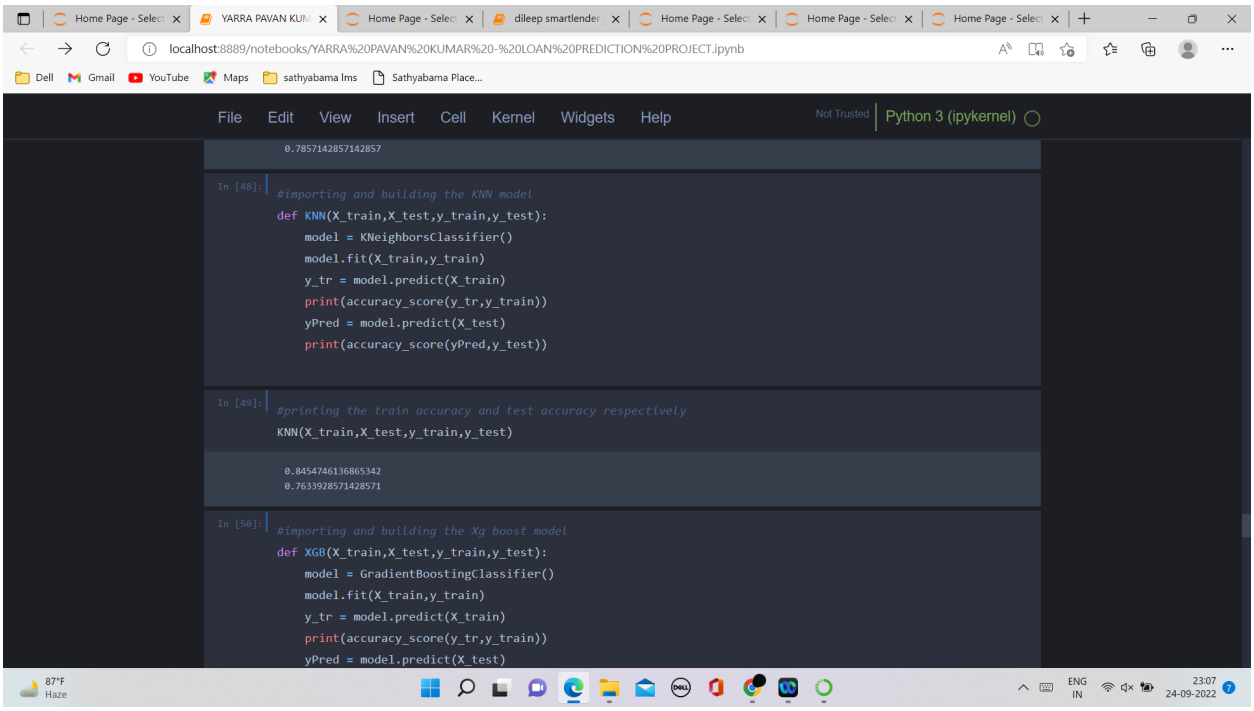
Random Forest Model

A function named randomForest is created and train and test data are passed as the parameters. Inside the function, the RandomForestClassifier algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report are done.



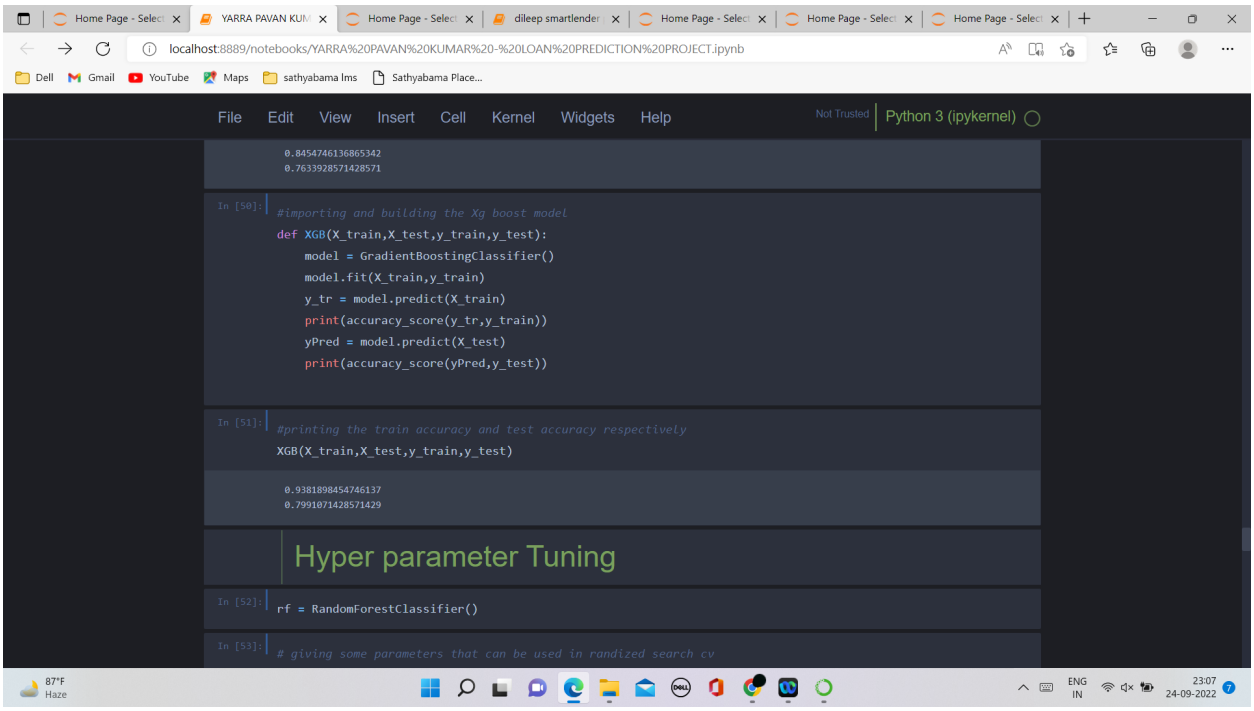
KNN Model

A function named KNN is created and train and test data are passed as the parameters. Inside the function, the KNeighborsClassifier algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.



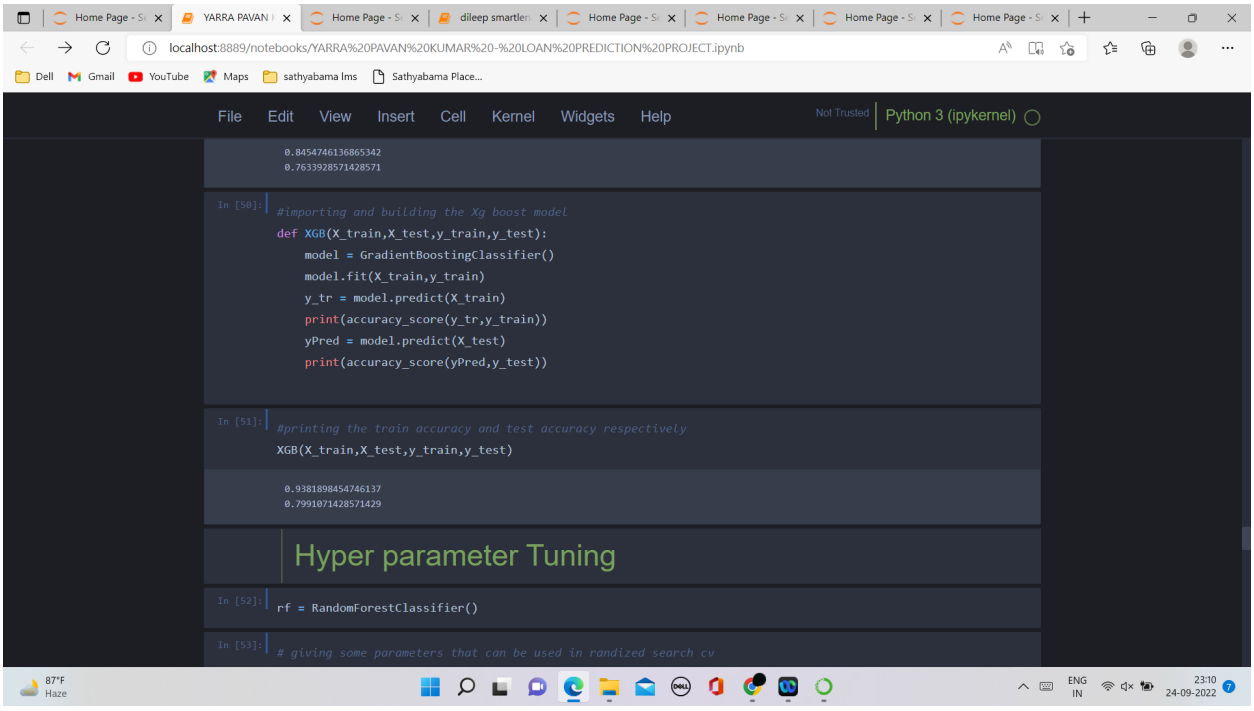
Xgboost Model

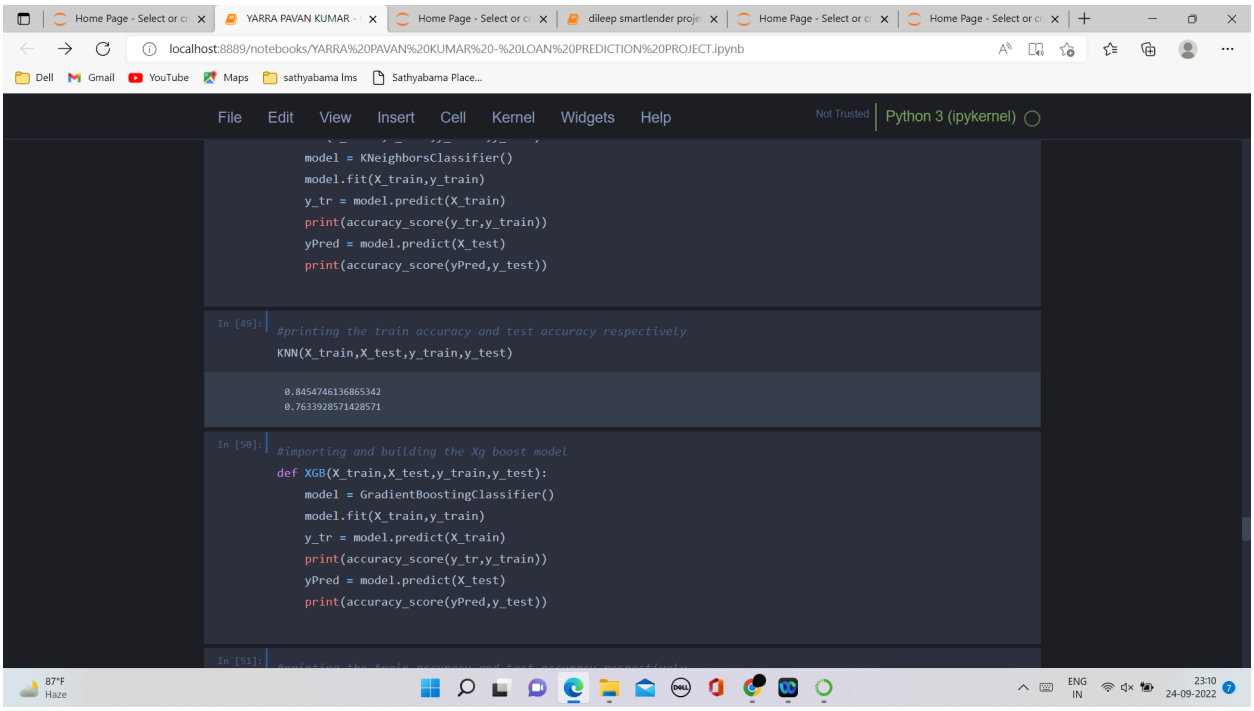
A function named xgboost is created and train and test data are passed as the parameters. Inside the function, the GradientBoostingClassifier algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report are done.



Compare The Model

For comparing the above four models compareModel function is defined.





Evaluating Performance Of The Model And Saving The Model

From sklearn, cross_val_score is used to evaluate the score of the model. On the parameters, we have given rf (model name), x, y, cv (as 5 folds). Our model is performing well. So, we are saving the model by pickle.dump().

Note: To understand cross-validation, refer this link

```
from sklearn.model_selection import cross_val_score

# Random forest model is selected

rf = RandomForestClassifier()
rf.fit(x_train,y_train)
yPred = rf.predict(x_test)

f1_score(yPred,y_test,average='weighted')

0.9679166666666668

cv = cross_val_score(rf,x,y,cv=5)

np.mean(cv)

0.985
```

```
#saviung the model by using pickle function
pickle.dump(model,open('rdf.pkl','wb'))
```

Application Building

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

- Building HTML Pages
- Building serverside script

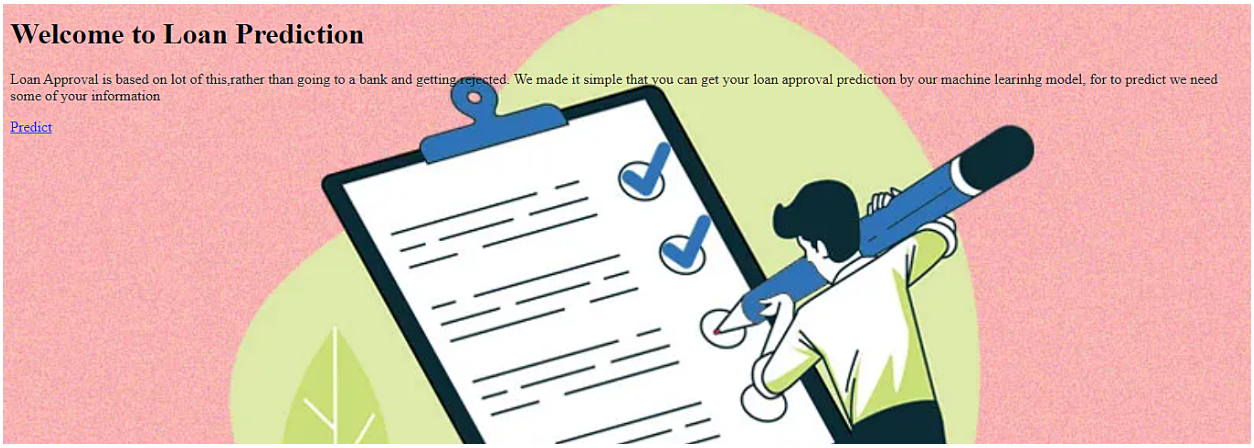
Building Html Pages

For this project create three HTML files namely

- home.html
- predict.html
- submit.html

and save them in the templates folder.

Let’s see how our home.html page looks like:



Now when you click on predict button from top right corner you will get redirected to predict.html

Lets look how our predict.html file looks like:

Now when you click on submit button from left bottom corner you will get redirected to submit.html

Lets look how our submit.html file looks like:

Build Python Code

```
from flask import Flask, render_template, request
import numpy as np
import pickle
```

Load the saved model. Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (__name__) as an argument.

```
app = Flask(__name__)
model = pickle.load(open(r'rdf.pkl', 'rb'))
scale = pickle.load(open(r'scale1.pkl','rb'))
```

Render HTML page:

```
@app.route('/') # rendering the html template
def home():
    return render_template('home.html')
```

Here we will be using declared constructor to route to the HTML page which we have created earlier.

In the above example, '/' URL is bound with home.html function. Hence, when the home page of the web server is opened in browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

Retrieves the value from UI:

```
@app.route('/submit',methods=["POST","GET"])# route to show the predictions in a web UI
def submit():
    #  reading the inputs given by the user
    input_feature=[int(x) for x in request.form.values() ]
    #input_feature = np.transpose(input_feature)
    input_feature=[np.array(input_feature)]
    print(input_feature)
    names = ['Gender', 'Married', 'Dependents', 'Education', 'Self_Employed', 'ApplicantIncome',
       'CoapplicantIncome','LoanAmount','Loan_Amount_Term','Credit_History','Property_Area']
    data = pandas.DataFrame(input_feature,columns=names)
    print(data)

    #data_scaled = scale.fit_transform(data)
    #data = pandas.DataFrame(,columns=names)



     # predictions using the loaded model file
    prediction=model.predict(data)
    print(prediction)
    prediction = int(prediction)
    print(type(prediction))

    if (prediction == 0):
       return render_template("output.html",result ="Loan wiil Not be Approved")
    else:
       return render_template("output.html",result = "Loan will be Approved")
     # showing the prediction results in a UI
```

Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using a Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will be rendered to the text that we have mentioned in the submit.html page earlier.

Main Function:

```
if __name__=="__main__":

    # app.run(host='0.0.0.0', port=8000,debug=True)    # running the app
    port=int(os.environ.get('PORT',5000))
    app.run(debug=False)
```

Run The Application

Run the application

- Open the anaconda prompt from the start menu

- Navigate to the folder where your python script is.
- Now type “python app.py” command
- Navigate to the localhost where you can view your web page.

Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web

```
base) D:\TheSmartBridge\Projects\2. DrugClassification\Drug c
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a p
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```