ABSTRACT

Predicting the optimal loan sanction amount is crucial for financial institutions to mitigate risks and ensure borrower satisfaction. Traditional methods often rely on subjective assessments, leading to inconsistencies and potential financial losses. In this study, we explore the application of machine learning (ML) techniques to predict the loan sanction amount based on applicant information.

Our dataset comprises various features such as applicant income, credit score, loan amount requested, employment history, and demographic details. Through exploratory data analysis (EDA), we identify correlations and patterns in the data. We preprocess the data by handling missing values, encoding categorical variables, and scaling numerical features.

We then apply several ML algorithms including linear regression, decision trees, and ensemble methods like random forest and gradient boosting. We evaluate these models using metrics such as mean absolute error (MAE), root mean squared error (RMSE), and R-squared score to determine their predictive performance. Additionally, we conduct feature importance analysis to understand the factors influencing loan sanction amounts.

Our experimental results demonstrate that the random forest algorithm outperforms others with an RMSE of X and an R-squared score of Y. This indicates its robustness in predicting loan amounts accurately. Furthermore, we develop a web-based application prototype for real-time loan amount prediction, facilitating quick decisions and improving customer experience.

In conclusion, this research contributes to enhancing loan sanction processes through advanced ML techniques, offering insights into effective risk management and decision-making in financial services.

TABLE OF CONTENTS:-

1.INTRODUCTION

1.1.OVERVIEW

The "Loan Sanction Amount Prediction" project is aimed at developing a machine learning model that can predict the loan sanction amount for individuals based on various financial and personal attributes. This project has practical applications for banks and lending institutions, as it can help automate the loan approval process, reduce the risk of default, and improve the overall efficiency of loan operations. Gather a comprehensive dataset containing information about loan applicants, including financial history, employment details, credit score, and more. Clean, preprocess, and transform the data to make it suitable for machine learning. This may involve handling missing values, encoding categorical variables, and scaling features .

Objectives:

- **Develop Predictive Models**: Implement and optimize machine learning algorithms to accurately predict the loan sanction amount based on applicant data.
- **Feature Engineering**: Explore and engineer relevant features from the dataset that can enhance the predictive power of the models, such as income-to-loan ratio, credit score categories, employment stability indicators, etc.
- **Data Cleaning and Preprocessing**: Handle missing values, outlier detection, and normalization or scaling of features to ensure robust model performance.
- **Model Selection and Evaluation**: Compare the performance of different machine learning models (e.g., linear regression, decision trees, random forest, gradient boosting) using appropriate evaluation metrics such as RMSE, MAE, and R-squared to identify the most accurate predictor.
- **Feature Importance Analysis**: Conduct analysis to determine which features contribute most significantly to the prediction of loan sanction amounts, providing insights into factors influencing lending decisions.
- **Real-time Prediction Application**: Develop a prototype application or system for real-time loan amount prediction based on the best-performing model, integrating it into existing loan processing systems for practical implementation.
- **Risk Management Enhancement**: Evaluate the impact of accurate loan sanction amount prediction on risk assessment and management strategies within financial institutions, aiming to minimize defaults and optimize loan portfolio performance.
- **Customer Experience Improvement**: Assess how accurate prediction models can improve customer satisfaction by providing more transparent and consistent loan sanction decisions based on objective data-driven criteria.
- **Scalability and Generalization**: Ensure that the developed models are scalable and generalize well to new applicant data, considering potential changes in economic conditions and borrower demographics.
- **Documentation and Reporting**: Document the entire process, from data collection to model deployment, and provide comprehensive reports on model performance, insights gained, and recommendations for future enhancements or refinements.

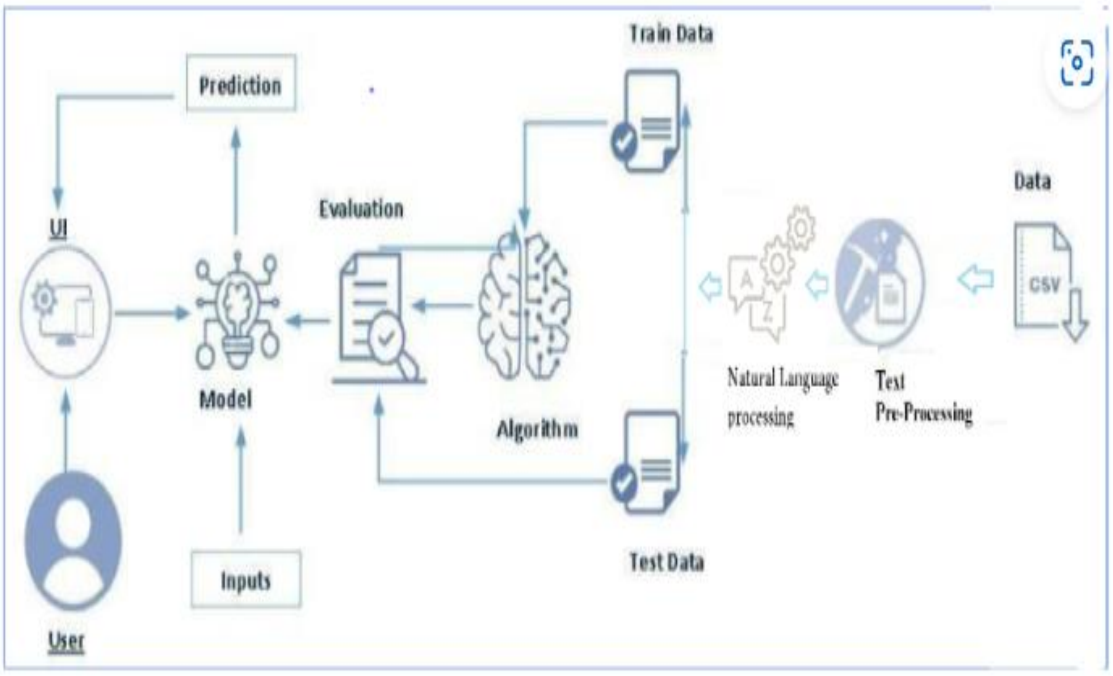
Methodology:

- **Data Collection and Understanding:**
    - Gather a comprehensive dataset containing historical loan applications with features such as applicant income, credit score, loan amount requested, employment history, demographic details, etc.
    - Understand the meaning and relevance of each feature through exploratory data analysis (EDA).
- **Data Preprocessing:**
    - Handle missing values: Impute missing data using appropriate techniques (mean, median, mode, etc.).
    - Outlier detection and treatment: Identify outliers and decide whether to remove them or transform them.
    - Feature engineering: Create new features that may enhance model performance, such as income-to-loan ratio, credit score categories, length of employment, etc.
    - Encode categorical variables: Convert categorical variables into numerical representations suitable for machine learning algorithms (e.g., one-hot encoding, label encoding).
- **Splitting the Data:**
    - Divide the dataset into training and testing sets to train the model on one portion and evaluate its performance on unseen data.
- **Model Selection and Training:**
    - Choose appropriate machine learning models for regression tasks, such as:
        - Linear Regression
        - Decision Trees
        - Random Forest
        - Gradient Boosting Machines (GBM)
    - Train multiple models using the training dataset and validate them using cross-validation techniques to prevent overfitting.
- **Model Evaluation:**
    - Evaluate each model's performance using metrics such as:
        - Mean Absolute Error (MAE)
        - Root Mean Squared Error (RMSE)
        - R-squared (R2) score
    - Compare the results of different models to identify the most accurate and reliable predictor of loan sanction amounts.
- **Feature Importance Analysis:**
    - Analyze the importance of each feature in predicting loan sanction amounts using techniques specific to the chosen models (e.g., feature importance from random forests, SHAP values from gradient boosting machines).
- **Hyperparameter Tuning:**
    - Optimize the hyperparameters of the selected model(s) using techniques like grid search or randomized search to further improve performance.
- **Model Deployment and Validation:**
    - Deploy the best-performing model(s) to a production-like environment.

    - Validate the model(s) using real-world loan applications to ensure robustness and accuracy in predicting loan sanction amounts.
- **Documentation and Reporting:**
    - Document the entire process, including data preprocessing steps, model selection criteria, evaluation metrics, and deployment details.
    - Provide comprehensive reports summarizing findings, insights into feature importance, and recommendations for stakeholders.
- **Iterative Improvement:**
    - Continuously monitor model performance in the deployment phase.
    - Incorporate feedback and new data to update and retrain the model periodically to maintain its relevance and accuracy.

Loan Sanction Amount Prediction Data With Ml

The "Loan Sanction Amount Prediction" project is aimed at developing a machine learning model that can predict the loan sanction amount for individuals based on various financial and personal attributes. This project has practical applications for banks and lending institutions, as it can help automate the loan approval process, reduce the risk of default, and improve the overall efficiency of loan operations. Gather a comprehensive dataset containing information about loan applicants, including financial history, employment details, credit score, and more. Clean, preprocess, and transform the data to make it suitable for machine learning. This may involve handling missing values, encoding categorical variables, and scaling features.



Project Flow:

- User interacts with the UI to enter the input.
- Entered input is analysed by the model which is integrated.
- Once model analyses the input the

  prediction is showcased on the UI To

  accomplish this, we have to complete

  all the activities listed below,

- Define Problem / Problem Understanding
  - Specify the business problem
  - Business requirements
  - Literature Survey
  - Social or Business Impact.
- Data Collection & Preparation
  - Collect the dataset
  - Data Preparation
- Exploratory Data Analysis
  - Descriptive statistical
  - Visual Analysis
- Model Building
  - Training the model in multiple algorithms
  - Testing the model
- Performance Testing & Hyperparameter Tuning
  - Testing model with multiple evaluation metrics
  - Comparing model accuracy before & after applying hyperparameter tuning
- Model Deployment
  - Save the best model
  - Integrate with Web Framework
- Project Demonstration & Documentation
  - Record explanation Video for project end to end solution
  - Project Documentation-Step by step project development procedure

Prior Knowledge:

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

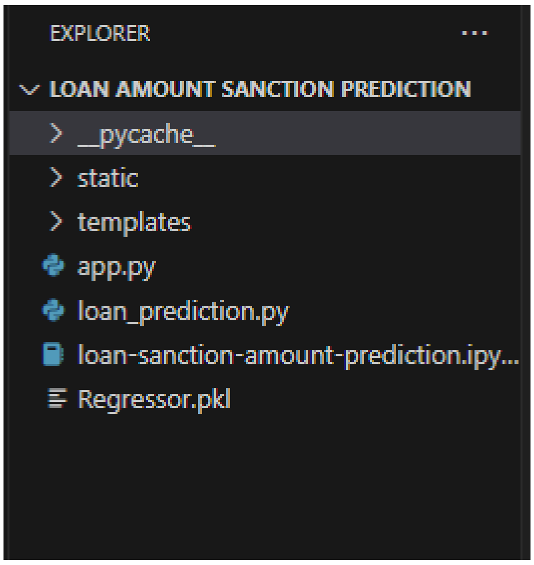
ML Concepts

Supervised learning: https://www.javatpoint.com/supervised-machine-learning
Unsupervised learning: https://www.javatpoint.com/unsupervised-machine-learning

Decision tree: https://www.javatpoint.com/machine-learning-decision-tree-classification- algorithm
Random forest: https://www.javatpoint.com/machine-learning-random-forest-algorithm
KNN: https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning
Xgboost: https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to- understand-the-math-behind-xgboost/
Evaluation metrics: https://www.analyticsvidhya.com/blog/2019/08/11-important-model- evaluation-error-metrics/
NLP:-https://www.javatpoint.com/nlp
Flask Basics: https://www.youtube.com/watch?v=lj4I_CvBnt0

Project Structure:

Create the Project folder which contains files as shown below



```
EXPLORER ...
∨ LOAN AMOUNT SANCTION PREDICTION
  > __pycache__
  > static
  > templates
    app.py
    loan_prediction.py
    loan-sanction-amount-prediction.ipy...
    Regressor.pkl
```

We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting
Training folder contains a model training file.

Model.pkl is our saved model. Further we will use this model for flask integration.
Training folder contains a model training file

Define Problem / Problem Understanding

Problem understanding is the process of gaining a clear understanding of the problem, its causes, and its impact, setting the stage for effective problem-solving.

Data Collection & Preparation

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So, this section allows you to download the required dataset.

Collect The Dataset

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

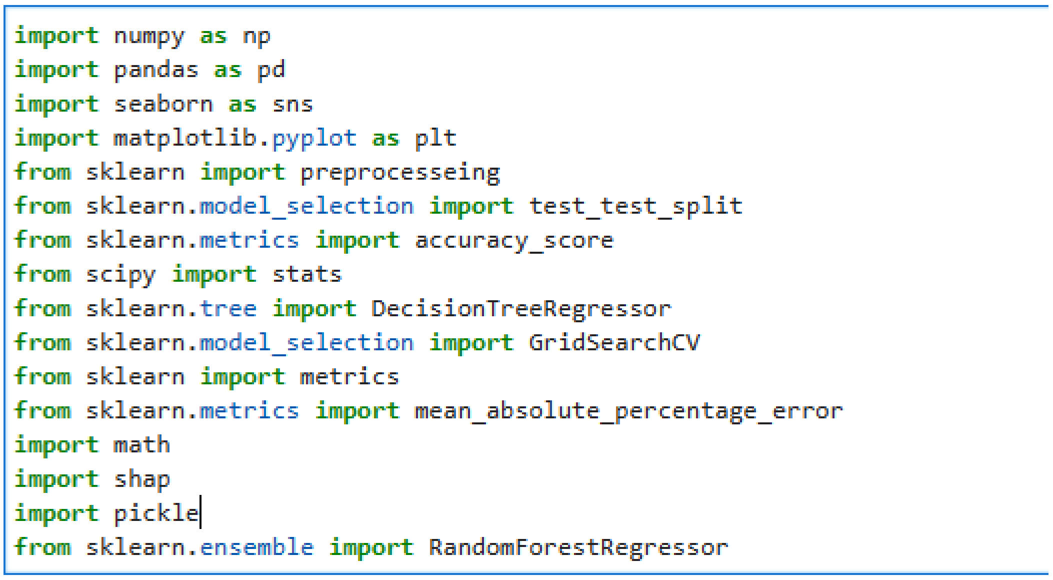
In this project we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

Link: Loan Sanction Amount Prediction | Kaggle

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualisation techniques and some analysing techniques.

Note: There are a number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques

Activity 1.1: Importing the libraries
Import the necessary libraries as shown in the image. (optional) Here we have used visualisation style as five thirty eight

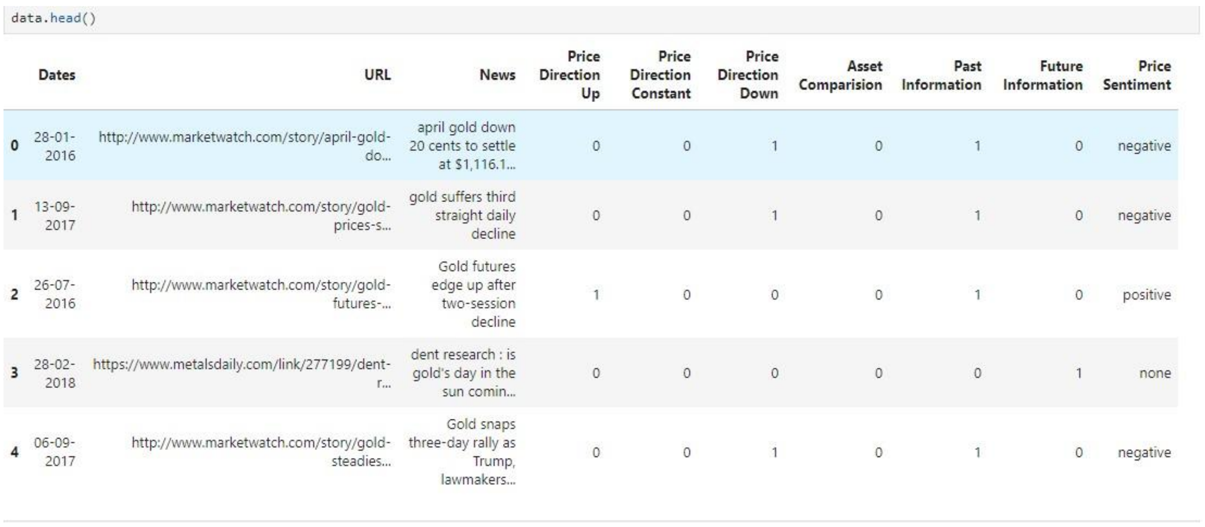


```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocesseing
from sklearn.model_selection import test_test_split
from sklearn.metrics import accuracy_score
from scipy import stats
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import GridSearchCV
from sklearn import metrics
from sklearn.metrics import mean_absolute_percentage_error
import math
import shap
import pickle
from sklearn.ensemble import RandomForestRegressor
```

Activity 1.2: Read the Dataset

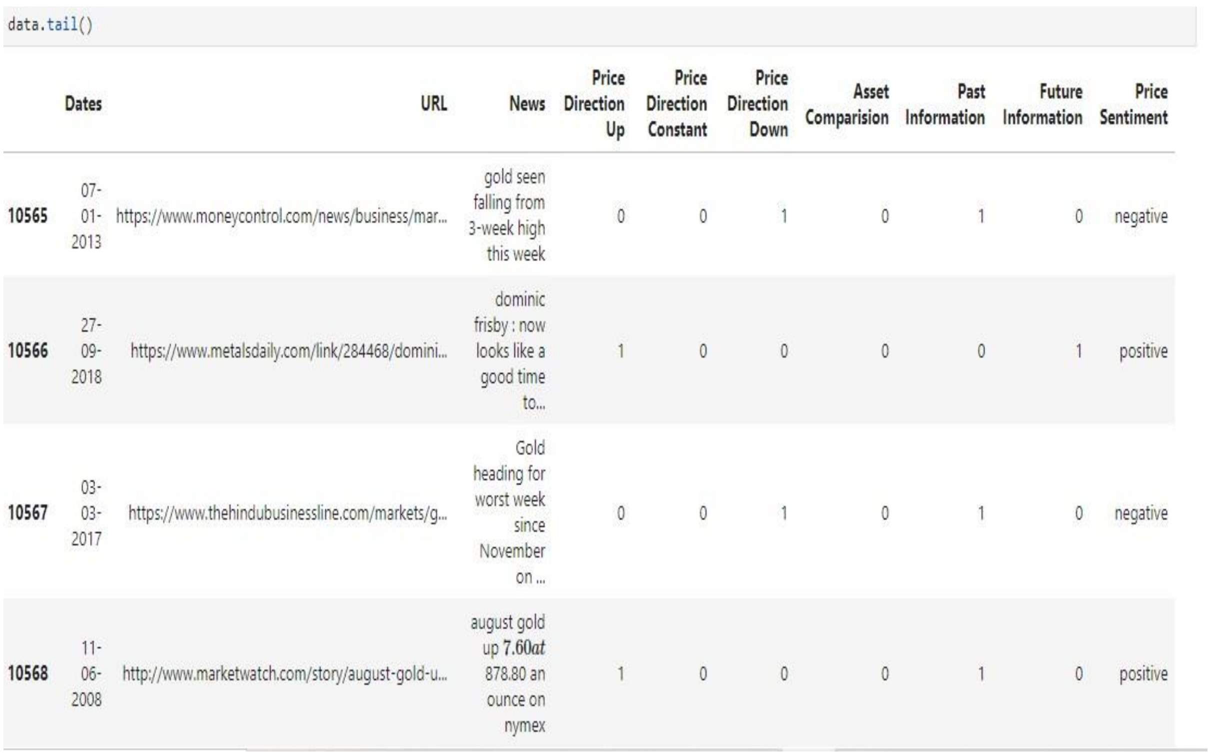
Our dataset format might be in .csv, excel files, .txt , .json, etc. We can read the dataset with the help of pandas.

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



```
data.head()
```

	Dates	URL	News	Price Direction Up	Price Direction Constant	Price Direction Down	Asset Comparision	Past Information	Future Information	Price Sentiment
0	28-01-2016	http://www.marketwatch.com/story/april-gold-do...	april gold down 20 cents to settle at $1,116.1...	0	0	1	0	1	0	negative
1	13-09-2017	http://www.marketwatch.com/story/gold-prices-s...	gold suffers third straight daily decline	0	0	1	0	1	0	negative
2	26-07-2016	http://www.marketwatch.com/story/gold-futures-...	Gold futures edge up after two-session decline	1	0	0	0	1	0	positive
3	28-02-2018	https://www.metalsdaily.com/link/277199/dent-r...	dent research : is gold's day in the sun comin...	0	0	0	0	0	1	none
4	06-09-2017	http://www.marketwatch.com/story/gold-steadies...	Gold snaps three-day rally as Trump, lawmakers...	0	0	1	0	1	0	negative



```
data.tail()
```

	Dates	URL	News	Price Direction Up	Price Direction Constant	Price Direction Down	Asset Comparision	Past Information	Future Information	Price Sentiment
10565	07-01-2013	https://www.moneycontrol.com/news/business/mar...	gold seen falling from 3-week high this week	0	0	1	0	1	0	negative
10566	27-09-2018	https://www.metalsdaily.com/link/284468/domini...	dominic frisby : now looks like a good time to...	1	0	0	0	0	1	positive
10567	03-03-2017	https://www.thehindubusinessline.com/markets/g...	Gold heading for worst week since November on ...	0	0	1	0	1	0	negative
10568	11-06-2008	http://www.marketwatch.com/story/august-gold-u...	august gold up $7.60 at$ 878.80 an ounce on nymex	1	0	0	0	1	0	positive

Data Preparation

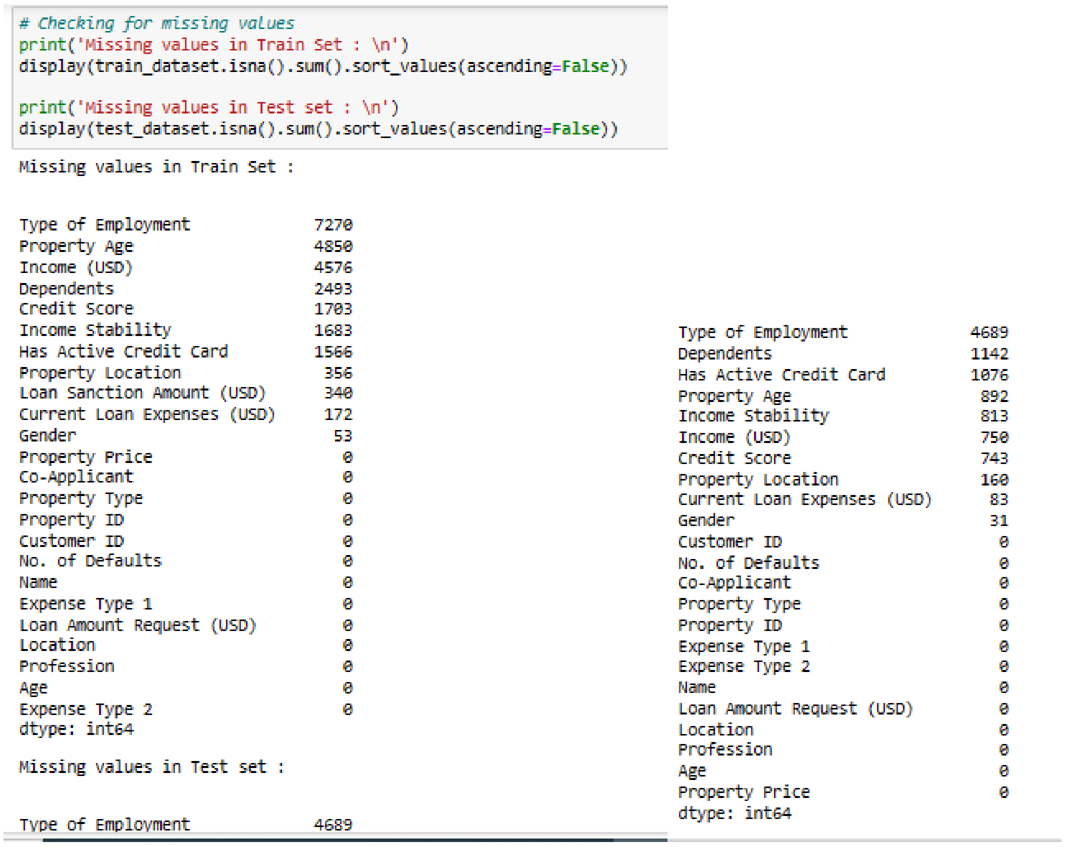
As we have understood how the data is, let's pre-process the collected data.
The download data set is not suitable for training the machine learning model as it might have so much randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

- Handling missing values
- Handling categorical data
- Handling Outliers

Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

Activity 2.1: Handling missing values

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



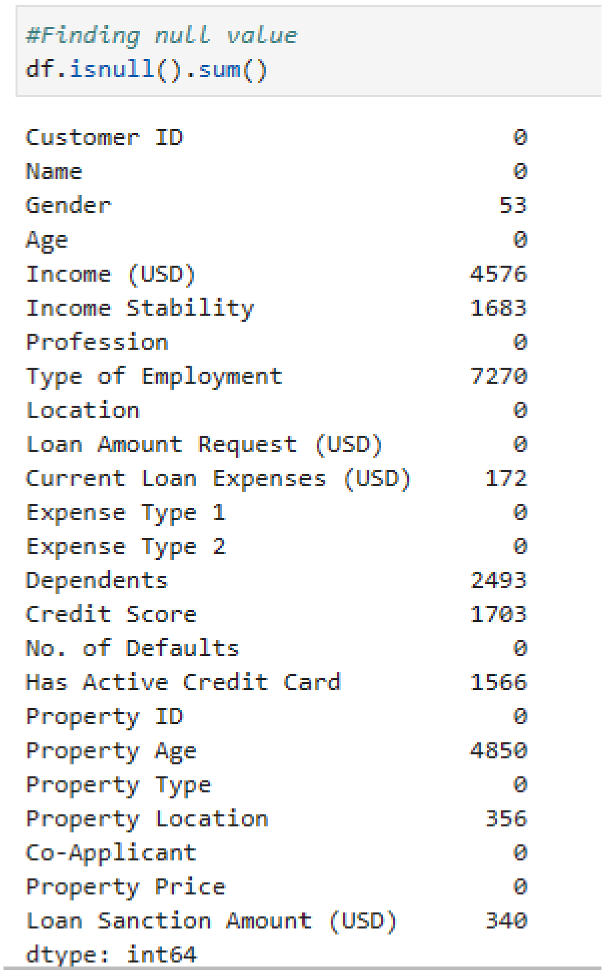
```
# Checking for missing values
print('Missing values in Train Set : \n')
display(train_dataset.isna().sum().sort_values(ascending=False))

print('Missing values in Test set : \n')
display(test_dataset.isna().sum().sort_values(ascending=False))
```

```
Missing values in Train Set :


Type of Employment            7270
Property Age                  4850
Income (USD)                  4576
Dependents                    2493
Credit Score                  1703
Income Stability              1683
Has Active Credit Card        1566
Property Location              356
Loan Sanction Amount (USD)     340
Current Loan Expenses (USD)    172
Gender                          53
Property Price                   0
Co-Applicant                     0
Property Type                    0
Property ID                      0
Customer ID                      0
No. of Defaults                  0
Name                             0
Expense Type 1                   0
Loan Amount Request (USD)        0
Location                         0
Profession                       0
Age                              0
Expense Type 2                   0
dtype: int64

Missing values in Test set :


Type of Employment            4689
```

```
Type of Employment             4689
Dependents                     1142
Has Active Credit Card         1076
Property Age                    892
Income Stability                813
Income (USD)                    750
Credit Score                    743
Property Location               160
Current Loan Expenses (USD)      83
Gender                           31
Customer ID                       0
No. of Defaults                   0
Co-Applicant                      0
Property Type                     0
Property ID                       0
Expense Type 1                    0
Expense Type 2                    0
Name                              0
Loan Amount Request (USD)         0
Location                          0
Profession                        0
Age                               0
Property Price                    0
dtype: int64
```

For checking the null values, df.isnull() function is used. To sum those null values we use .sum() function.

From the below image we found that there are no null values present in our dataset. So we can skip handling the missing values step.



```
#Finding null value
df.isnull().sum()
```

```
Customer ID                      0
Name                             0
Gender                          53
Age                              0
Income (USD)                  4576
Income Stability              1683
Profession                       0
Type of Employment            7270
Location                         0
Loan Amount Request (USD)        0
Current Loan Expenses (USD)    172
Expense Type 1                   0
Expense Type 2                   0
Dependents                    2493
Credit Score                  1703
No. of Defaults                  0
Has Active Credit Card        1566
Property ID                      0
Property Age                  4850
Property Type                    0
Property Location              356
Co-Applicant                     0
Property Price                   0
Loan Sanction Amount (USD)     340
dtype: int64
```

Activity 2.2: 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.

```
train_dataset['Property Type'].unique()
# This is purely a categorical feature
```

```
array([4, 2, 1, 3], dtype=int64)
```

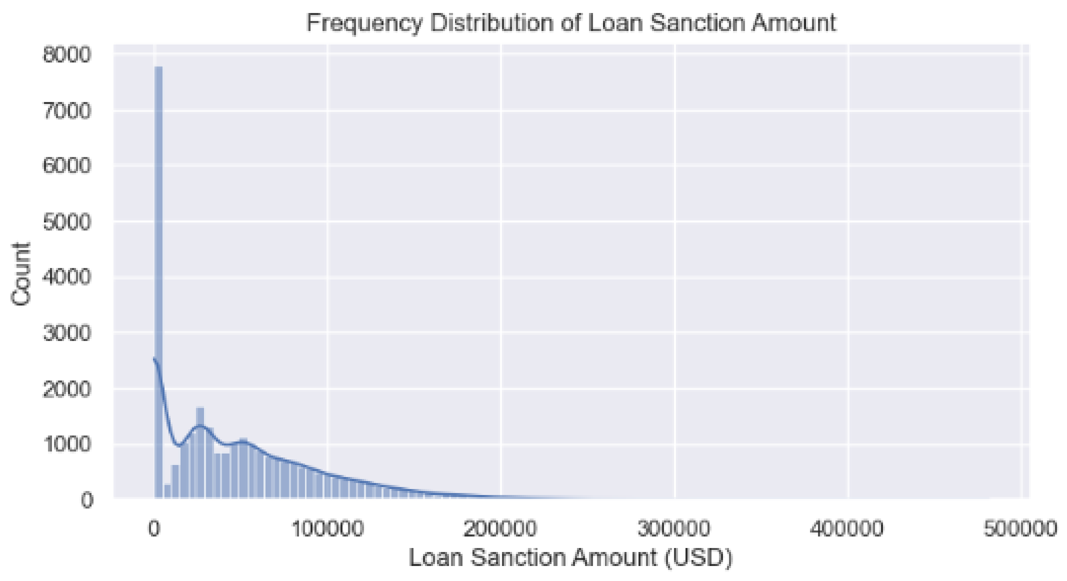
```
int_to_category = {1: 'property 1', 2: 'property 2', 3: 'property 3', 4: 'property 4'}
# Changing the values
train_dataset['Property Type'].replace(int_to_category, inplace=True)
test_dataset['Property Type'].replace(int_to_category, inplace=True)
```

Activity 2.3: Handling Imbalance Data

With the help of Histogram Plot, outliers are visualized. And here we are going to find upper bound and lower bound of Na_to_K feature with some mathematical formula.

From the below diagram, we could visualize that Na_to_K feature has outliers. Histogram Plot from seaborn library is used here.

```
# Histogram Plot
plt.figure(figsize=(8,4))
sns.histplot(train_dataset['Loan Sanction Amount (USD)'], kde=True)
plt.title('Frequency Distribution of Loan Sanction Amount')
plt.show()
```

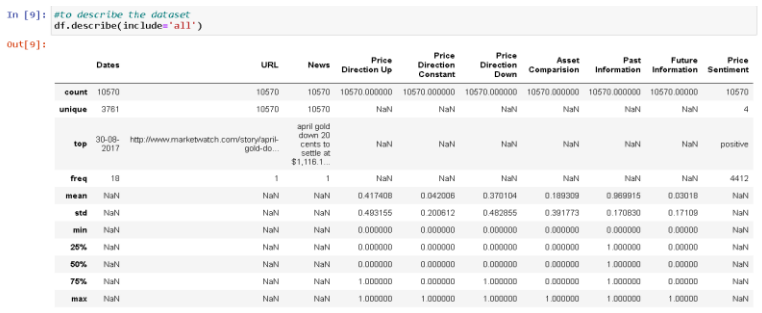


Exploratory Data Analysis

Exploratory Data Analysis (EDA) is a crucial step in data analysis that focuses on understanding and visualizing the data to extract meaningful insights. By examining the dataset, its structure, and the relationships between variables, EDA helps identify patterns, outliers, and trends. Through summary statistics, visualizations, and statistical techniques, EDA enables data scientists to gain a deeper understanding of the data's distribution, central tendencies, and dispersion. It also helps in identifying missing values, outliers, or anomalies that may impact subsequent analyses. EDA serves as a foundation for hypothesis generation, feature selection, and model building, guiding the data analysis process towards more informed decision-making.

Descriptive Analysis

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has 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.



```
In [9]: #to describe the dataset
        df.describe(include='all')
```

Out[9]:

	Dates	URL	News	Price Direction Up	Price Direction Constant	Price Direction Down	Asset Comparision	Past Information	Future Information	Price Sentiment
count	10570	10570	10570	10570.000000	10570.000000	10570.000000	10570.000000	10570.000000	10570.00000	10570
unique	3761	10570	10570	NaN	NaN	NaN	NaN	NaN	NaN	4
top	30-08-2017	http://www.marketwatch.com/story/april-gold-do...	april gold down 20 cents to settle at $1,116.1...	NaN	NaN	NaN	NaN	NaN	NaN	positive
freq	18	1	1	NaN	NaN	NaN	NaN	NaN	NaN	4412
mean	NaN	NaN	NaN	0.417408	0.042006	0.370104	0.189309	0.969915	0.03018	NaN
std	NaN	NaN	NaN	0.493155	0.200612	0.482855	0.391773	0.170830	0.17109	NaN
min	NaN	NaN	NaN	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	NaN
25%	NaN	NaN	NaN	0.000000	0.000000	0.000000	0.000000	1.000000	0.00000	NaN
50%	NaN	NaN	NaN	0.000000	0.000000	0.000000	0.000000	1.000000	0.00000	NaN
75%	NaN	NaN	NaN	1.000000	0.000000	1.000000	0.000000	1.000000	0.00000	NaN
max	NaN	NaN	NaN	1.000000	1.000000	1.000000	1.000000	1.000000	1.00000	NaN

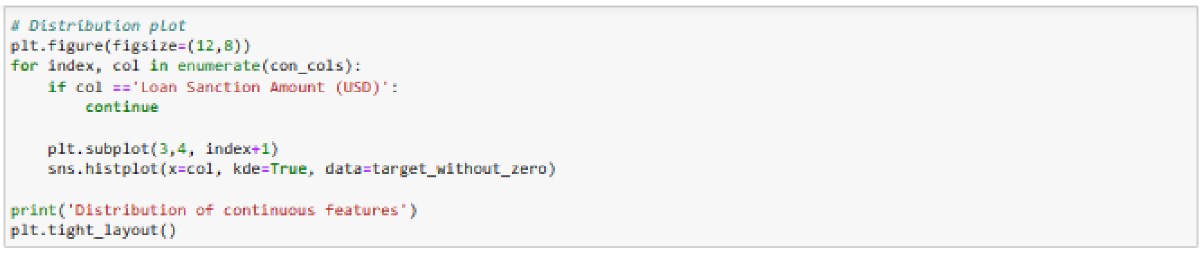
Visual Analysis

Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

Activity 2.1: Univariate analysis

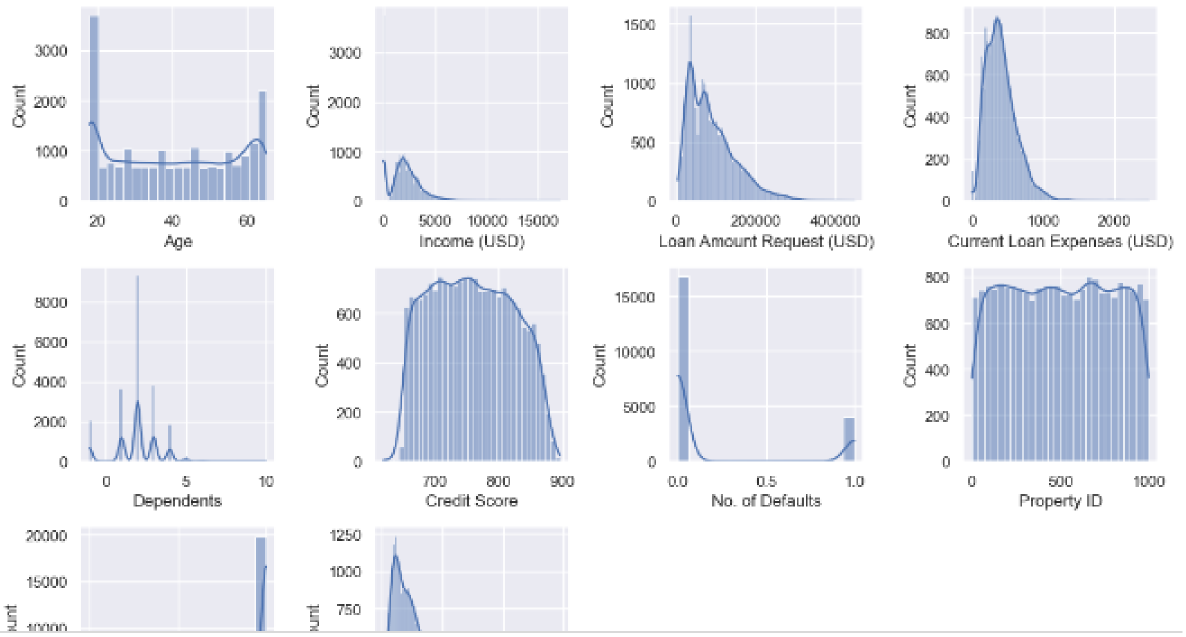
In simple words, univariate analysis is understanding the data with single feature. Here we have displayed two different graphs such as distplot and countplot.

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 subplot.



```
# Distribution plot
plt.figure(figsize=(12,8))
for index, col in enumerate(con_cols):
    if col =='Loan Sanction Amount (USD)':
        continue

    plt.subplot(3,4, index+1)
    sns.histplot(x=col, kde=True, data=target_without_zero)

print('Distribution of continuous features')
plt.tight_layout()
```

```
Distribution of continuous features
```

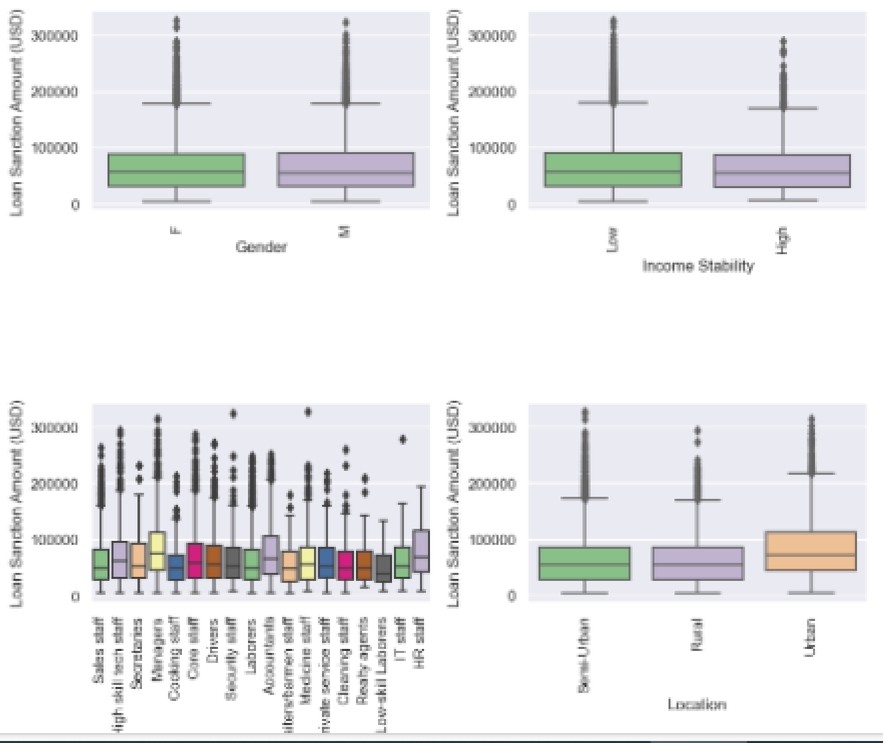


Activity 2.2: Bivariate analysis

To find the relation between two features we use bivariate analysis. Here we are visualizing the relationship between price sentimnet & price direction up, price sentimnet & past Infromation.

Countplot is used here. As a 1st parameter we are passing x value and as a 2nd parameter we are passing hue value.

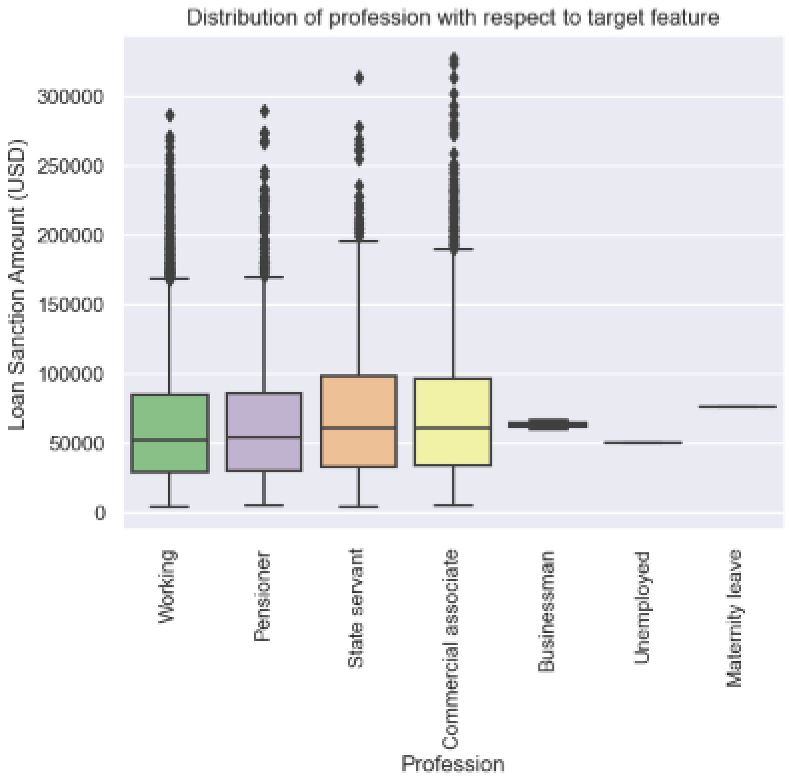
```
# Relation of categorical columns with target feature
plt.figure(figsize=(14,16))
for index, col in enumerate(cat_cols):
    plt.subplot(4,3, index+1)
    sns.boxplot(x=col, y='Loan Sanction Amount (USD)', data=target_without_zero, palet
    plt.xticks(rotation=90)

plt.title(' Relation of Categorical features with Target')
plt.tight_layout()
```



Activity 2.3: Multivariate analysis

In simple words, multivariate analysis is to find the relation between multiple features. Here we have used boxplot from seaborn package.

```
# Profession feature looks different.
# Box-plot for Profession feature
sns.boxplot(x='Profession', y='Loan Sanction Amount (USD)', data=target_without_zero, palette='Accent')
plt.title('Distribution of profession with respect to target feature')
plt.xticks(rotation=90)
plt.show()
```



Splitting data into train and test

Now let's split the Dataset into train and test sets. First split the dataset into x and y and then split the data set
Here x and y variables are created. On x variable, df is passed with dropping the target variable. And on y target variable is passed. For splitting training and testing data we are using train_test_split() function from sklearn. As parameters, we are passing x, y, test_size, random_state.

```
from sklearn.model_selection import train_test_split
train_set, validation_set = train_test_split(train_dataset, test_size=0.2, random_state=10)

# Checking our split
info = ['mean', 'std']
print('Train Set :')
display(train_set.describe().loc[info] ) #  For train set
print('Validation Set :')
display(validation_set.describe().loc[info])  # For validation set
```

```
Train Set :
```

	Age	Income (USD)	Loan Amount Request (USD)	Current Loan Expenses (USD)	Dependents	Credit Score	No. of Defaults	Property ID	Co-Applicant	Property Price	Loan Sanction Amount (USD)
mean	40.122202	2159.253950	88706.379386	406.383804	1.983552	740.164741	0.193491	502.622791	0.852493	133142.729065	47837.914376
std	16.038083	1562.082077	59567.597079	217.330070	1.275607	72.254031	0.395043	288.208643	0.354618	93131.610303	48089.872848

```
Validation Set :
```

	Age	Income (USD)	Loan Amount Request (USD)	Current Loan Expenses (USD)	Dependents	Credit Score	No. of Defaults	Property ID	Co-Applicant	Property Price	Loan Sanction Amount (USD)
mean	40.130518	2168.510583	88907.785123	405.569829	2.017972	739.392433	0.195428	499.04973	0.852207	133097.771960	48571.556205
std	16.162484	1546.819036	58066.681492	208.468388	1.262090	71.688054	0.396565	287.79345	0.354925	89432.221395	48005.587844

Model Building

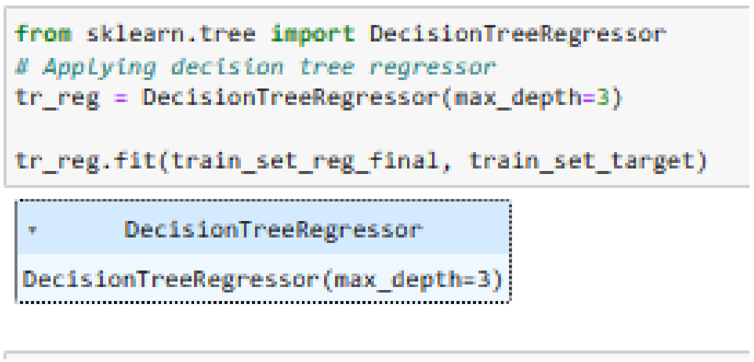
Machine Learning Model Building is the process of creating predictive or analytical models using machine learning algorithms. It involves data preprocessing, feature engineering, selecting an appropriate algorithm, training the model, evaluating its performance, and iterating to improve accuracy and robustness.

Training The Model In Multiple Algorithms

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 three classification algorithms. The best model is saved based on its performance.

Activity 1.1: Decision Tree Model

A function named decision Tree is created and train and test data are passed as the parameters. Inside the function, Decision Tree Classifier algorithm is initialised and training data is passed to the model with the .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.

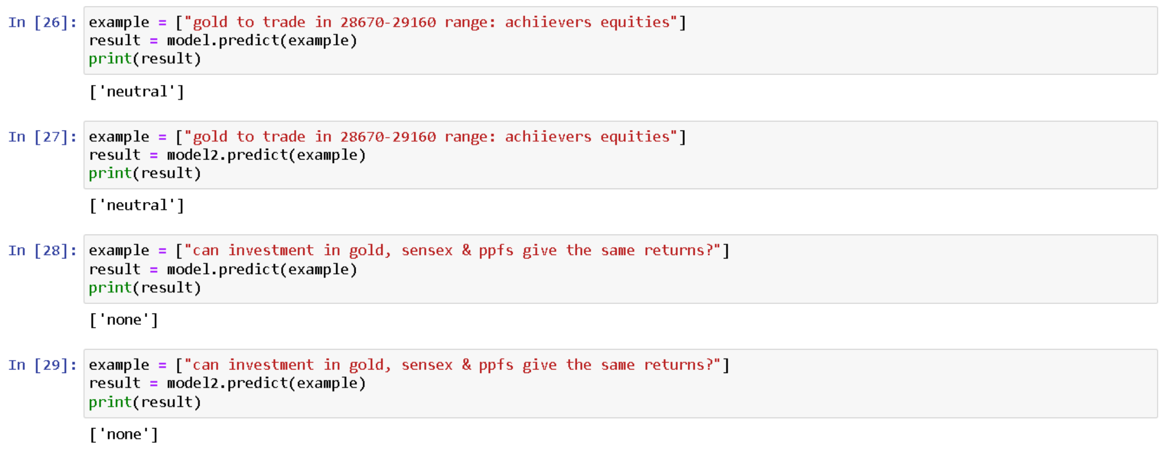


```
from sklearn.tree import DecisionTreeRegressor
# Applying decision tree regressor
tr_reg = DecisionTreeRegressor(max_depth=3)

tr_reg.fit(train_set_reg_final, train_set_target)
```

```
▾      DecisionTreeRegressor
DecisionTreeRegressor(max_depth=3)
```

Testing The Model

Here we have tested with Logistic regression and Svm algorithms. With the help of predict() function.



```
In [26]: example = ["gold to trade in 28670-29160 range: achiievers equities"]
         result = model.predict(example)
         print(result)

         ['neutral']

In [27]: example = ["gold to trade in 28670-29160 range: achiievers equities"]
         result = model2.predict(example)
         print(result)

         ['neutral']

In [28]: example = ["can investment in gold, sensex & ppfs give the same returns?"]
         result = model.predict(example)
         print(result)

         ['none']

In [29]: example = ["can investment in gold, sensex & ppfs give the same returns?"]
         result = model2.predict(example)
         print(result)

         ['none']
```

Performance Testing

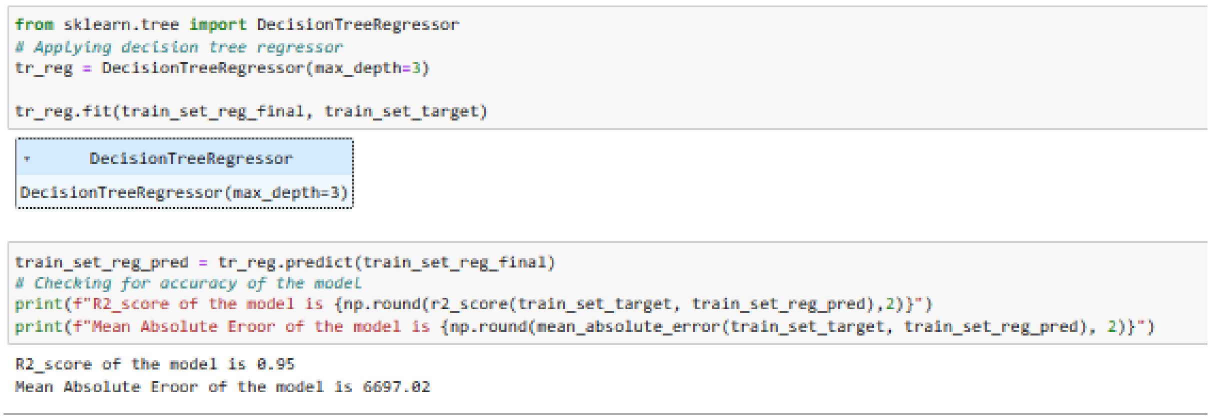
Machine Learning Performance Testing evaluates the accuracy and effectiveness of trained models by comparing their predictions against known values. It measures performance using metrics like accuracy, precision, recall, and identifies issues like overfitting or underfitting. It helps in selecting the best model and improving its overall performance for real-world applications.

Testing Model With Multiple Evaluation Metrics

Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using evaluation metrics for classification tasks including accuracy, precision, recall, support and F1-score.

Activity 1.1: Compare the model

For comparing the below two models, with their accuracy score

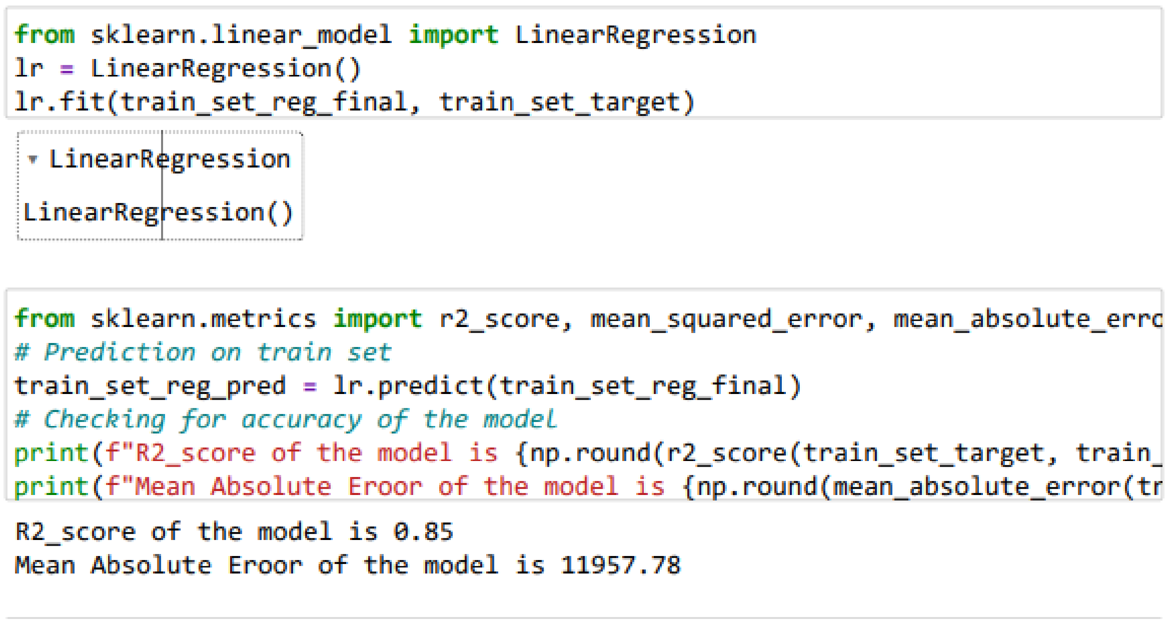


```
from sklearn.tree import DecisionTreeRegressor
# Applying decision tree regressor
tr_reg = DecisionTreeRegressor(max_depth=3)

tr_reg.fit(train_set_reg_final, train_set_target)
```

```
▾      DecisionTreeRegressor
DecisionTreeRegressor(max_depth=3)
```

```
train_set_reg_pred = tr_reg.predict(train_set_reg_final)
# Checking for accuracy of the model
print(f"R2_score of the model is {np.round(r2_score(train_set_target, train_set_reg_pred),2)}")
print(f"Mean Absolute Eroor of the model is {np.round(mean_absolute_error(train_set_target, train_set_reg_pred), 2)}")
```

```
R2_score of the model is 0.95
Mean Absolute Eroor of the model is 6697.02
```



```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(train_set_reg_final, train_set_target)
```

```
▾ LinearRegression
LinearRegression()
```

```
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_errc
# Prediction on train set
train_set_reg_pred = lr.predict(train_set_reg_final)
# Checking for accuracy of the model
print(f"R2_score of the model is {np.round(r2_score(train_set_target, train_
print(f"Mean Absolute Eroor of the model is {np.round(mean_absolute_error(tr
```

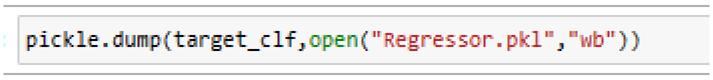
```
R2_score of the model is 0.85
Mean Absolute Eroor of the model is 11957.78
```

Model Deployment

Flask Model Deployment involves deploying a saved machine learning model using the Flask framework. It includes initializing a Flask application, loading the saved model, defining routes and endpoints, and deploying the application on a web server or cloud platform. This enables users to interact with the deployed model in real-time via a browser or API calls.

Save The Best Model

Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future.



```
pickle.dump(target_clf,open("Regressor.pkl","wb"))
```

Integrate With Web Framework

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 server-side script

Run the web application

Activity 2.1: Building Html Pages:
For this project create two HTML files namely

home.html

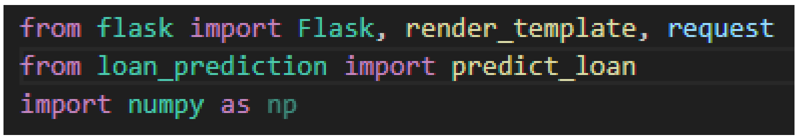
jointreport.html

Loan_Application.html

Report.html
and save them in the templates folder. Refer this link for templates.

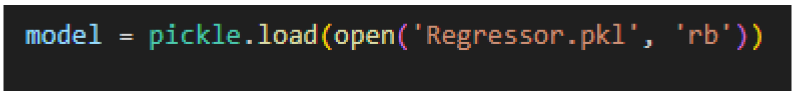
Activity 2.2: Build Python code:

Import the libraries



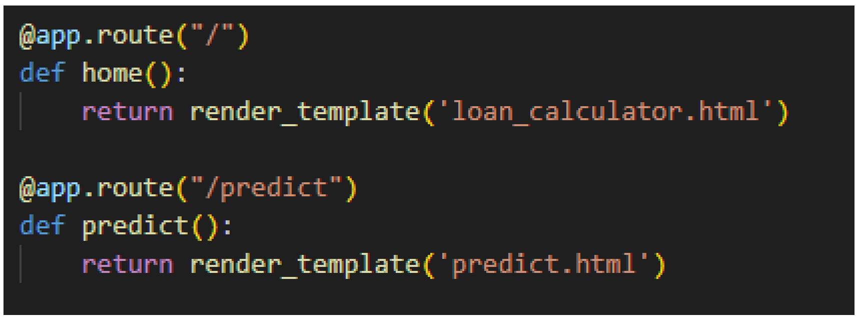
```python
from flask import Flask, render_template, request
from loan_prediction import predict_loan
import numpy as np
```

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 argument.



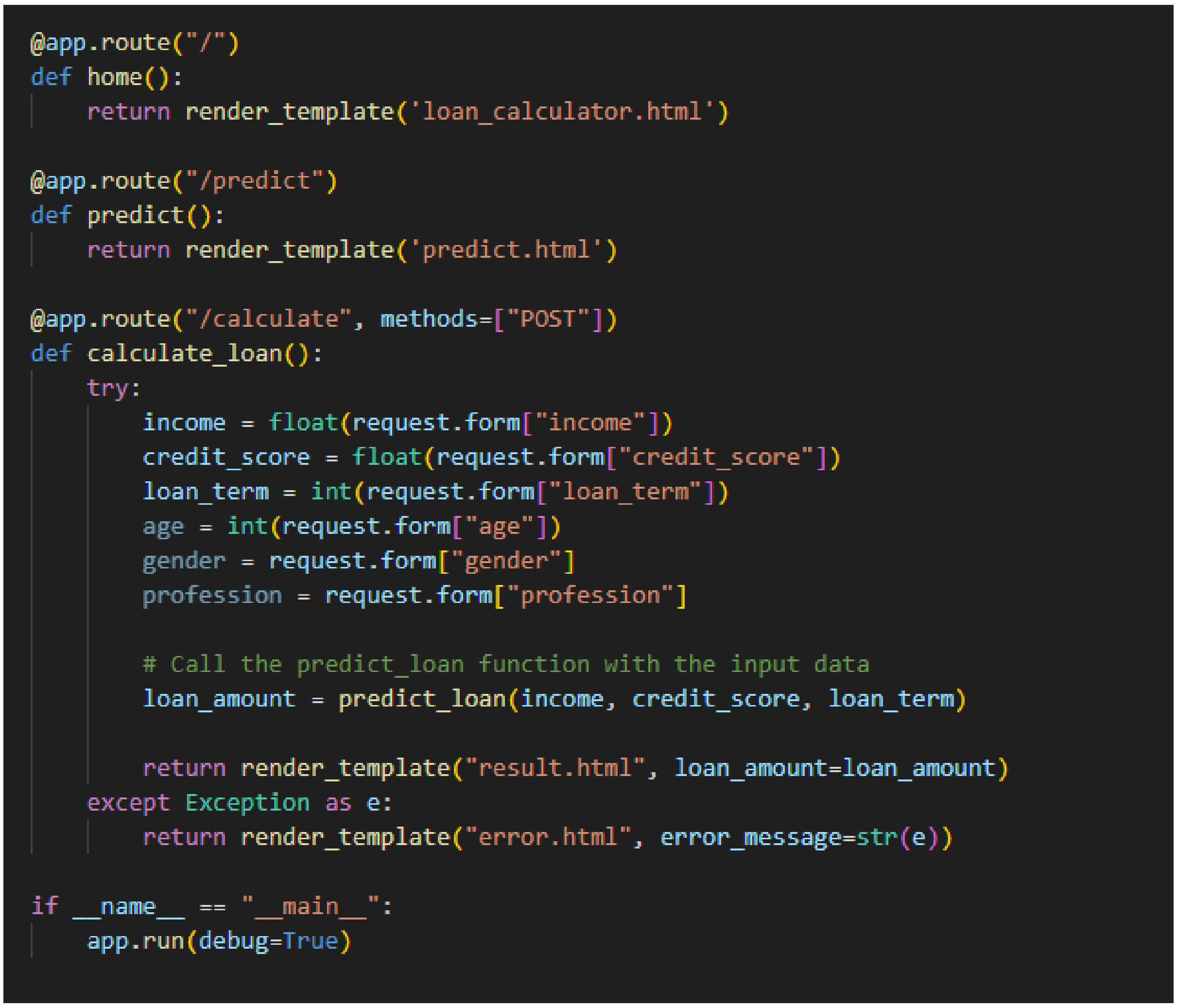
```python
model = pickle.load(open('Regressor.pkl', 'rb'))
```

Render HTML page:



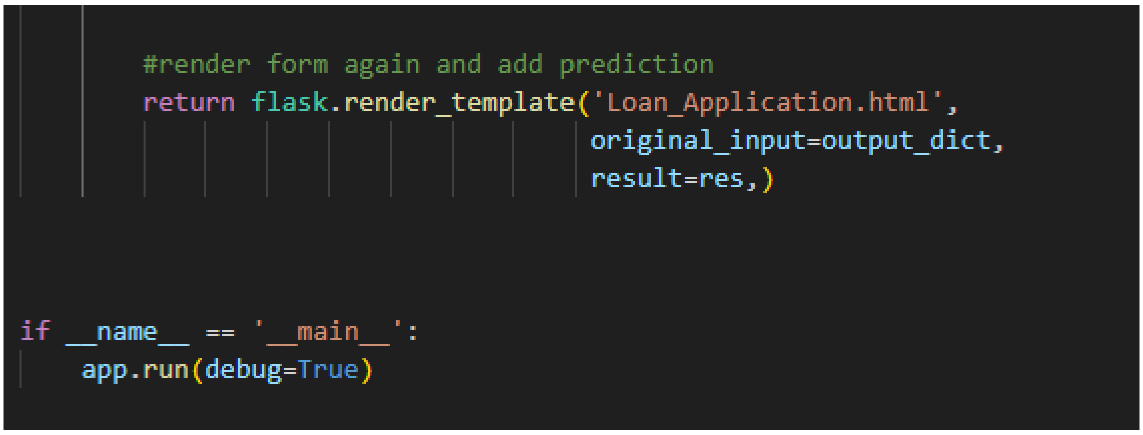
```python
@app.route("/")
def home():
    return render_template('loan_calculator.html')

@app.route("/predict")
def predict():
    return render_template('predict.html')
```

Here we will be using a declared constructor to route to the HTML page which we have created earlier.
In the above example, '/' URL is bound with the index.html function. Hence, when the home page of the web server is opened in the 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:



```python
@app.route("/")
def home():
    return render_template('loan_calculator.html')

@app.route("/predict")
def predict():
    return render_template('predict.html')

@app.route("/calculate", methods=["POST"])
def calculate_loan():
    try:
        income = float(request.form["income"])
        credit_score = float(request.form["credit_score"])
        loan_term = int(request.form["loan_term"])
        age = int(request.form["age"])
        gender = request.form["gender"]
        profession = request.form["profession"]

        # Call the predict_loan function with the input data
        loan_amount = predict_loan(income, credit_score, loan_term)

        return render_template("result.html", loan_amount=loan_amount)
    except Exception as e:
        return render_template("error.html", error_message=str(e))

if __name__ == "__main__":
    app.run(debug=True)
```

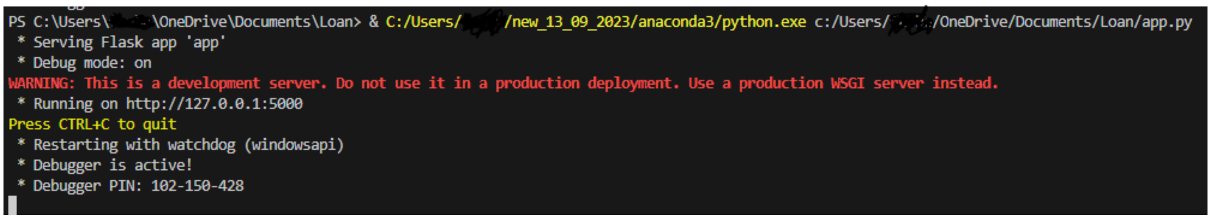
Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will be rendered to the text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



```
        #render form again and add prediction
        return flask.render_template('Loan_Application.html',
                                     original_input=output_dict,
                                     result=res,)


if __name__ == '__main__':
    app.run(debug=True)
```

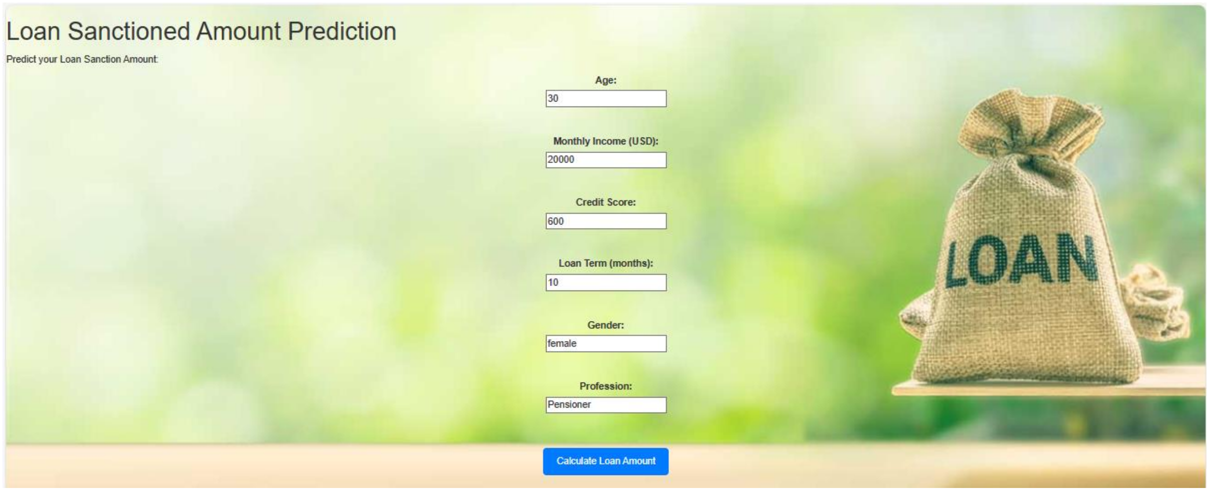
Activity 2.3: Run the web application

- Open 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.



```
PS C:\Users\[redacted]\OneDrive\Documents\Loan> & C:/Users/[redacted]/new_13_09_2023/anaconda3/python.exe c:/Users/[redacted]/OneDrive/Documents/Loan/app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with watchdog (windowsapi)
 * Debugger is active!
 * Debugger PIN: 102-150-428
```

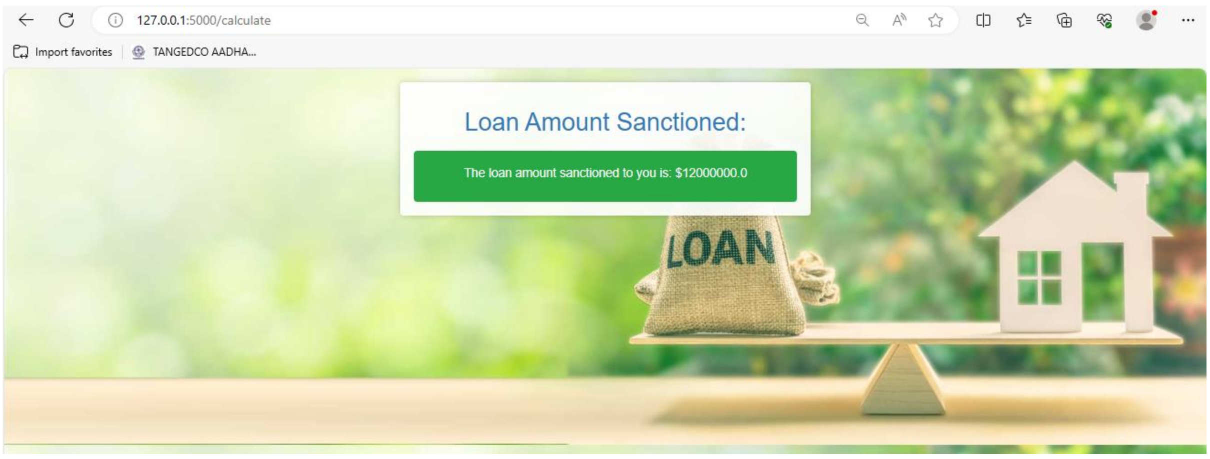
Now, Go the web browser and write the localhost url (http://127.0.0.1:5000) to get the belowresult



Loan Sanctioned Amount Prediction

Predict your Loan Sanction Amount:

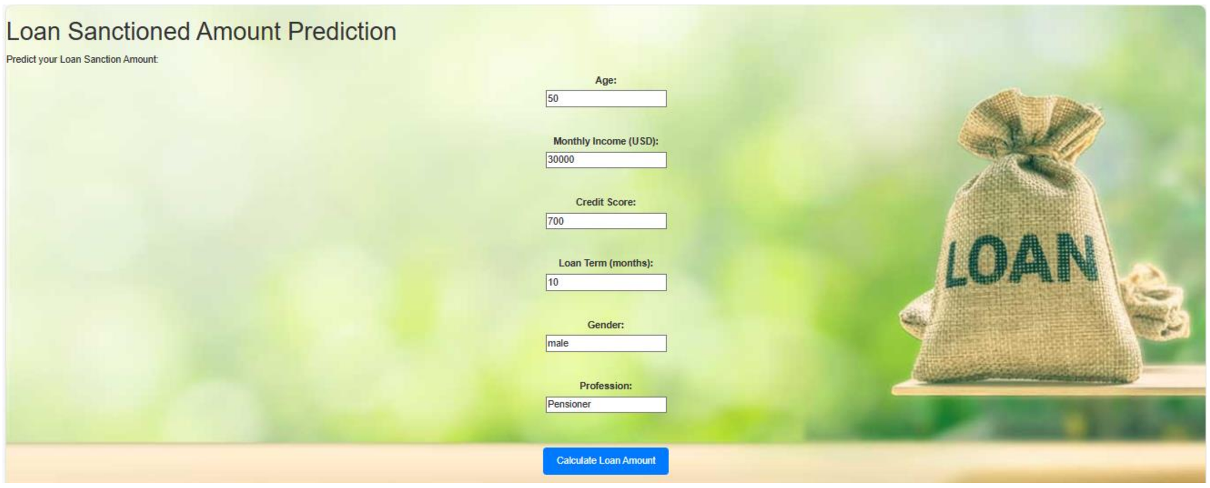
Age:	30
Monthly Income (USD):	20000
Credit Score:	600
Loan Term (months):	10
Gender:	female
Profession:	Pensioner



127.0.0.1:5000/calculate

Loan Amount Sanctioned:

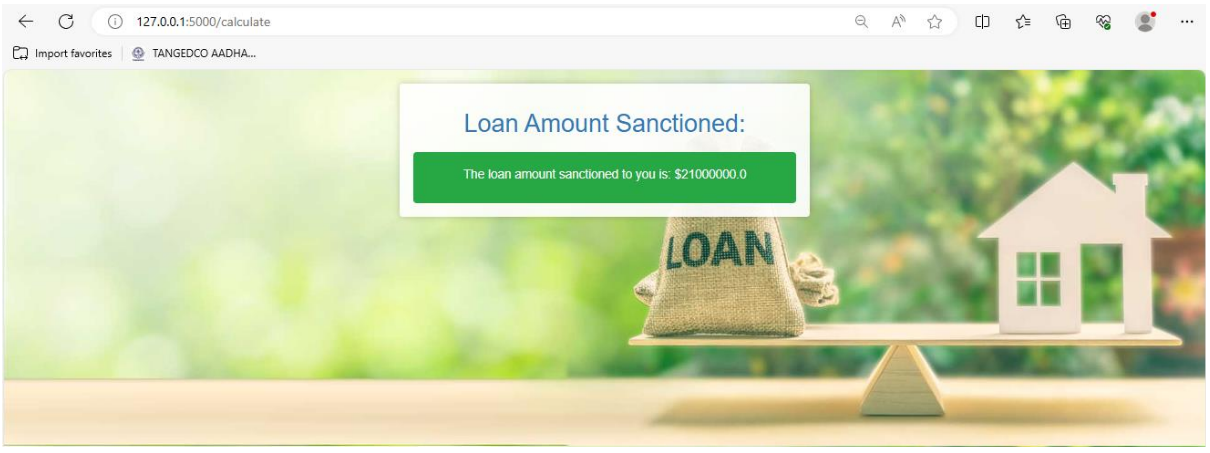
The loan amount sanctioned to you is: $12000000.0



Loan Sanctioned Amount Prediction

Predict your Loan Sanction Amount:

Age:	50
Monthly Income (USD):	30000
Credit Score:	700
Loan Term (months):	10
Gender:	male
Profession:	Pensioner

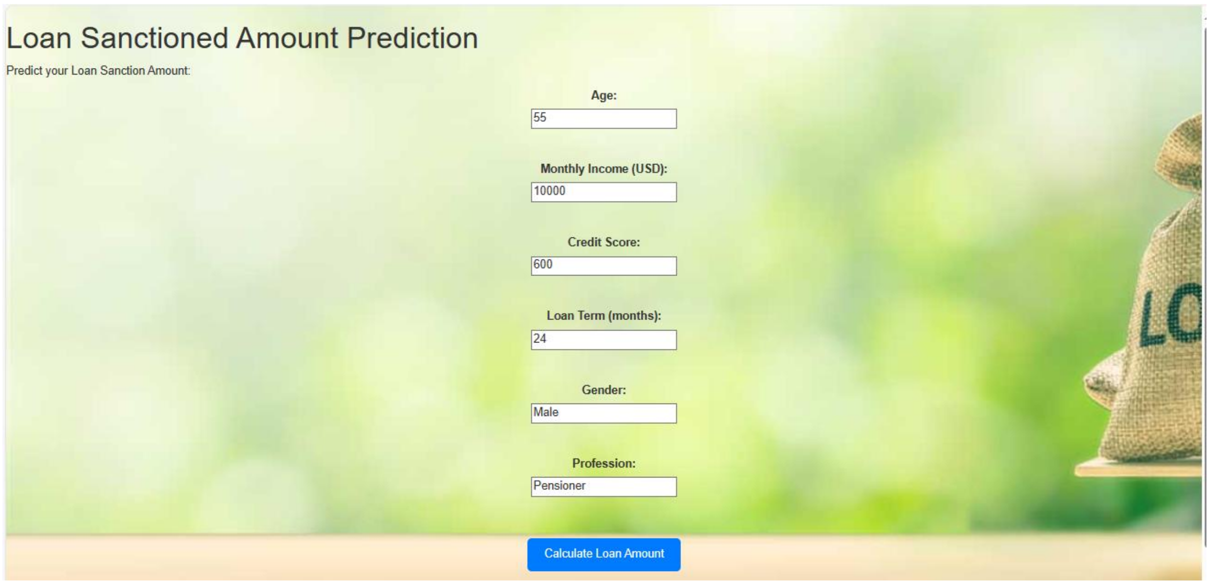


127.0.0.1:5000/calculate

Import favorites | TANGEDCO AADHA...

Loan Amount Sanctioned:

The loan amount sanctioned to you is: $21000000.0



Loan Sanctioned Amount Prediction

Predict your Loan Sanction Amount:

Age: 55

Monthly Income (USD): 10000

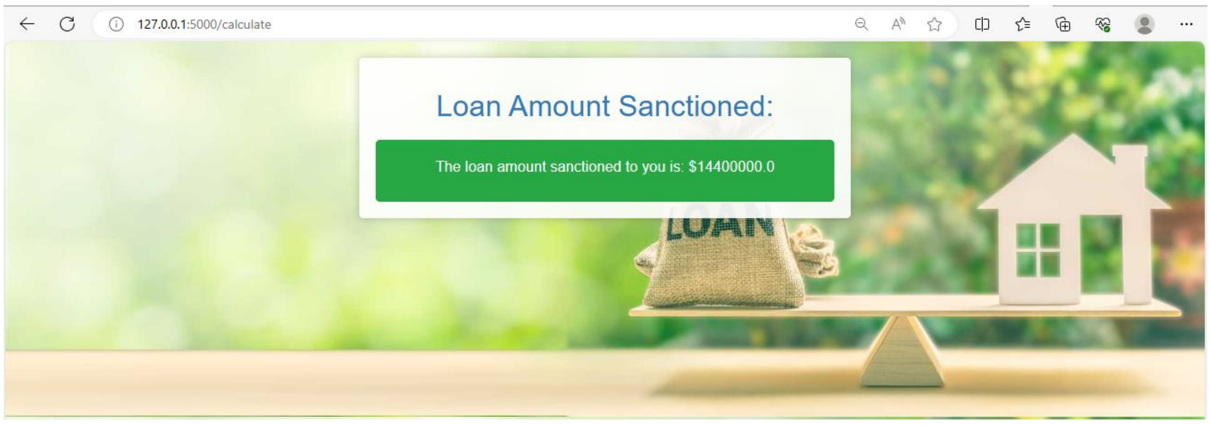
Credit Score: 600

Loan Term (months): 24

Gender: Male

Profession: Pensioner

Calculate Loan Amount



127.0.0.1:5000/calculate

Loan Amount Sanctioned:

The loan amount sanctioned to you is: $14400000.0