Financial Forecasting On Adult's Income Prediction Models

Project Description

Imagine you are a data analyst tasked with predicting the Income in United states across different demographic areas such as country, Gender, Race, Occupation and so on. You have access to a dataset containing detailed information on the different types of occupation, their native countries, and their Work Class they belong to. Your goal is to analyse this data and predict the user earning range of the user for providing the future Financial Investment plannings.

Scenario 1: Personalized Investment Recommendations

A financial advisory firm seeks to enhance its service by offering personalized investment recommendations to their clients. Understanding each client's financial capacity and investment needs is crucial for providing relevant and effective advice. By predicting an individual's income using demographic and occupational data, the firm can categorize clients into different income brackets. This enables the firm to offer customized investment plans that align with the financial goals and capacities of each client, ensuring higher client satisfaction and better investment outcomes.

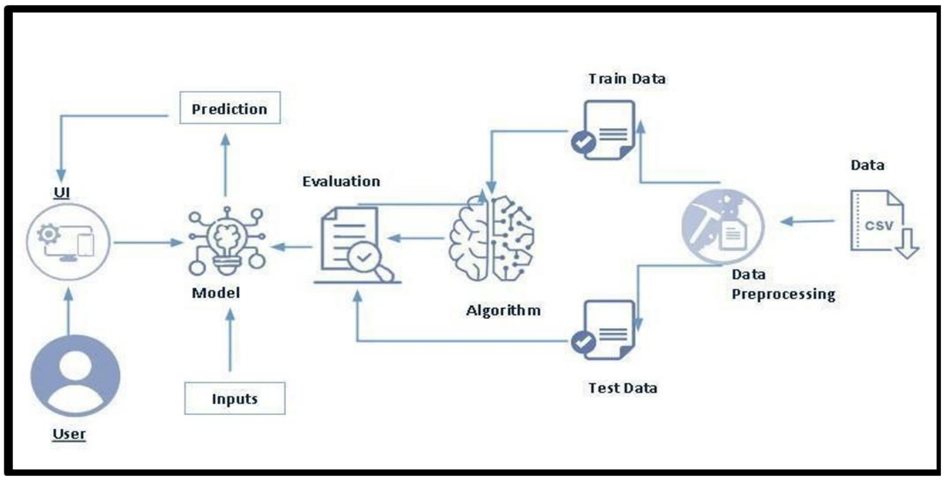
Scenario 2: Targeted Marketing for Investment Products

An investment company plans to launch a new premium investment product aimed at high-income individuals. The company needs to identify and target high-income individuals for their marketing campaigns to maximize the return on investment. Using the income prediction model, the company can analyse its customer database and segment it based on predicted income levels. This allows the marketing team to focus their efforts on individuals who are more likely to have an income exceeding $50,000, thereby increasing the efficiency and effectiveness of their marketing campaigns.

Scenario 3: Optimizing Financial Planning Services

A financial planning service aims to optimize its service offerings based on the income levels of its clients. Different income groups have varying financial planning needs and risk tolerance levels. The prediction model can help the service identify the income range of their clients. With this information, financial planners can develop and offer tailored financial plans that match the specific needs and risk profiles of different income groups, improving client satisfaction and financial outcomes.

Technical Architecture:



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:

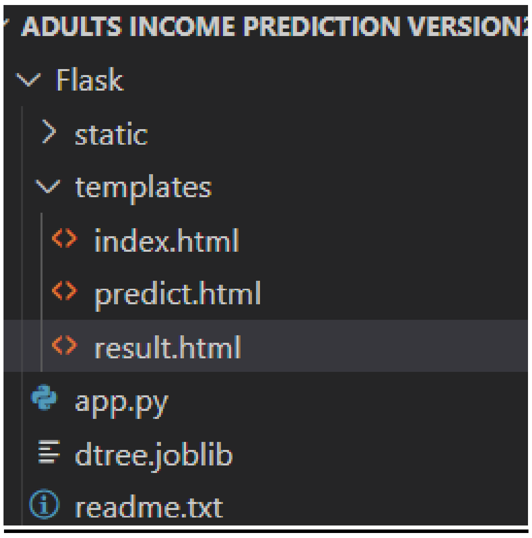
- **Data Collection & Preparation**
  - Collect the dataset
  - Data Preparation
- **Exploratory Data Analysis**
  - Descriptive statistical
  - Visual Analysis
- **Model Building**
  - Training the model in multiple algorithms
  - Testing the model
- **Performance Testing**
  - Testing model with multiple evaluation metrics
  - Comparing model accuracy before & after applying hyperparameter tuning
- **Model Deployment**
  - Save the best model
  - Integrate with Web Framework

Prior Knowledge:

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

- ML Concepts:
- Supervised learning: https://www.javatpoint.com/supervised-machine-learning
- Logistic Regression: https://www.geeksforgeeks.org/understanding-logistic-regression/
- Gradient boosting regressor: https://www.geeksforgeeks.org/ml-gradient-boosting/
- Decision Tree Classifier: https://www.geeksforgeeks.org/decision-tree/
- Flask Basics: https://www.youtube.com/watch?v=lj4I_CvBnt0

Project Structure


```
ADULTS INCOME PREDICTION VERSION
  Flask
    static
    templates
      index.html
      predict.html
      result.html
    app.py
    dtree.joblib
    readme.txt
```

- We are building a flask application which needs HTML pages stored in the Template folder and python script app.py for scripting
- dtree.joblib is our saved model. Further we will use this model for flask integration.
- Training folder contains a model training file.

Milestone 1: 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.

Activity 1: 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.

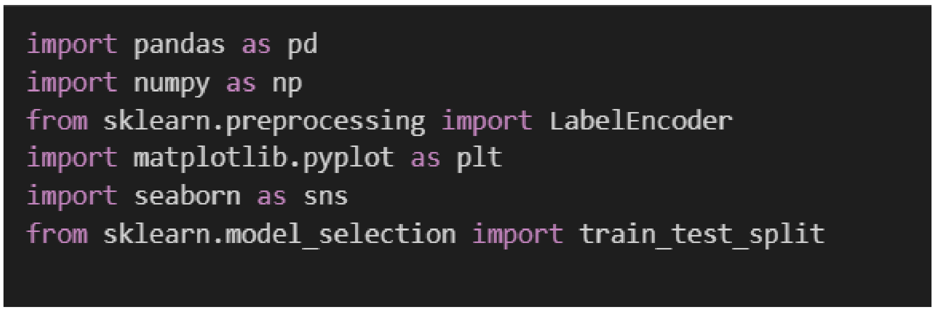
Dataset: LINK

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

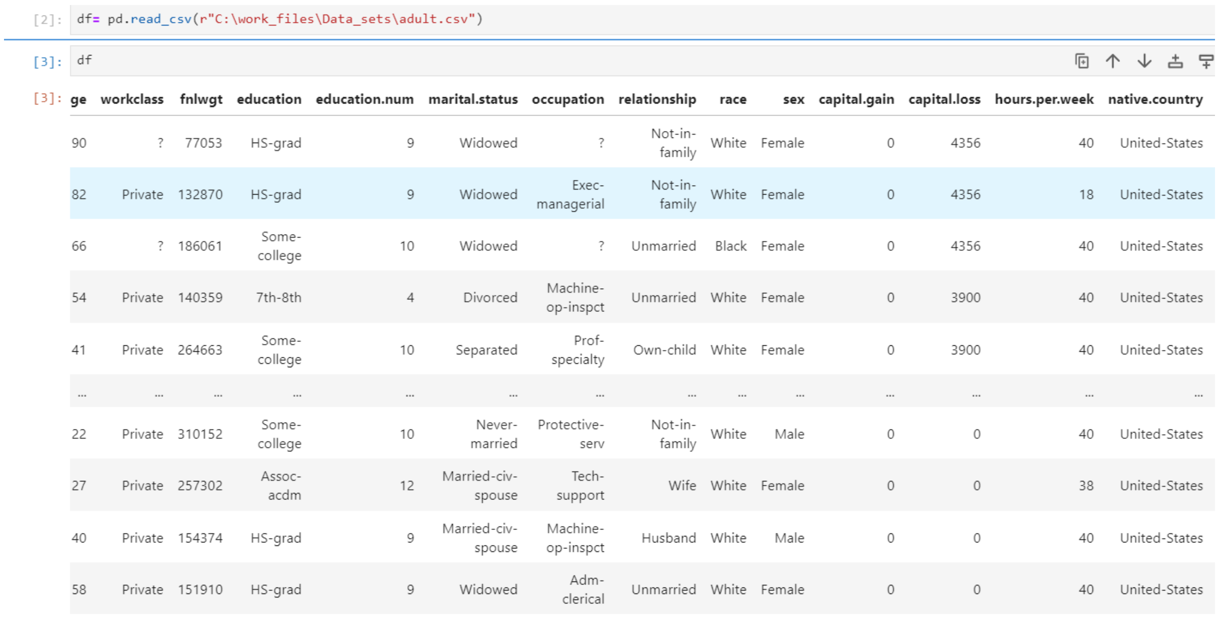
Import the necessary libraries as shown in the image.



```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
```

Activity 1.2: Read the Data set

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.



```
[2]: df= pd.read_csv(r"C:\work_files\Data_sets\adult.csv")
```

```
[3]: df
```

[3]:

ge	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	capital.loss	hours.per.week	native.country
90	?	77053	HS-grad	9	Widowed	?	Not-in-family	White	Female	0	4356	40	United-States
82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White	Female	0	4356	18	United-States
66	?	186061	Some-college	10	Widowed	?	Unmarried	Black	Female	0	4356	40	United-States
54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unmarried	White	Female	0	3900	40	United-States
41	Private	264663	Some-college	10	Separated	Prof-specialty	Own-child	White	Female	0	3900	40	United-States
...	...	...	...	...	...	...	...	...	...	...	...	...	...
22	Private	310152	Some-college	10	Never-married	Protective-serv	Not-in-family	White	Male	0	0	40	United-States
27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female	0	0	38	United-States
40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	0	40	United-States
58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female	0	0	40	United-States

Activity 2: Data Preparation

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

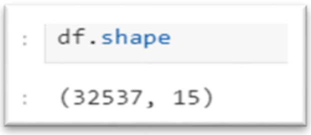
The download data set is not suitable for training the machine learning model as it might have so much randomness and noise .so we need to clean the dataset properly in order to fetch good results.

This activity includes the following steps.

- Handling missing values
- Handling categorical data
- Handling Outliers

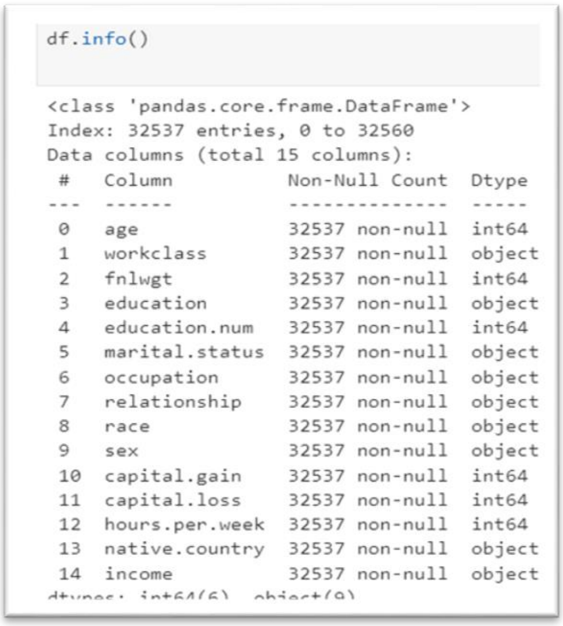
Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

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 is used.



```
df.shape

(32537, 15)
```

Above Figure Describes the Shape of the Dataset i.e, there are 32537 rows and 15 columns including the Target column as well.

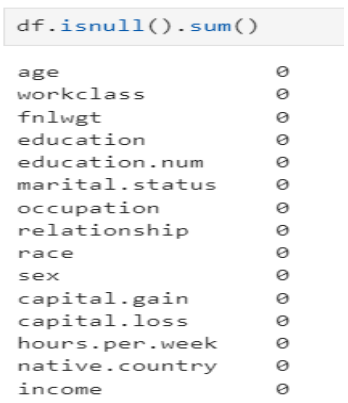


```
df.info()

<class 'pandas.core.frame.DataFrame'>
Index: 32537 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   age             32537 non-null  int64 
 1   workclass       32537 non-null  object
 2   fnlwgt          32537 non-null  int64 
 3   education       32537 non-null  object
 4   education.num   32537 non-null  int64 
 5   marital.status  32537 non-null  object
 6   occupation      32537 non-null  object
 7   relationship    32537 non-null  object
 8   race            32537 non-null  object
 9   sex             32537 non-null  object
 10  capital.gain    32537 non-null  int64 
 11  capital.loss    32537 non-null  int64 
 12  hours.per.week  32537 non-null  int64 
 13  native.country  32537 non-null  object
 14  income          32537 non-null  object
dtypes: int64(6), object(9)
```

df.info() provides the information about the column’s datatype and provides the count of non-null values in the column is concerned.

Dataset do not have any missing values.

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 no null values present in our dataset:



```
df.isnull().sum()
```

```
age               0
workclass         0
fnlwgt            0
education         0
education.num     0
marital.status    0
occupation        0
relationship      0
race              0
sex               0
capital.gain      0
capital.loss      0
hours.per.week    0
native.country    0
income            0
```

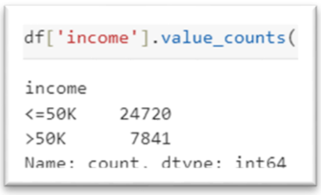
From the above figure we can observe that there are no null values present in the dataset.

Activity 2.2: Handling Categorical Values:

There are multiple categorical columns present in the dataset, they are Work class, Sex, Race, Education, Occupation, Relationship, Martial Status and native country.

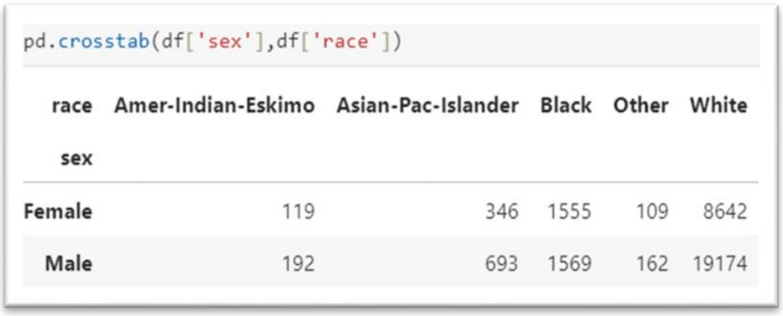
As we know they are no missing values/ null values present in the dataset. We need to know their number of categories present in each column with their counts.

There are several operations to find different insights using categorical values some of the functions are **value_counts**, **cross_tab(), mode**, **replacement of values.**



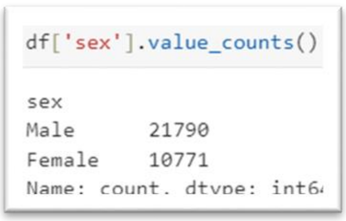
```
df['income'].value_counts(
```

```
income
<=50K     24720
>50K       7841
Name: count, dtype: int64
```



```
pd.crosstab(df['sex'],df['race'])
```

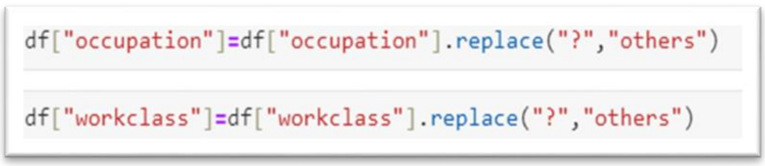
race	Amer-Indian-Eskimo	Asian-Pac-Islander	Black	Other	White
sex					
Female	119	346	1555	109	8642
Male	192	693	1569	162	19174



```
df['sex'].value_counts()
```

```
sex
Male      21790
Female    10771
Name: count, dtype: int6
```

There are few unknown values in the occupation and Work class column representing in "?" so we need to treat them by replaceing them with the general category as "Others" .



```
df["occupation"]=df["occupation"].replace("?","others")
```

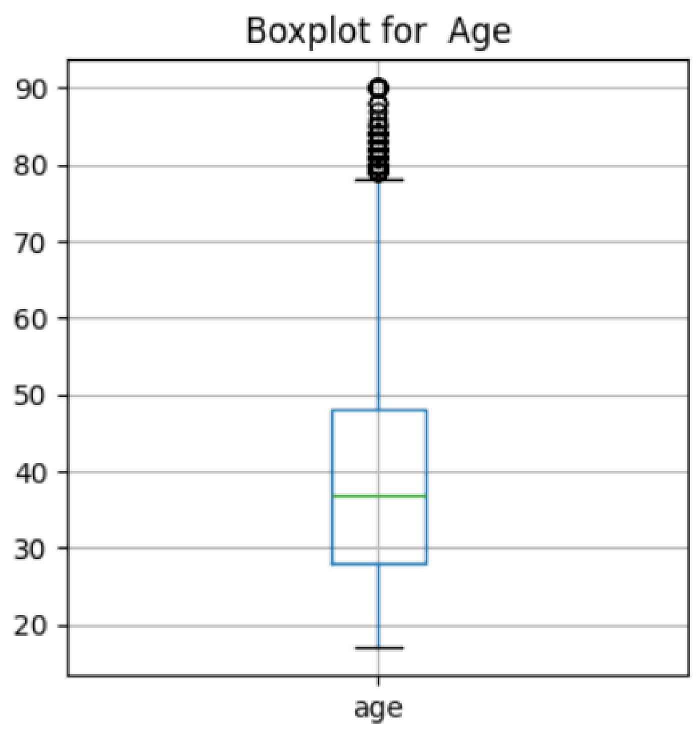
```
df["workclass"]=df["workclass"].replace("?","others")
```

At last we have all the values regarding the categorical column. For the further process process of model building we have to encode the categorical values with numerical values we have used label encoding the process of label encoding can be obtained form the below snippet of code.

```
#encoding
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['income']=le.fit_transform(df['income'])
df['workclass']=le.fit_transform(df['workclass'])
df['education']=le.fit_transform(df['education'])
df['relationship']=le.fit_transform(df['relationship'])
df['marital.status']=le.fit_transform(df['marital.status'])
df['occupation']=le.fit_transform(df['occupation'])
df['race']=le.fit_transform(df['race'])
df['sex']=le.fit_transform(df['sex'])
df['native.country']=le.fit_transform(df['native.country'])
```

Activity 2.3: Treating Outliers:

Outliers are the abnormal data which are away from the range of the distribution of the data of each column in the data. Here we have the box plot to find whether the Outliers present or not but we cannot know how many Outliers present in the Age column.

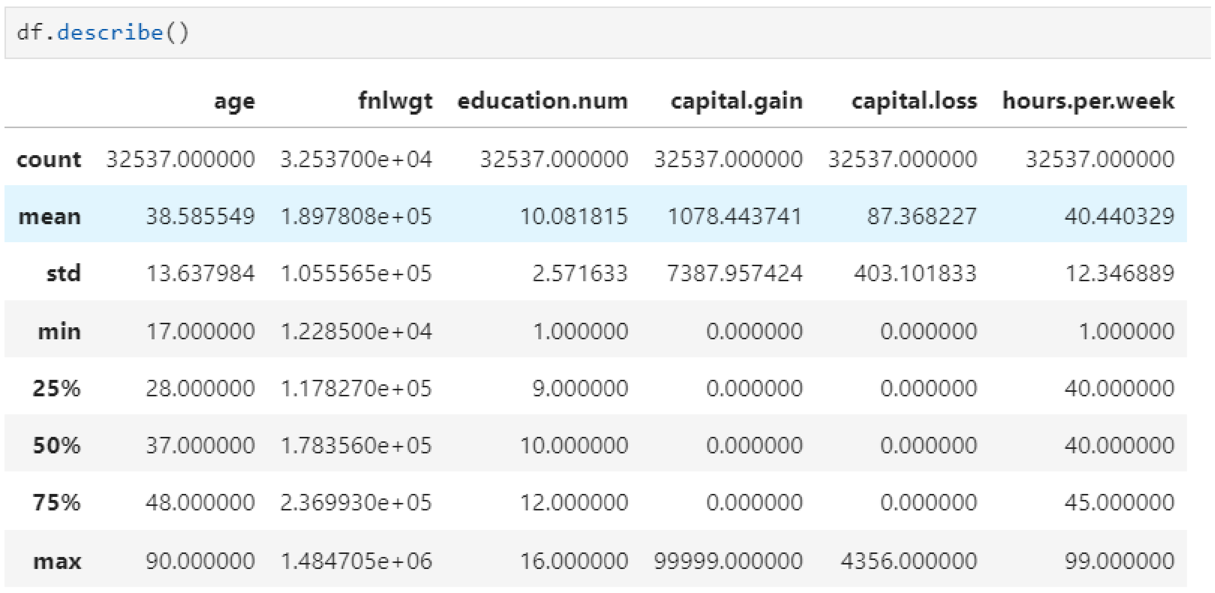


Actually we are preparing the model for the Adults income so adults age ranges from 20 to 90 but we must not consider the person whose age is 90 is not an outlier even though we need not to change or modify the Age column. It is observed that there is no data present below the lower limit of the data.

Milestone 2: Exploratory Data Analysis

Activity 1: Descriptive Analysis

Descriptive analysis involves examining fundamental characteristics of data using statistical methods. It provides insights into the mean, standard deviation, minimum, maximum, and percentile values of continuous features.



```
df.describe()
```

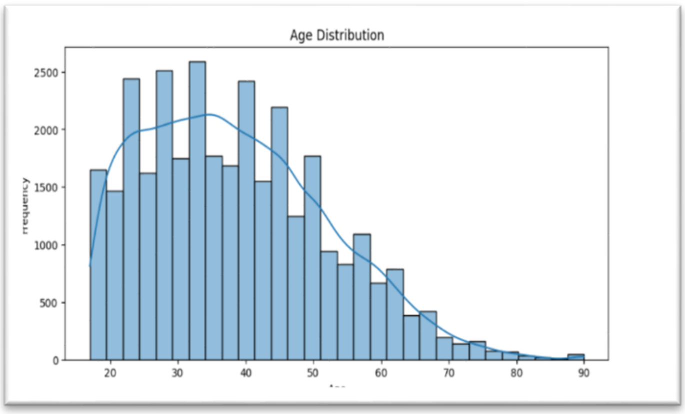
	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.week
count	32537.000000	3.253700e+04	32537.000000	32537.000000	32537.000000	32537.000000
mean	38.585549	1.897808e+05	10.081815	1078.443741	87.368227	40.440329
std	13.637984	1.055565e+05	2.571633	7387.957424	403.101833	12.346889
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.178270e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.783560e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.369930e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

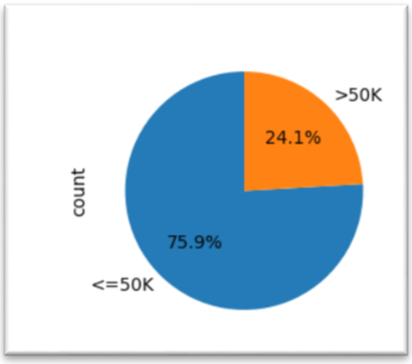
Activity 2: Visual analysis

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.

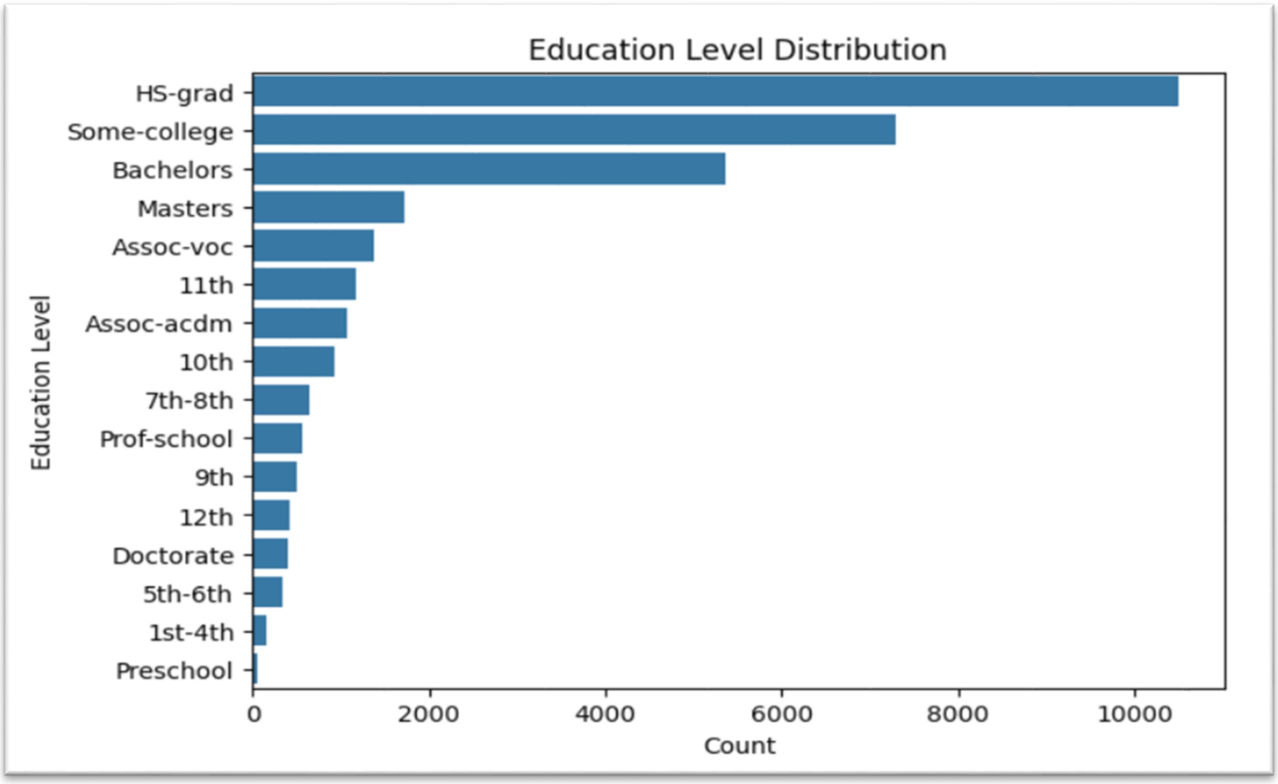
Activity 2.1: univariate analysis

In simple words, univariate analysis is understanding the data with single feature. Here we have displayed a histogram, Pie plot and Horizontal bar plot.





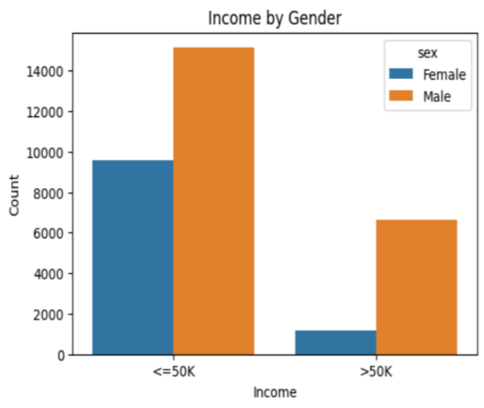
The histogram describes the distribution of Age in the dataset and a KDE plot is being added to find skewness of the age and range of distribution of ages. A pie plot is being displayed it represents the percentage of income earning by the Adults from the Data.

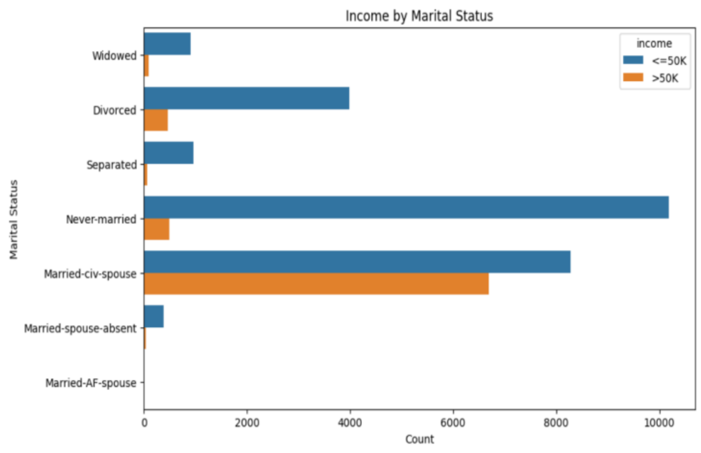


Above Horizontal histogram represents the educational Background of adults.

Activity 2.2: Bivariate analysis

Bivariate analysis is a statistical method that involves the analysis of two variables to determine the empirical relationship between them. Here we have grouped bar plots and Horizontal Grouped Bar charts.

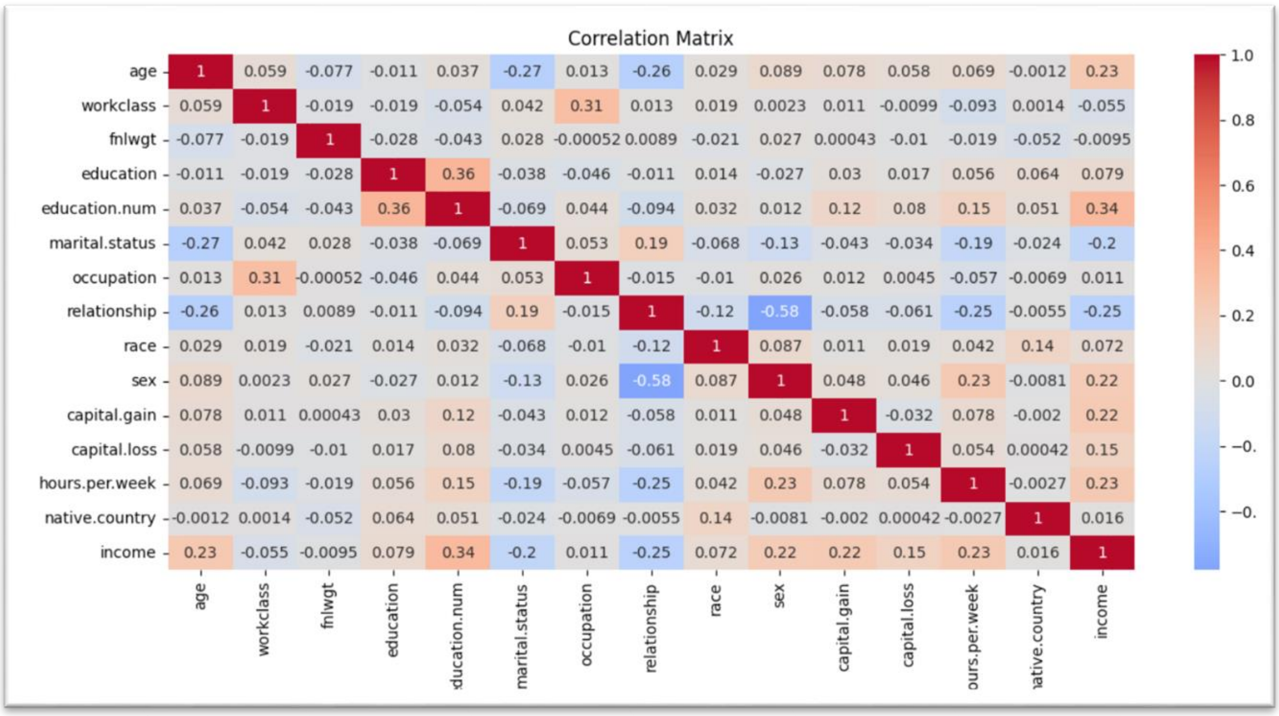




Above Grouped bar char represents the income ranges with respect to the gender is concerned it provides the insights regarding the frequency of male earning on both income categories with represents Blue in colour and female with Orange in colour,
Horizontal Grouped Bar chart represents the earning ranges of by each category of marital status. from the chart blue coloured bar represents the income with <=50k and Orange coloured bar represent the income with >50k.

Activity 2.3: Multi-variate analysis

Multi-variate analysis is a statistical method that involves the analysis more than 2 variables to determine the empirical relationship among them. Here we have a heatmap representing the correlation among the variables in the Data.



Activity 4: 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. “x” represents the whole data columns other than the target column,“y” represents the Target column in the dataset. We need to build the model by giving the training to the model and make the predictions on the test data.so we need to divide the whole dataset into training and testing data.

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.

```
#x and y partition
x=df.drop('income',axis=1)
y=df['income']
```

```
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=100)
```

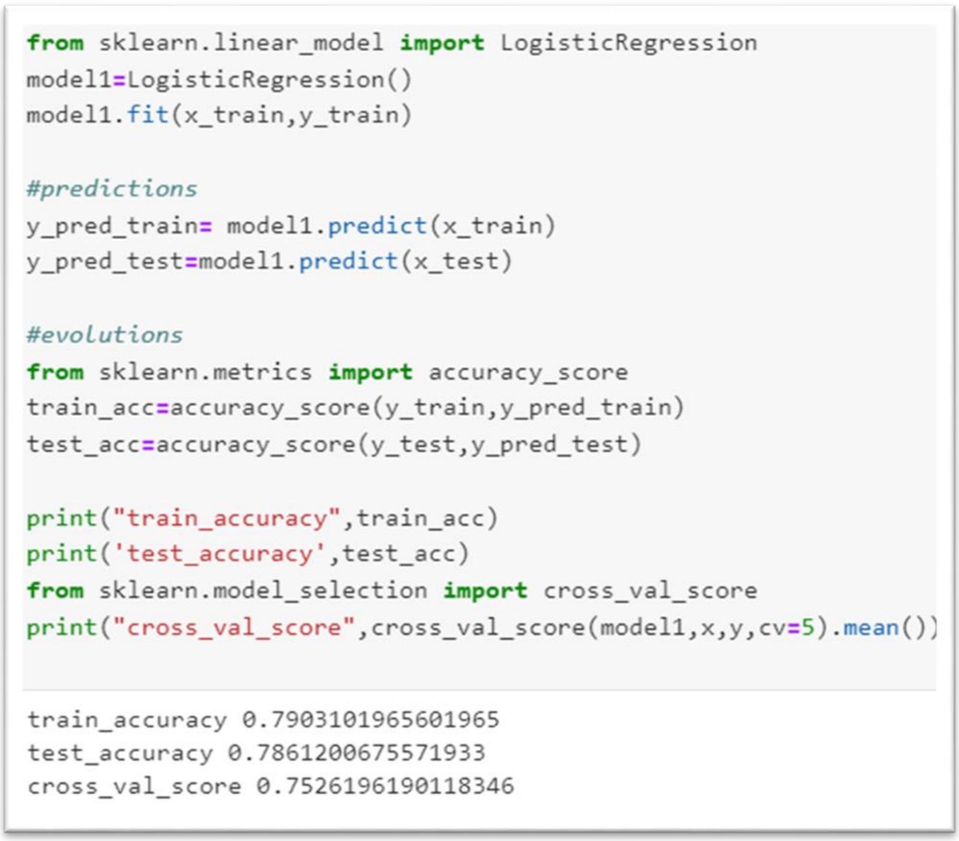
Milestone 3: Model Building

Activity 1: Training and testing the models using 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 classification algorithms. The best model is saved based on its performance.

Activity 2.1 Logistic Regression:

Logistic Regression is the base line model where we import the Logistic Regression from the sklearn library and importing the linear_model and assigning the model to the variable model1 and making the data to fit and moulding the data with respect to the training data.



```
from sklearn.linear_model import LogisticRegression
model1=LogisticRegression()
model1.fit(x_train,y_train)

#predictions
y_pred_train= model1.predict(x_train)
y_pred_test=model1.predict(x_test)

#evolutions
from sklearn.metrics import accuracy_score
train_acc=accuracy_score(y_train,y_pred_train)
test_acc=accuracy_score(y_test,y_pred_test)

print("train_accuracy",train_acc)
print('test_accuracy',test_acc)
from sklearn.model_selection import cross_val_score
print("cross_val_score",cross_val_score(model1,x,y,cv=5).mean())
```

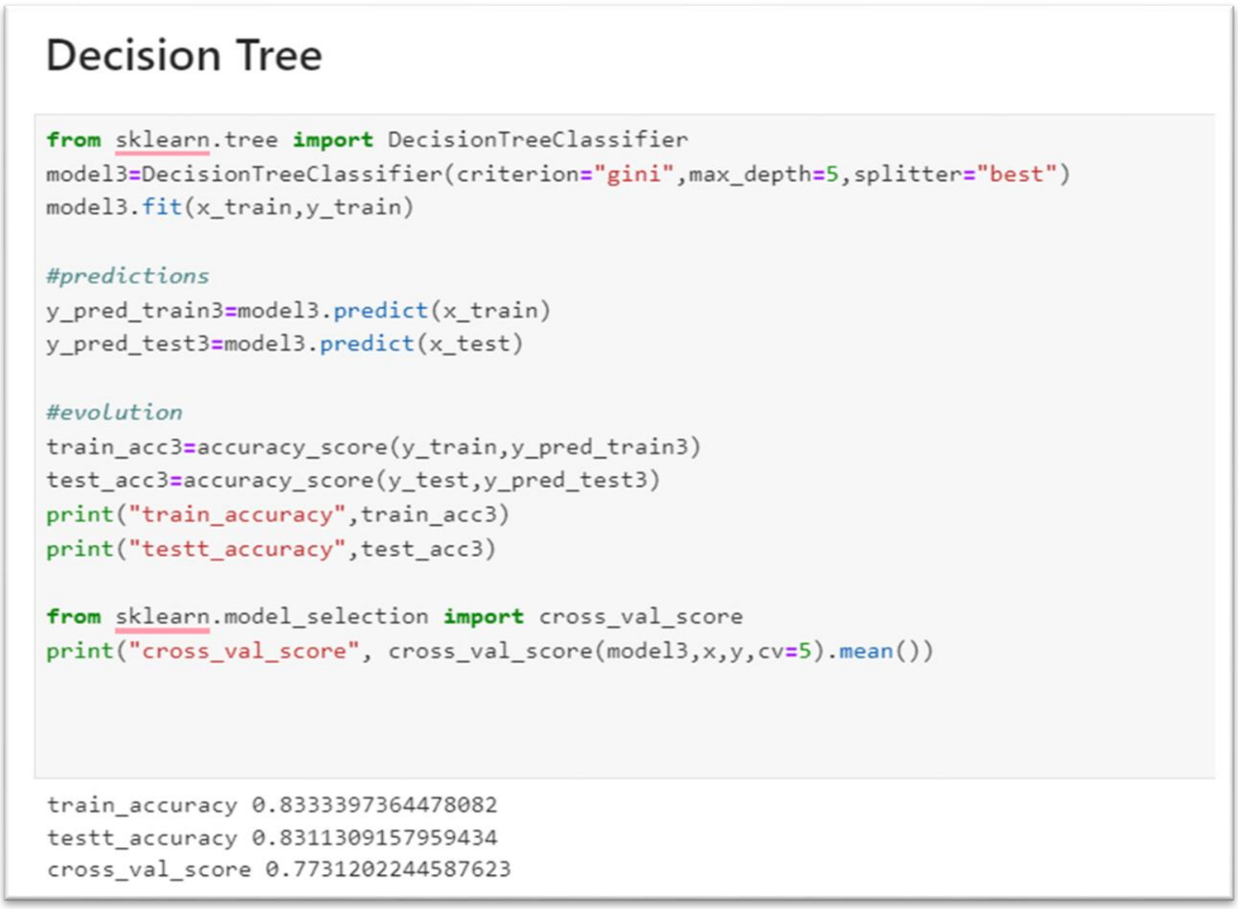
```
train_accuracy 0.7903101965601965
test_accuracy 0.7861200675571933
cross_val_score 0.7526196190118346
```

We make the predictions on the train and test data using predict function and assigning the predicted values to the y_pred_train and y_pred_test

We are calculating the accuracy scores on how the model is working on the data and we use cross-validations to reduce the Bias and trade condition to the dataset. Actually we are able to divide the dataset based on the chosen test size. If CV=4. then we are training the model with 75% of data and we are testing with 25% of data. If CV=5 then we are training the model with 80% of data and testing the model with 20% of data.

Activity 2.2: Decision Tree Classifier:

Decision Tree Classifier and regression algorithm is initialized and the training data is passed to the model and assigning the variable as model3 with the `.fit()` function. Test data is predicted with model3`.predict()` function and saved in a new variable. For evaluating the model, accuracy is calculated



Decision Tree

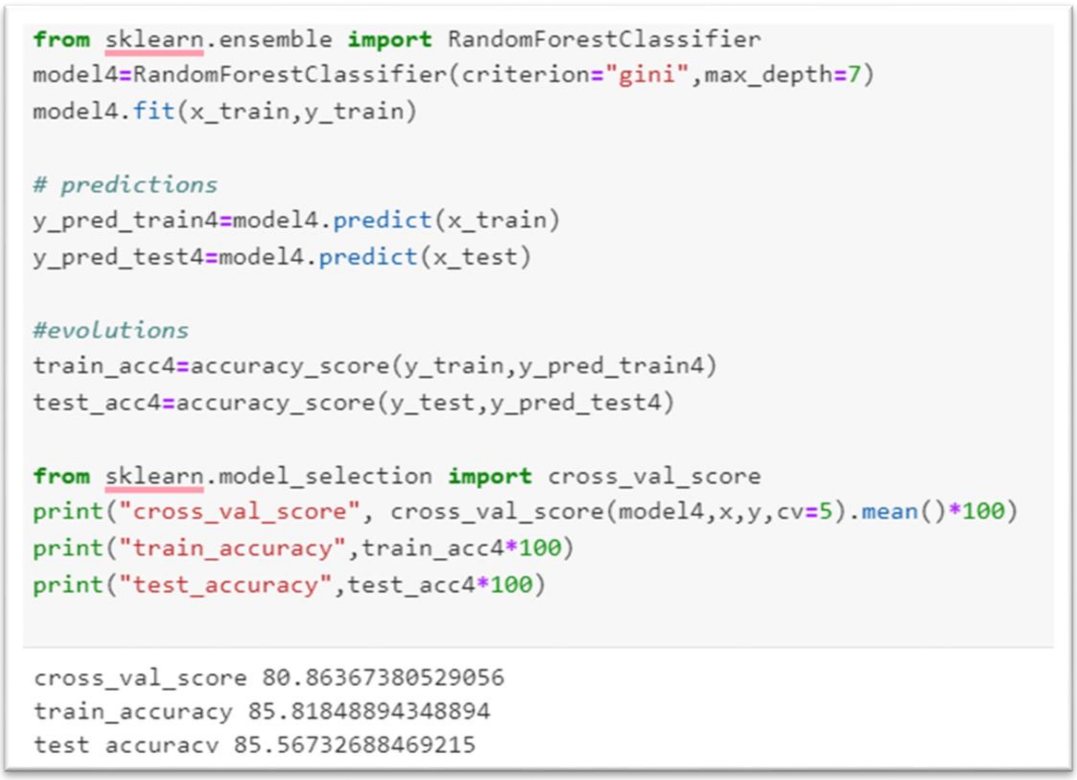
```
from sklearn.tree import DecisionTreeClassifier
model3=DecisionTreeClassifier(criterion="gini",max_depth=5,splitter="best")
model3.fit(x_train,y_train)

#predictions
y_pred_train3=model3.predict(x_train)
y_pred_test3=model3.predict(x_test)

#evolution
train_acc3=accuracy_score(y_train,y_pred_train3)
test_acc3=accuracy_score(y_test,y_pred_test3)
print("train_accuracy",train_acc3)
print("testt_accuracy",test_acc3)

from sklearn.model_selection import cross_val_score
print("cross_val_score", cross_val_score(model3,x,y,cv=5).mean())
```

```
train_accuracy 0.8333397364478082
testt_accuracy 0.8311309157959434
cross_val_score 0.7731202244587623
```

At the end we need to observe the train accuracy and test accuracy with cross-validation score. There is a constrain that the difference of train and test accuracy should not be more than 5%and the difference of test accuracy and cross-validation should not be more than 5% the best model is fixed by this scenario.

Activity 2.2 : Random-Forest Classifier

Random Forest algorithm is the classification and regressions algorithm initialized and training data is passed to the model and assigned to the variable as model4 with .fit() function. Test data is predicted with model4.predict() function and saved in a new variable. For evaluating the model accuracy is calculated. For the best obtaining of accuracy we use hyper parameter tuning to tune the model with the best hyper parameters using the Grid Search CV by choosing the best params we can able to get the best accuracy this method is known as Hyper parameter Tuning.

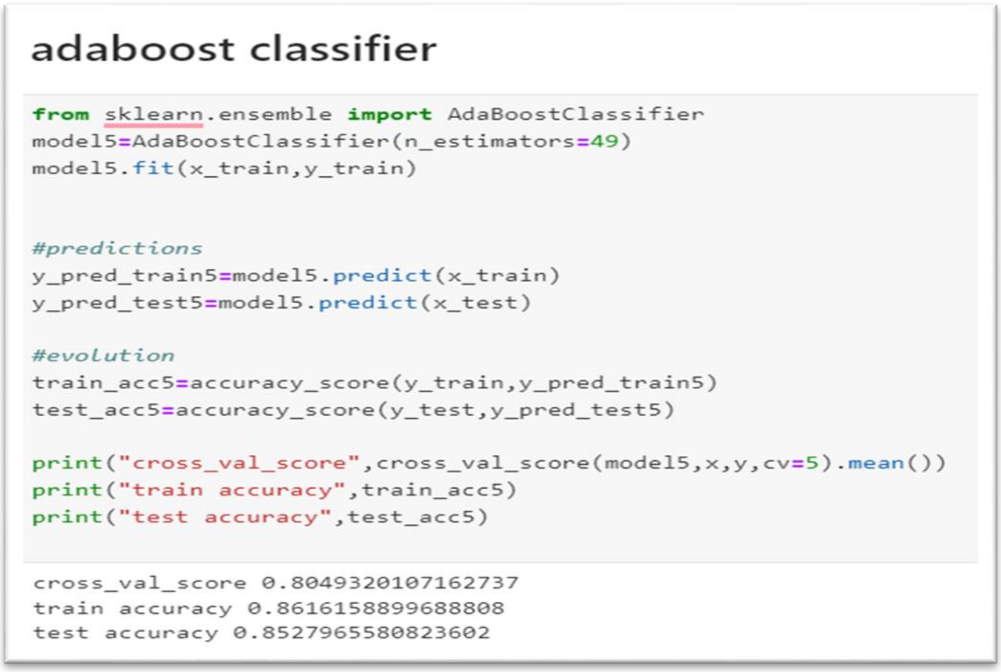


```
from sklearn.ensemble import RandomForestClassifier
model4=RandomForestClassifier(criterion="gini",max_depth=7)
model4.fit(x_train,y_train)

# predictions
y_pred_train4=model4.predict(x_train)
y_pred_test4=model4.predict(x_test)

#evolutions
train_acc4=accuracy_score(y_train,y_pred_train4)
test_acc4=accuracy_score(y_test,y_pred_test4)

from sklearn.model_selection import cross_val_score
print("cross_val_score", cross_val_score(model4,x,y,cv=5).mean()*100)
print("train_accuracy",train_acc4*100)
print("test_accuracy",test_acc4*100)
```

```
cross_val_score 80.86367380529056
train_accuracy 85.81848894348894
test accuracy 85.56732688469215
```

Activity 2.3 : Ada Boost Classifier

Ada Boost Classifier algorithm is initialized and training data is passed to the model and assigned to the variable as model5 with .fit() function. Test data is predicted with model5.predict() function and saved in a new variable. For evaluating the model accuracy is calculated.



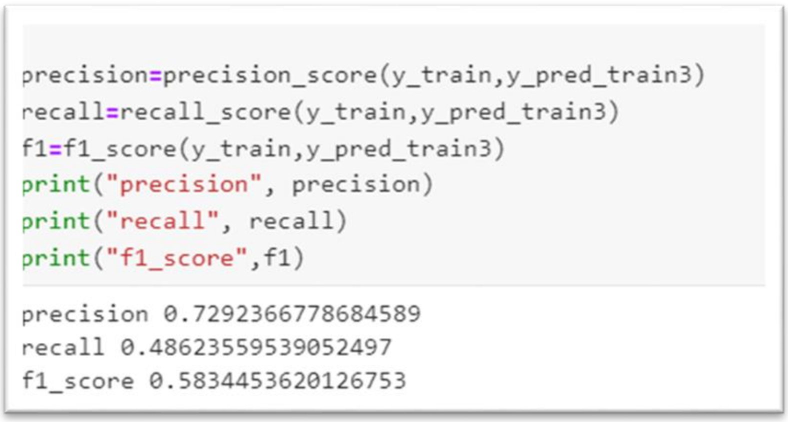
adaboost classifier

```
from sklearn.ensemble import AdaBoostClassifier
model5=AdaBoostClassifier(n_estimators=49)
model5.fit(x_train,y_train)


#predictions
y_pred_train5=model5.predict(x_train)
y_pred_test5=model5.predict(x_test)

#evolution
train_acc5=accuracy_score(y_train,y_pred_train5)
test_acc5=accuracy_score(y_test,y_pred_test5)

print("cross_val_score",cross_val_score(model5,x,y,cv=5).mean())
print("train accuracy",train_acc5)
print("test accuracy",test_acc5)
```

```
cross_val_score 0.8049320107162737
train accuracy 0.8616158899688808
test accuracy 0.8527965580823602
```

Milestone 4: Performance Testing

Under performance Testing we need to test the model's accuracy with different testing Metrics like Precision, Recall and F1_score. Below are the performance Metrics of final fixed model



```
precision=precision_score(y_train,y_pred_train3)
recall=recall_score(y_train,y_pred_train3)
f1=f1_score(y_train,y_pred_train3)
print("precision", precision)
print("recall", recall)
print("f1_score",f1)
```

```
precision 0.7292366778684589
recall 0.48623559539052497
f1_score 0.5834453620126753
```

Milestone 5: Model Deployment

Activity 1: Save and load 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.

```
import joblib
joblib.dump(model3,open("dtree.joblib","wb"))
```

-

Activity 2 : Test the model

Let's test the model first in python notebook itself. As we have 7 features in this model, let's check the output by giving all the inputs.

```
print(model3.predict([[40,4,11,2,6,0,4,1,0,0,40,39]]))
```

```
[0]
```

```
print(model3.predict([[74,6,10,4,9,2,4,0,0,3683,20,39]]))
```

```
[1]
```

```
print(model3.predict([[90,8,11,6,14,1,4,0,0,4356,40,39]]))
```

```
[1]
```

The predicted value and actual value are results same.

Activity 3: 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

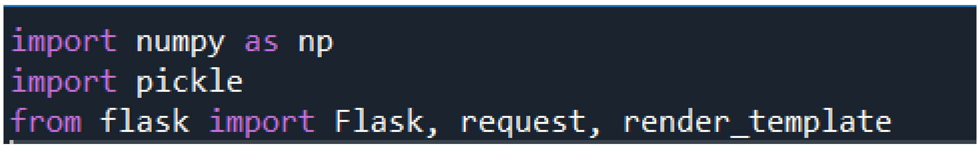
Activity 3.1: Building Html Pages:

For this project create two HTML files namely
- index.html
- predict.html
- result.html

and save them in the templates folder.

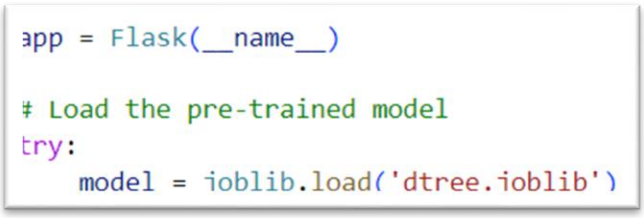
Activity 3.2: Build Python code:

Import the libraries



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

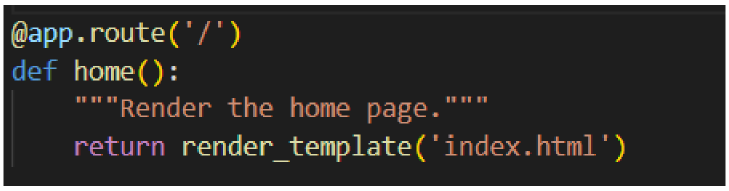
Load the saved model. Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (__name__) as argument.



```
app = Flask(__name__)

# Load the pre-trained model
try:
    model = joblib.load('dtree.joblib')
```

We render index.html for the displaying the web application , similarly we render the predict.html for the user input values of the forms to predict the income. Simultaneously we render result .html to display the result of the prediction value.

Render Index.html:

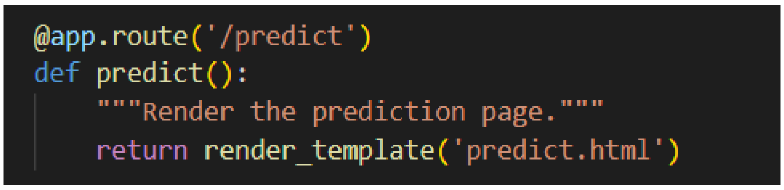


```python
@app.route('/')
def home():
    """Render the home page."""
    return render_template('index.html')
```

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.

Render Predict.html:

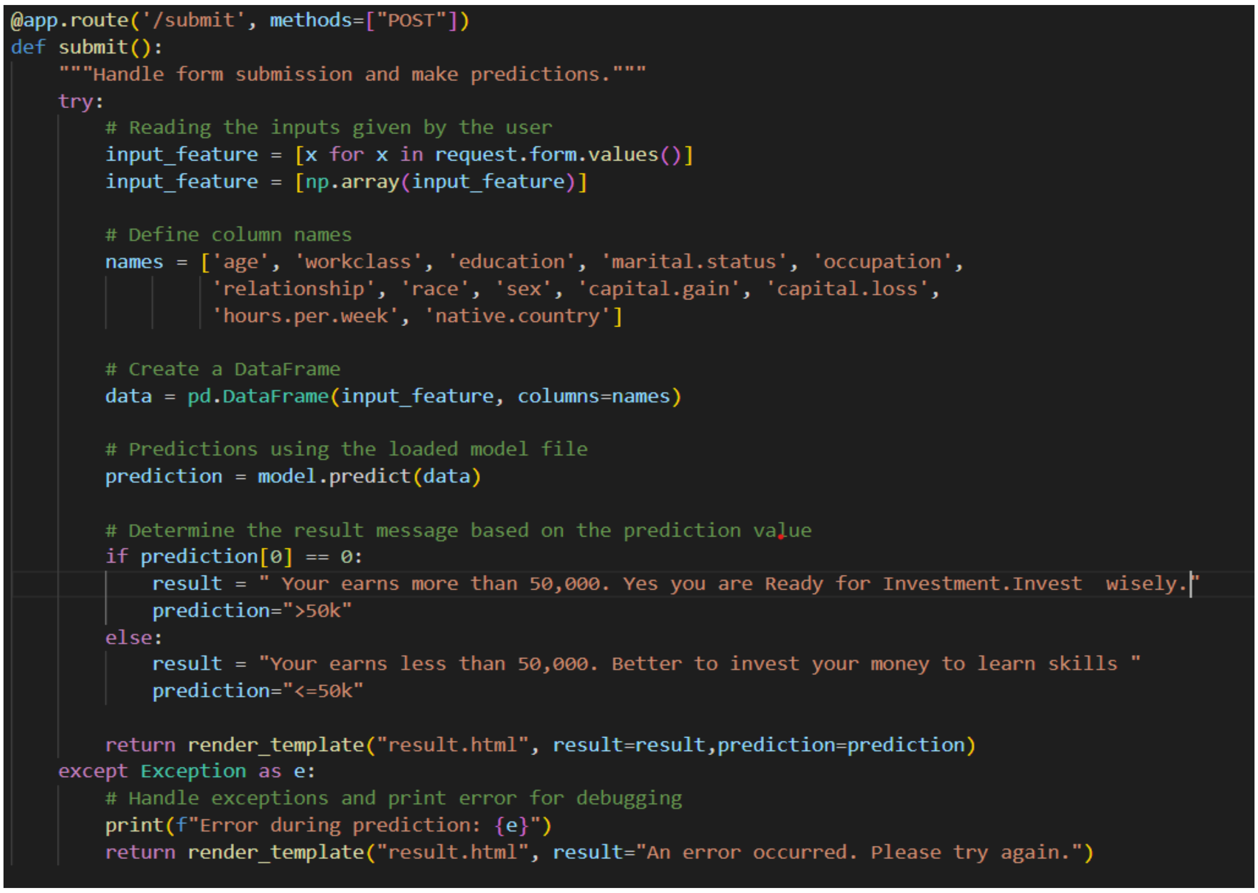


```python
@app.route('/predict')
def predict():
    """Render the prediction page."""
    return render_template('predict.html')
```

In the predict.html where we provide the user inputs in the form for the prediction of income

Whenever you enter the values from the html page the values can be retrieved using POST and GET Methods.

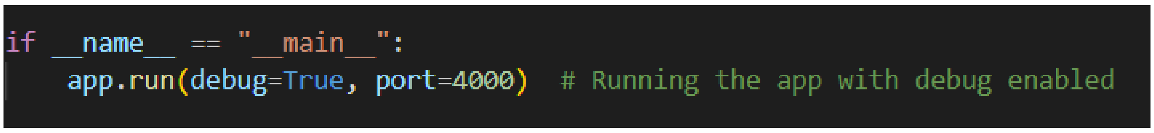
Retrieving the value from UI:



```python
@app.route('/submit', methods=["POST"])
def submit():
    """Handle form submission and make predictions."""
    try:
        # Reading the inputs given by the user
        input_feature = [x for x in request.form.values()]
        input_feature = [np.array(input_feature)]

        # Define column names
        names = ['age', 'workclass', 'education', 'marital.status', 'occupation',
                 'relationship', 'race', 'sex', 'capital.gain', 'capital.loss',
                 'hours.per.week', 'native.country']

        # Create a DataFrame
        data = pd.DataFrame(input_feature, columns=names)

        # Predictions using the loaded model file
        prediction = model.predict(data)

        # Determine the result message based on the prediction value
        if prediction[0] == 0:
            result = " Your earns more than 50,000. Yes you are Ready for Investment.Invest  wisely."
            prediction=">50k"
        else:
            result = "Your earns less than 50,000. Better to invest your money to learn skills "
            prediction="<=50k"

        return render_template("result.html", result=result,prediction=prediction)
    except Exception as e:
        # Handle exceptions and print error for debugging
        print(f"Error during prediction: {e}")
        return render_template("result.html", result="An error occurred. Please try again.")
```

Here we are routing our app to conditional statement. This will retrieve 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.

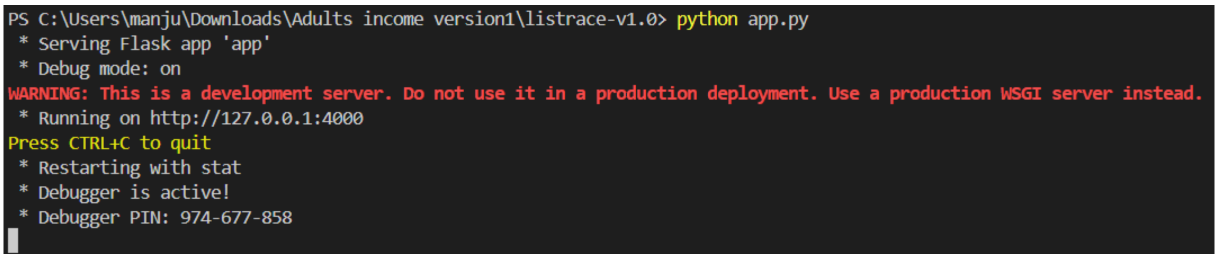
Main Function:



```
if __name__ == "__main__":
    app.run(debug=True, port=4000)  # Running the app with debug enabled
```

Activity 3.3: Run the web application

- Open vs code application in the search menu.
- Navigate to the folder where your flask folder of your files exist.
- Click on the view button in the vs code nav bar and click on the terminal option in the dropdown menu.
- Now type “app.py” command
- You will have a link displayed in the terminal as “http://127.0.0.1:4000 “.
- Double click on the link then you will be navigated to the web application.
- Click on the predict button in the nav bar , enter the inputs, click on the predict button, and see the result/prediction in the result.html..

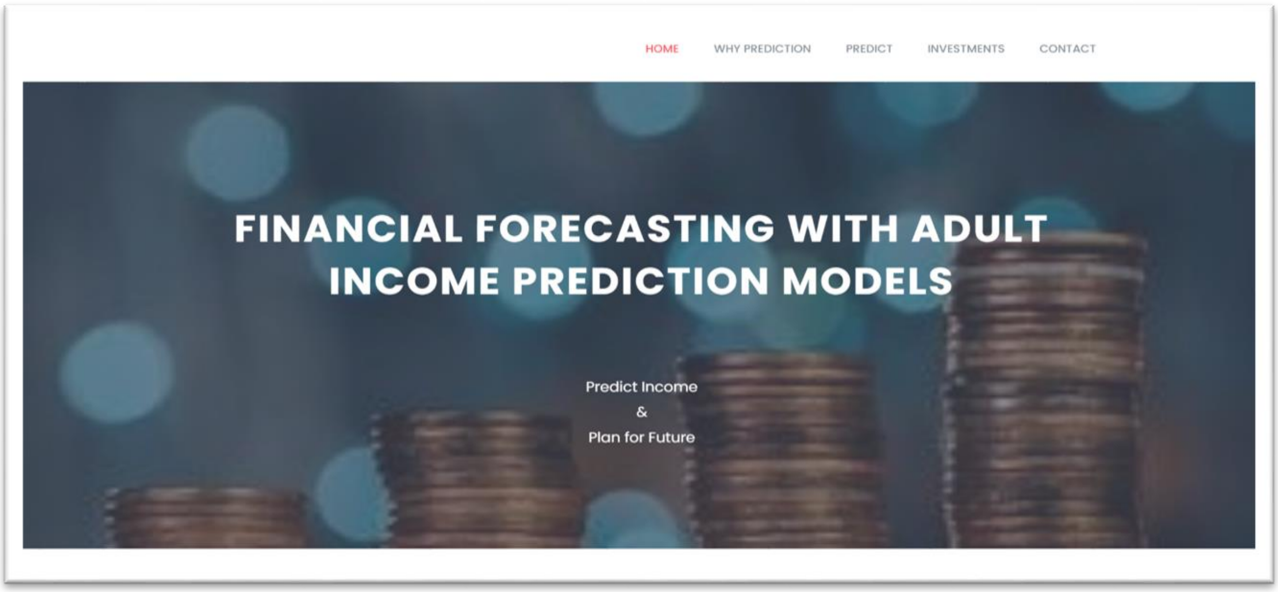


```
PS C:\Users\manju\Downloads\Adults income version1\listrace-v1.0> python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:4000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 974-677-858
```

Now, Go the web browser and write the localhost URL (http://127.0.0.1:4000) to get the below results

Results:

a. Index page (Index.html)

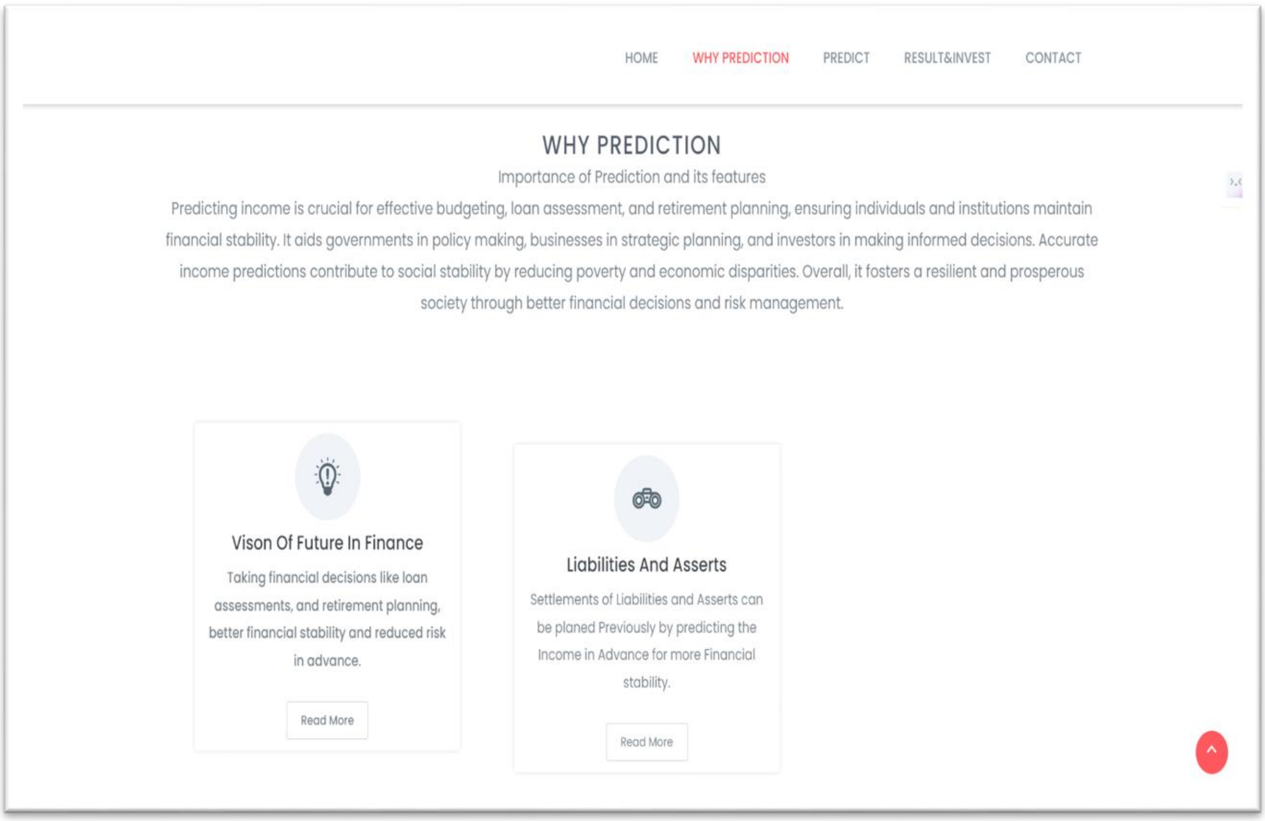


FINANCIAL FORECASTING WITH ADULT INCOME PREDICTION MODELS

Predict Income

&

Plan for Future



WHY PREDICTION

Importance of Prediction and its features

Predicting income is crucial for effective budgeting, loan assessment, and retirement planning, ensuring individuals and institutions maintain financial stability. It aids governments in policy making, businesses in strategic planning, and investors in making informed decisions. Accurate income predictions contribute to social stability by reducing poverty and economic disparities. Overall, it fosters a resilient and prosperous society through better financial decisions and risk management.

Vison Of Future In Finance

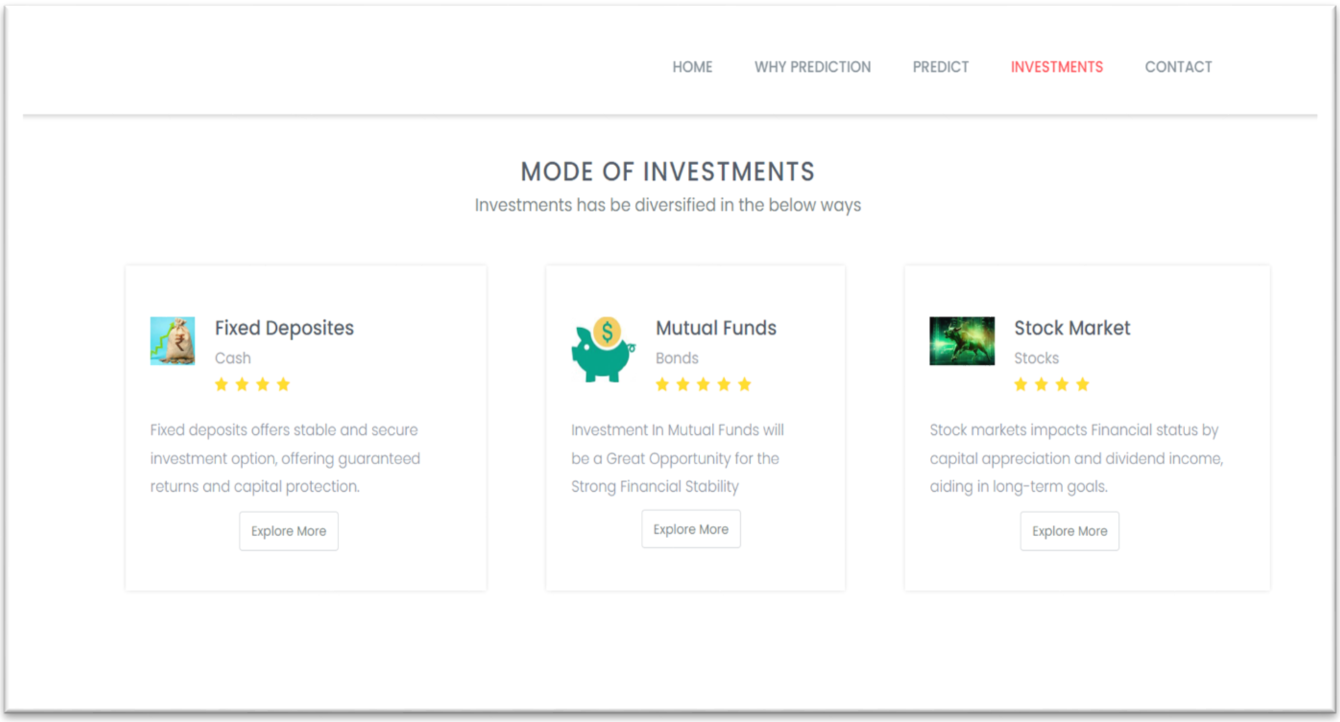
Taking financial decisions like loan assessments, and retirement planning, better financial stability and reduced risk in advance.

Read More

Liabilities And Asserts

Settlements of Liabilities and Asserts can be planed Previously by predicting the Income in Advance for more Financial stability.

Read More


HOME WHY PREDICTION PREDICT INVESTMENTS CONTACT

MODE OF INVESTMENTS

Investments has be diversified in the below ways

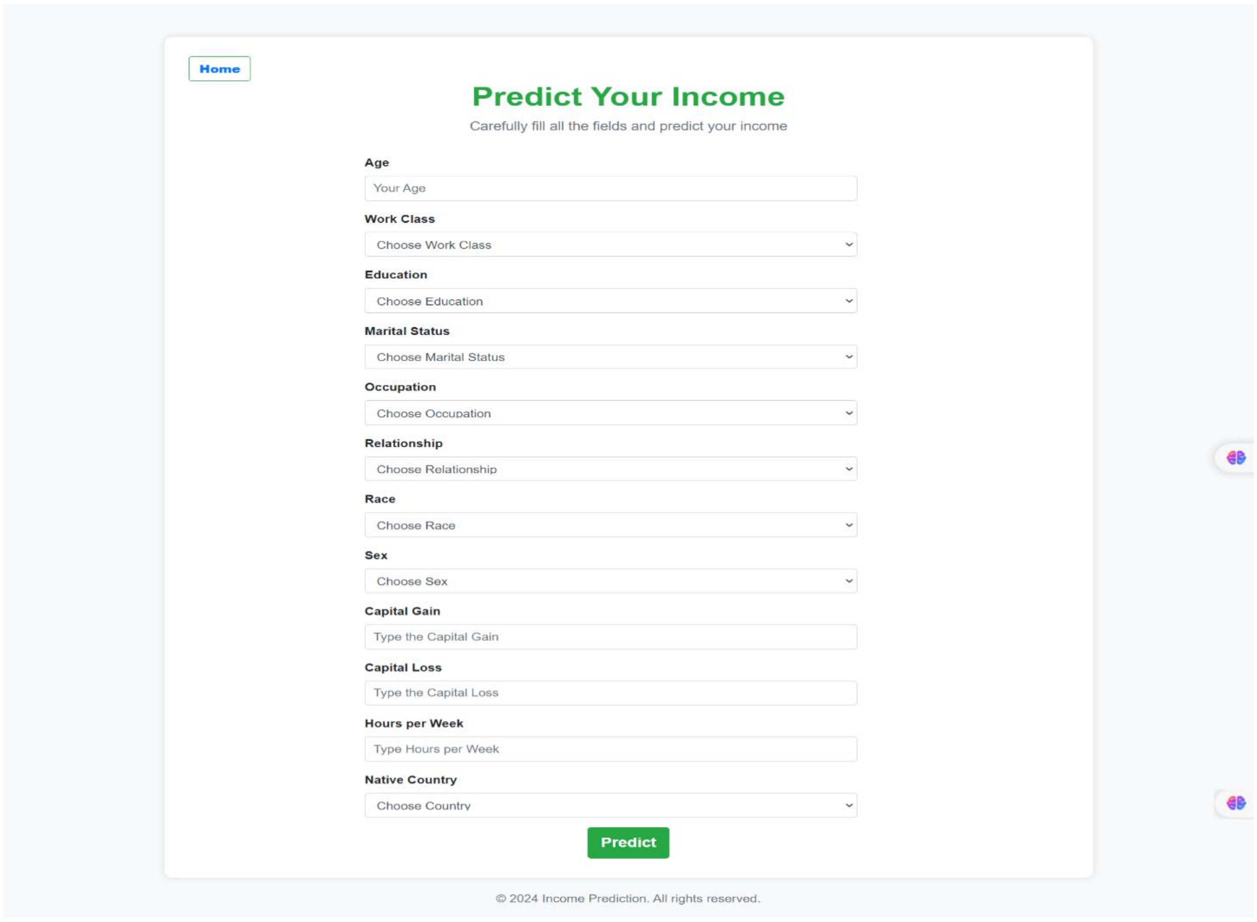
Fixed Deposites
Cash
★★★★
Fixed deposits offers stable and secure investment option, offering guaranteed returns and capital protection.
Explore More

Mutual Funds
Bonds
★★★★★
Investment In Mutual Funds will be a Great Opportunity for the Strong Financial Stability
Explore More

Stock Market
Stocks
★★★★
Stock markets impacts Financial status by capital appreciation and dividend income, aiding in long-term goals.
Explore More

If you click on the predict on the nav bar we can able to navigate to predict.html below figure represents the UI of predict.html page

b) Prediction page (Predict.html)


Home

Predict Your Income

Carefully fill all the fields and predict your income

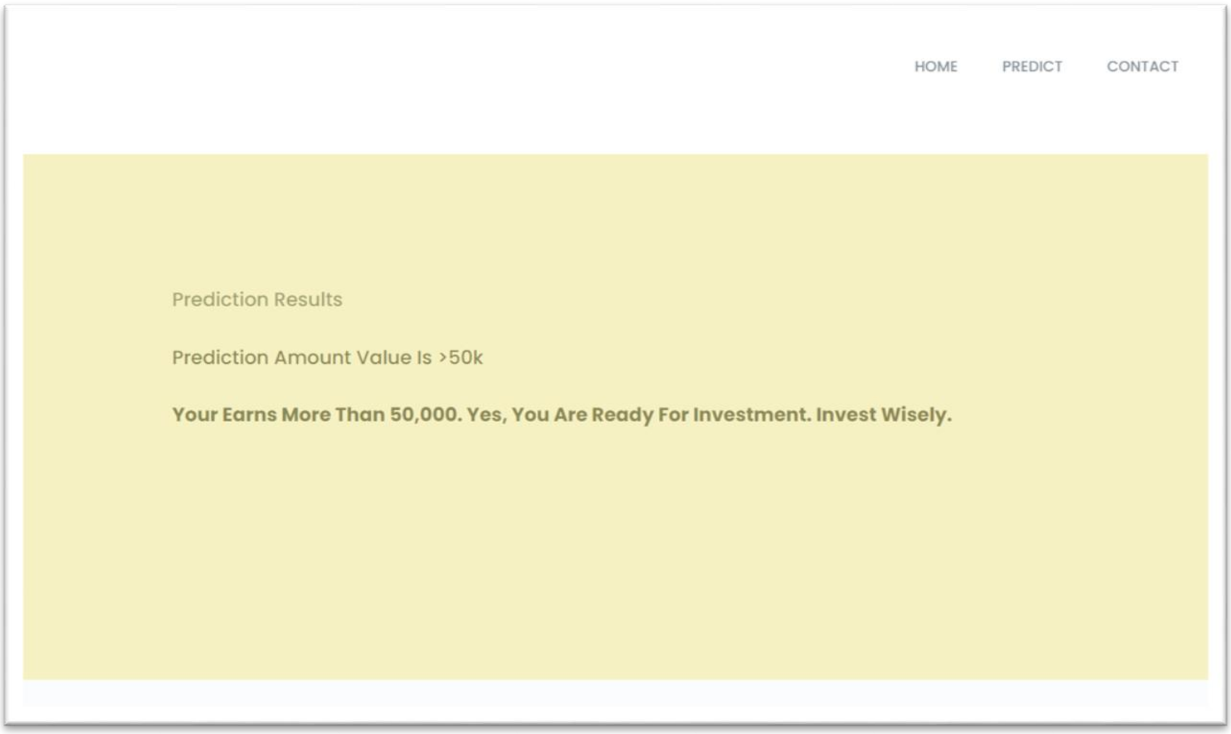
Age	Your Age
Work Class	Choose Work Class
Education	Choose Education
Marital Status	Choose Marital Status
Occupation	Choose Occupation
Relationship	Choose Relationship
Race	Choose Race
Sex	Choose Sex
Capital Gain	Type the Capital Gain
Capital Loss	Type the Capital Loss
Hours per Week	Type Hours per Week
Native Country	Choose Country

Predict

© 2024 Income Prediction. All rights reserved.

By providing the inputs by the user and click on the predict button you will be navigating to the result.html it displays the result. Based on the result you can navigate to the home and click on the investments in the nav bar of index.html then you can have an info about different mode of investments.

Result.html:



Prediction Results

Prediction Amount Value Is >50k

Your Earns More Than 50,000. Yes, You Are Ready For Investment. Invest Wisely.