# DATA SCIENCE

Name: K Vinoth Kumar

Roll No: CH.EN.U4ARE22011

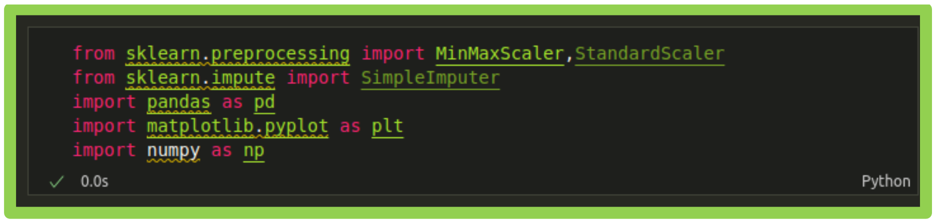
## AIM :

The aim of this code is to perform comprehensive data preprocessing and exploratory data analysis on a diabetes dataset.

The Code used in this assignment can be found here

## CODE :

1. Import Libraries for performing numerical operations, data manipulation & analysis, for creating plots.

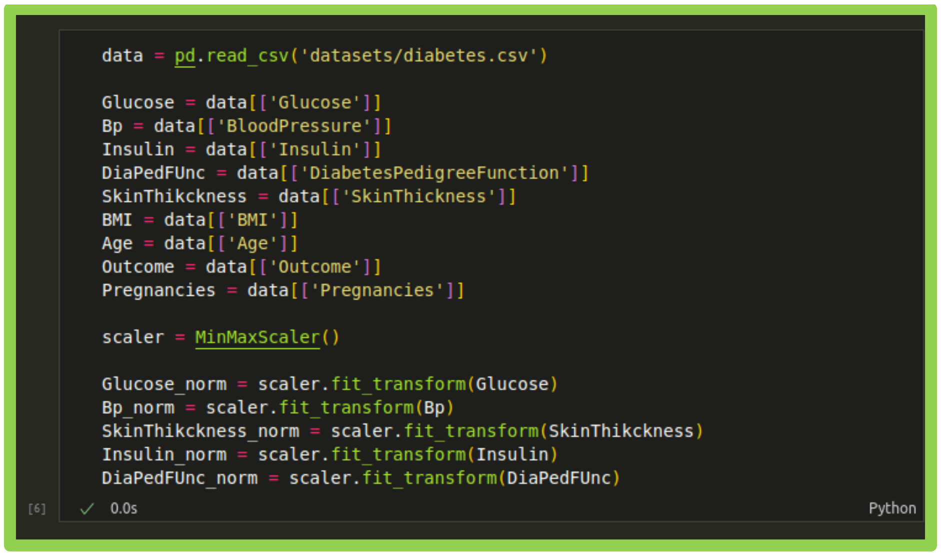


```python
from sklearn.preprocessing import MinMaxScaler,StandardScaler
from sklearn.impute import SimpleImputer
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

```
✓ 0.0s                                                    Python
```

*Data* – Dataset loaded from ‘diabetes.csv’

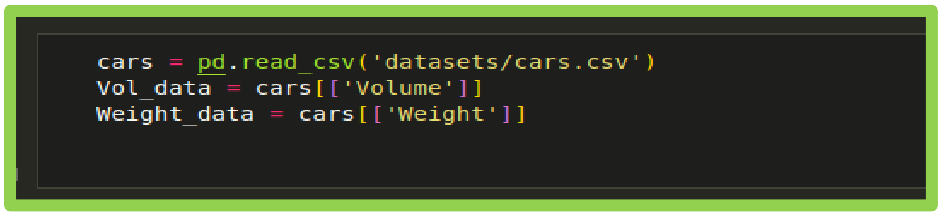
Columns *‘Glucose’*, *‘skin thickness’*, *'DiabetesPedigreeFunction'*, *'BloodPressure'*, and '*Insulin'* are normalized using **MinMaxScaler.**



```python
data = pd.read_csv('datasets/diabetes.csv')

Glucose = data[['Glucose']]
Bp = data[['BloodPressure']]
Insulin = data[['Insulin']]
DiaPedFUnc = data[['DiabetesPedigreeFunction']]
SkinThikckness = data[['SkinThickness']]
BMI = data[['BMI']]
Age = data[['Age']]
Outcome = data[['Outcome']]
Pregnancies = data[['Pregnancies']]

scaler = MinMaxScaler()

Glucose_norm = scaler.fit_transform(Glucose)
Bp_norm = scaler.fit_transform(Bp)
SkinThikckness_norm = scaler.fit_transform(SkinThikckness)
Insulin_norm = scaler.fit_transform(Insulin)
DiaPedFUnc_norm = scaler.fit_transform(DiaPedFUnc)
```

```
[6]  ✓ 0.0s                                               Python
```

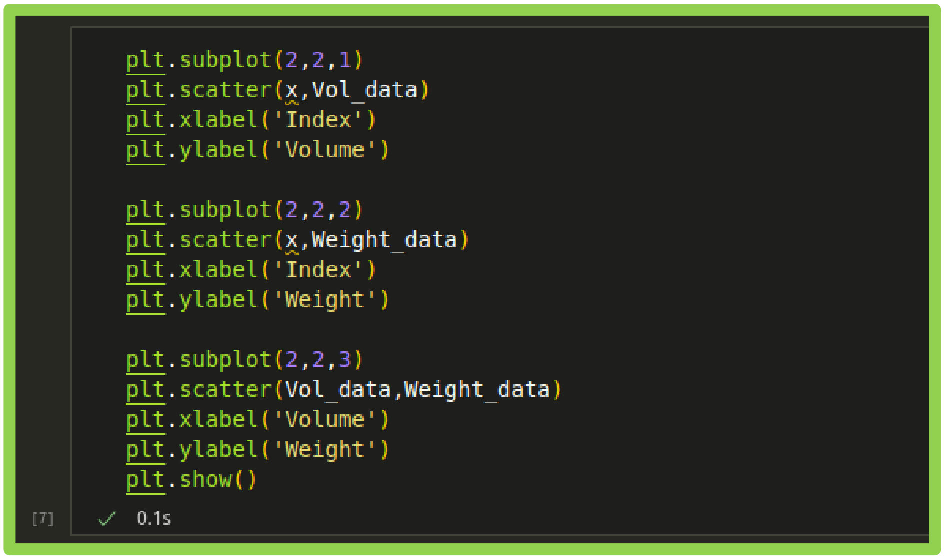
Cars – Dataset loaded from 'cars.csv'



```
cars = pd.read_csv('datasets/cars.csv')
Vol_data = cars[['Volume']]
Weight_data = cars[['Weight']]
```

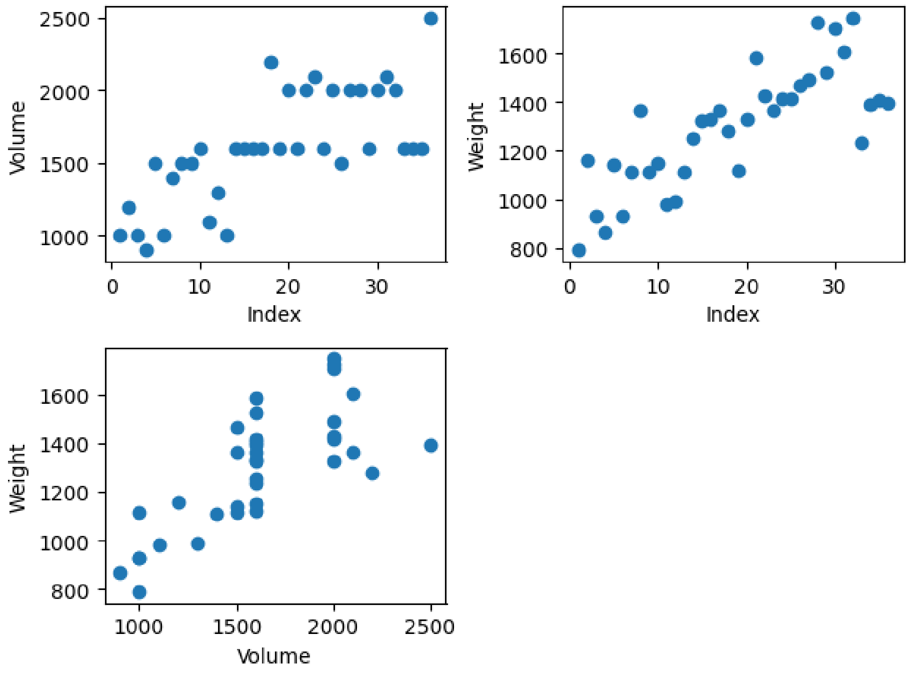
The first two subplots has Volume and Weight respectively scatter plotted with respect to index.

The third subplot has a scatter plot with Volume data on the x-axis and Weight data on the y-axis.



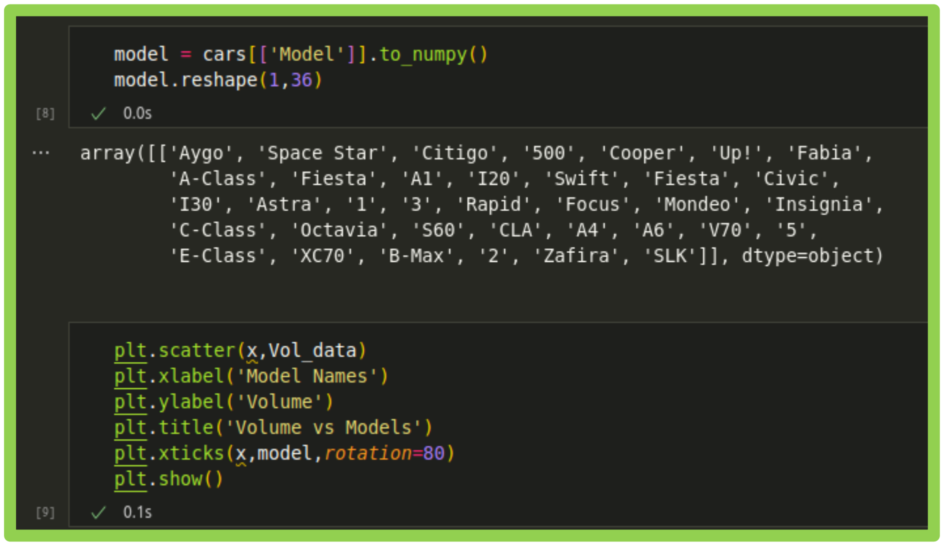
```
plt.subplot(2,2,1)
plt.scatter(x,Vol_data)
plt.xlabel('Index')
plt.ylabel('Volume')

plt.subplot(2,2,2)
plt.scatter(x,Weight_data)
plt.xlabel('Index')
plt.ylabel('Weight')

plt.subplot(2,2,3)
plt.scatter(Vol_data,Weight_data)
plt.xlabel('Volume')
plt.ylabel('Weight')
plt.show()
```

```
[7]   ✓ 0.1s
```



The model names are extracted to a variable called *model*. It is then converted and reshaped to numpy array to be plotted in y axis.

First, Scatter plot is plotted between index and Volumes. Then the label is set on the x-axis with a rotation of 80 degrees for better readability.



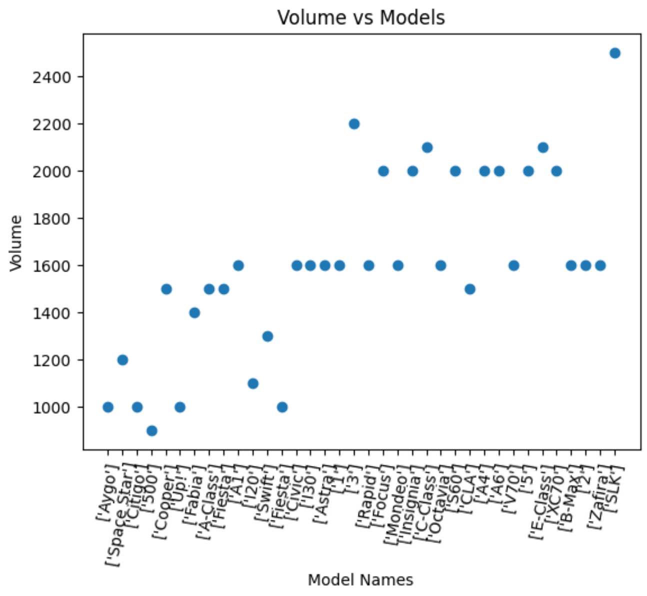
```
model = cars[['Model']].to_numpy()
model.reshape(1,36)
```

[8] ✓ 0.0s

```
array([['Aygo', 'Space Star', 'Citigo', '500', 'Cooper', 'Up!', 'Fabia',
        'A-Class', 'Fiesta', 'A1', 'I20', 'Swift', 'Fiesta', 'Civic',
        'I30', 'Astra', '1', '3', 'Rapid', 'Focus', 'Mondeo', 'Insignia',
        'C-Class', 'Octavia', 'S60', 'CLA', 'A4', 'A6', 'V70', '5',
        'E-Class', 'XC70', 'B-Max', '2', 'Zafira', 'SLK']], dtype=object)
```

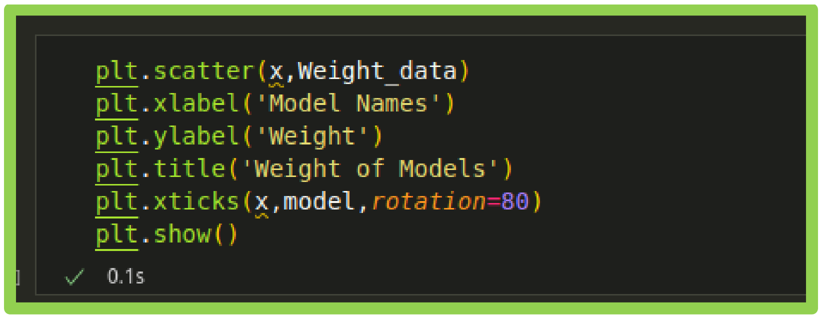
```
plt.scatter(x,Vol_data)
plt.xlabel('Model Names')
plt.ylabel('Volume')
plt.title('Volume vs Models')
plt.xticks(x,model,rotation=80)
plt.show()
```

[9] ✓ 0.1s



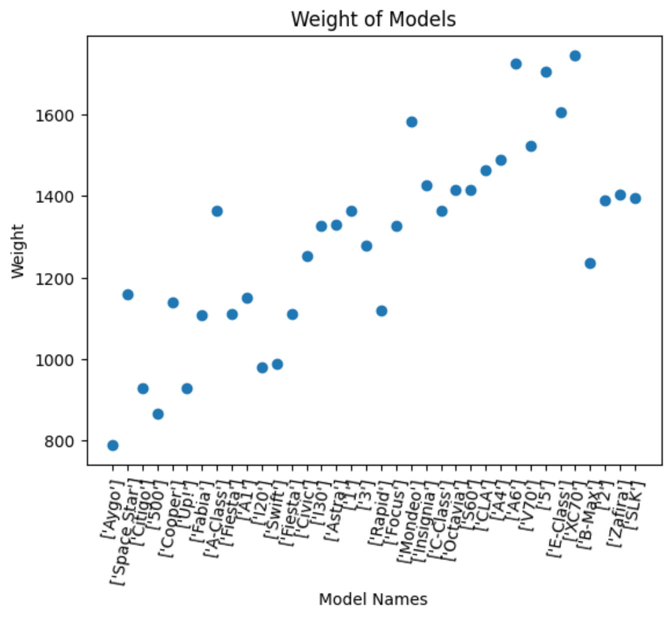
Here also, a scatter plot has been plotted between Model Name and Weight.

After that the label is set on the x-axis with a rotation of 80 degrees for better readability.



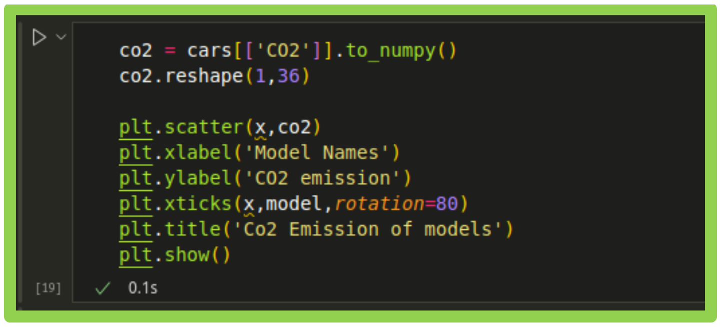
```
plt.scatter(x,Weight_data)
plt.xlabel('Model Names')
plt.ylabel('Weight')
plt.title('Weight of Models')
plt.xticks(x,model,rotation=80)
plt.show()
```

```
✓ 0.1s
```



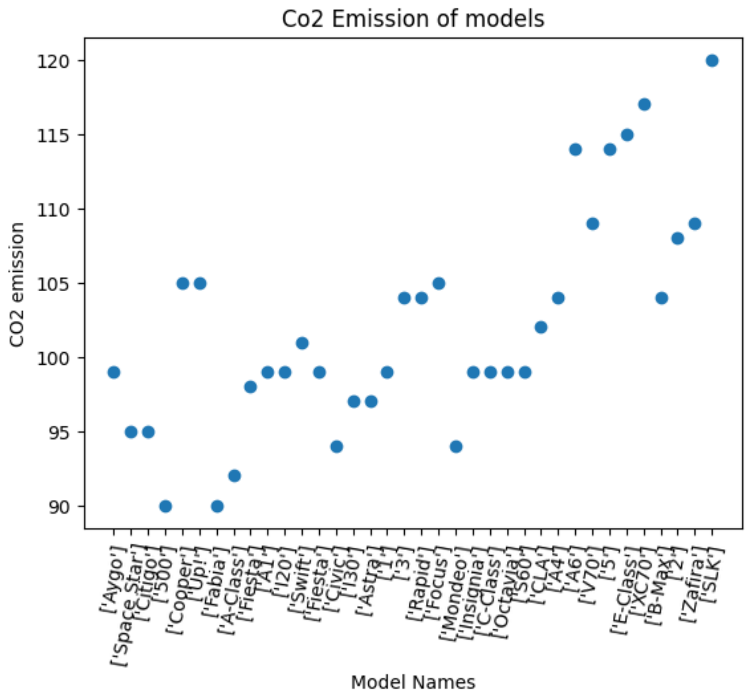
The Co2 emission data are extracted to a variable called *co2*. It is then converted and reshaped to numpy array to be plotted in y axis.

Then a scatter plot has been plotted between Model names and CO2 Emissions which says the level of emission for different models



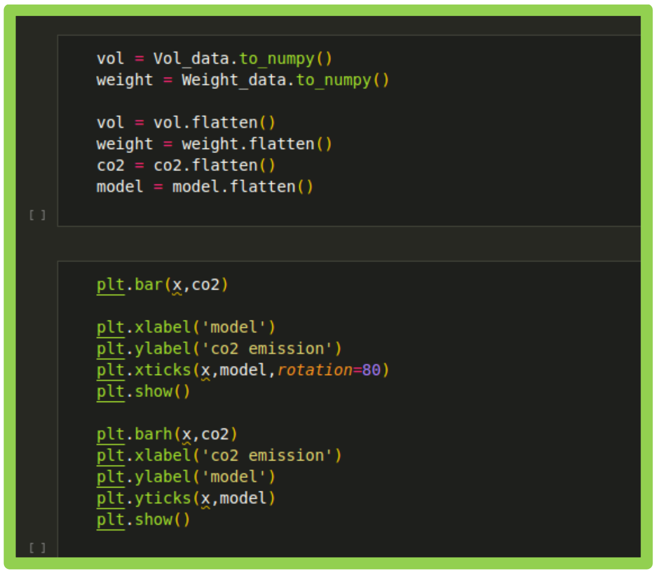
```
co2 = cars[['CO2']].to_numpy()
co2.reshape(1,36)

plt.scatter(x,co2)
plt.xlabel('Model Names')
plt.ylabel('CO2 emission')
plt.xticks(x,model,rotation=80)
plt.title('Co2 Emission of models')
plt.show()
```

```
[19]  ✓ 0.1s
```



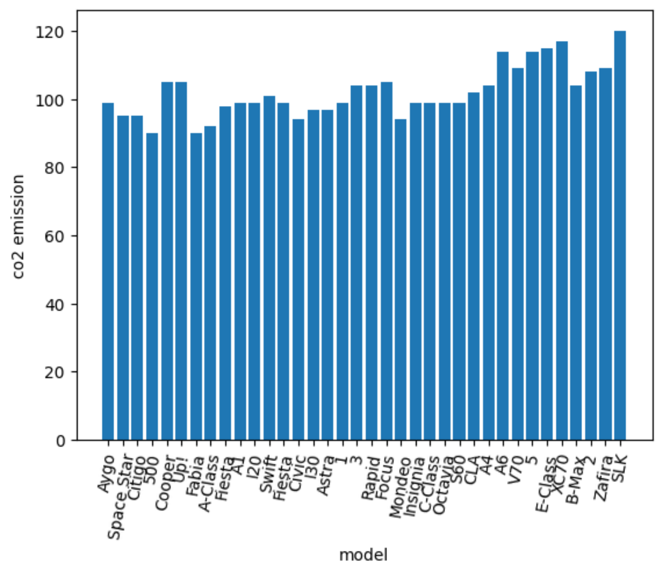
Volume, Weights and Co2 data are extracted separately stored in variables. Then they are converted to 1D array by flatten().

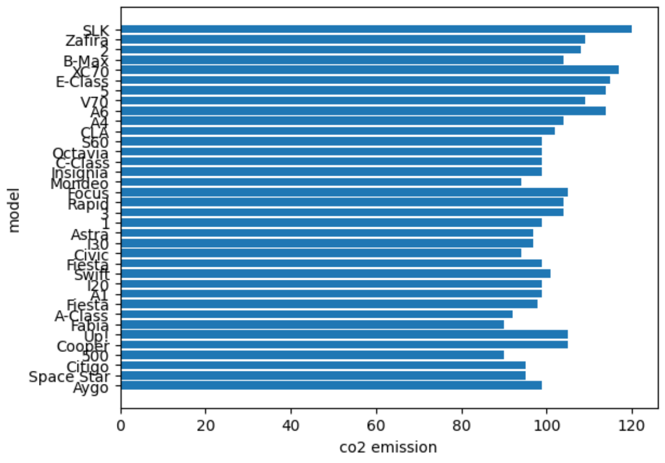
1D arrays are often required for performing computations and plottting.



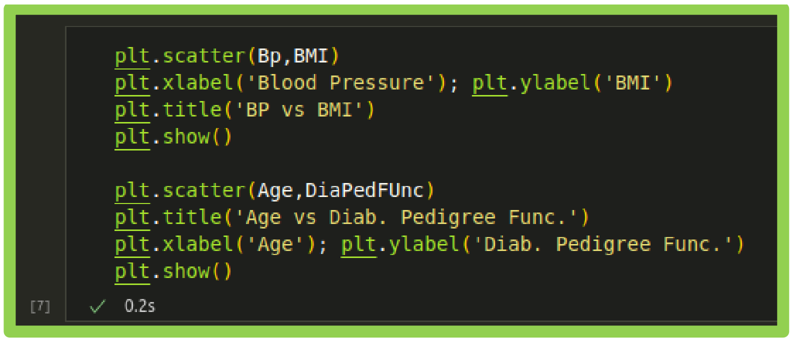
```
vol = Vol_data.to_numpy()
weight = Weight_data.to_numpy()

vol = vol.flatten()
weight = weight.flatten()
co2 = co2.flatten()
model = model.flatten()
```

```
plt.bar(x,co2)

plt.xlabel('model')
plt.ylabel('co2 emission')
plt.xticks(x,model,rotation=80)
plt.show()

plt.barh(x,co2)
plt.xlabel('co2 emission')
plt.ylabel('model')
plt.yticks(x,model)
plt.show()
```



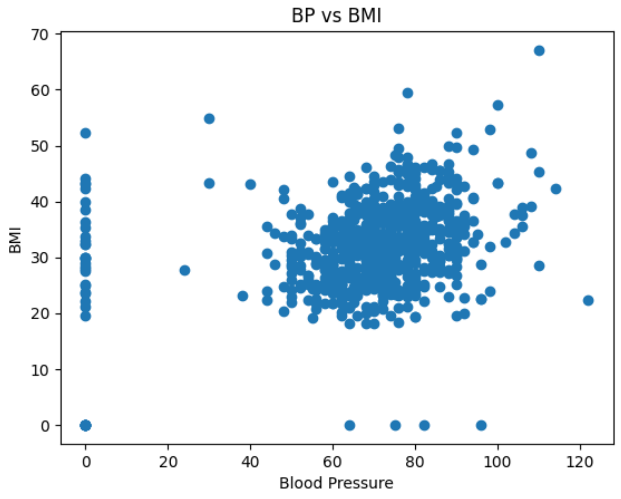


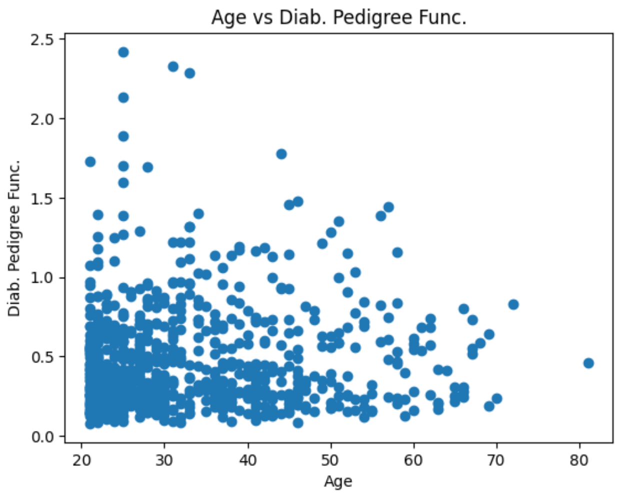
Scatter plotted between Blood Pressure – BMI & Age – Diabetes Pedigree Function to see the relationships between them.



```
plt.scatter(Bp,BMI)
plt.xlabel('Blood Pressure'); plt.ylabel('BMI')
plt.title('BP vs BMI')
plt.show()

plt.scatter(Age,DiaPedFUnc)
plt.title('Age vs Diab. Pedigree Func.')
plt.xlabel('Age'); plt.ylabel('Diab. Pedigree Func.')
plt.show()
```

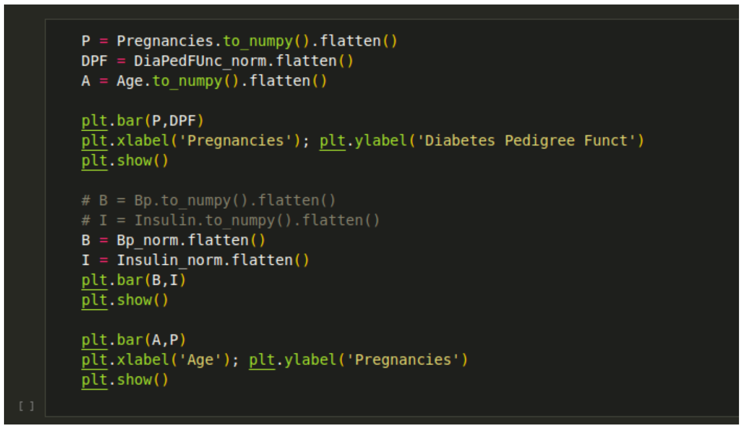
[7] ✓ 0.2s



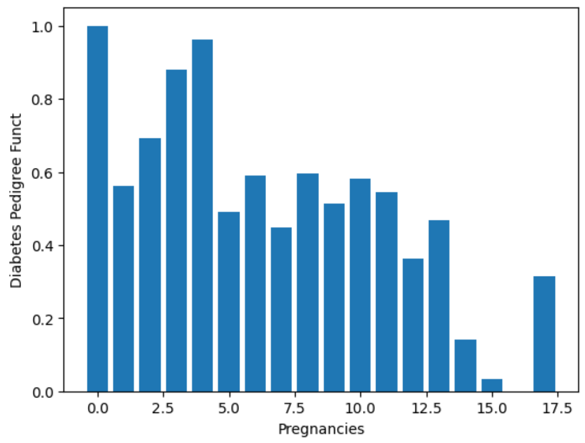


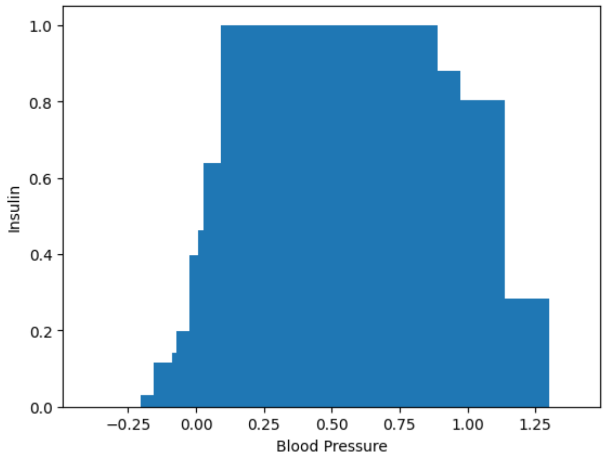
Here, Pregnancies Count of patients, Diabetes Pedigree Function and Age are converted to numpy arrays.

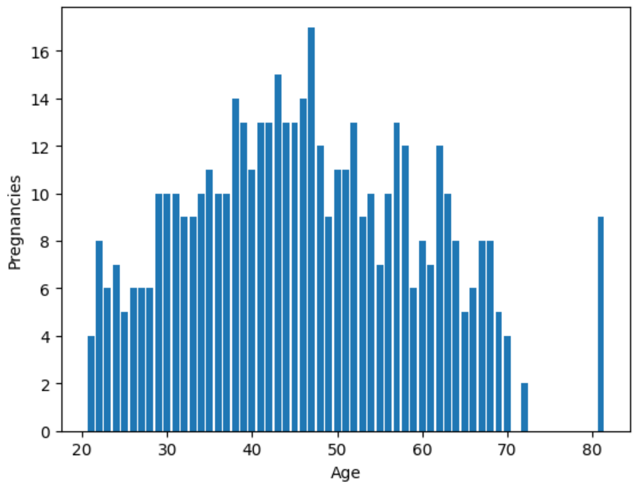
Checked the relationship between Pregnancies and Diabetes Pedigree Function, Blood pressure and Insulin using bar graphs.



```
P = Pregnancies.to_numpy().flatten()
DPF = DiaPedFUnc_norm.flatten()
A = Age.to_numpy().flatten()

plt.bar(P,DPF)
plt.xlabel('Pregnancies'); plt.ylabel('Diabetes Pedigree Funct')
plt.show()

# B = Bp.to_numpy().flatten()
# I = Insulin.to_numpy().flatten()
B = Bp_norm.flatten()
I = Insulin_norm.flatten()
plt.bar(B,I)
plt.show()

plt.bar(A,P)
plt.xlabel('Age'); plt.ylabel('Pregnancies')
plt.show()
```



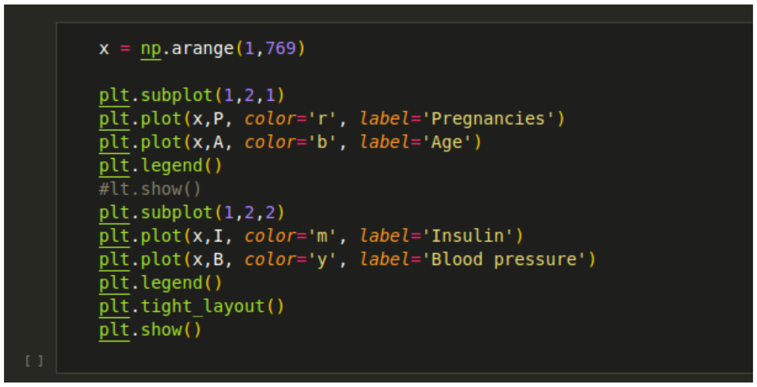




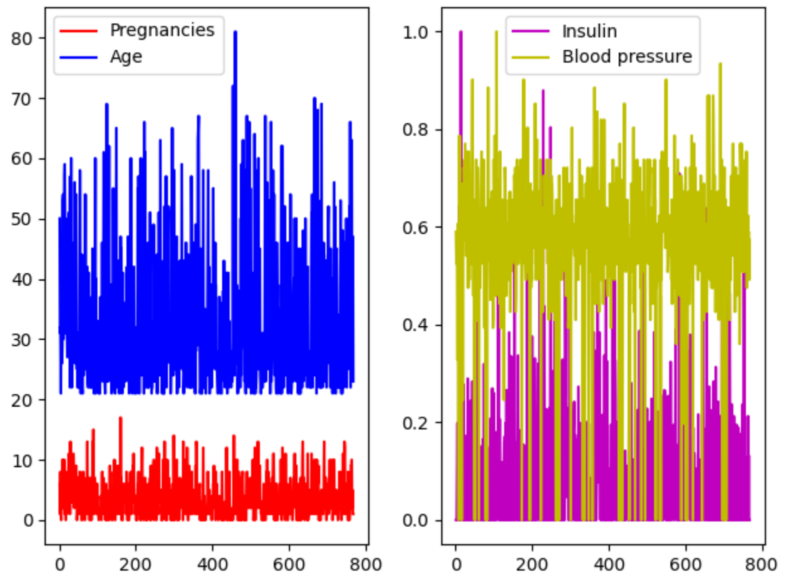
<u>#OVERLAY PLOTS</u>

Created two line plots within a single figure, comparing different sets of data.

This code snippet results in a figure with two subplots side by side, showing the trends of 'Pregnancies' and 'Age' in the first subplot, and 'Insulin' and 'Blood Pressure' in the second subplot.



```
x = np.arange(1,769)

plt.subplot(1,2,1)
plt.plot(x,P, color='r', label='Pregnancies')
plt.plot(x,A, color='b', label='Age')
plt.legend()
#lt.show()
plt.subplot(1,2,2)
plt.plot(x,I, color='m', label='Insulin')
plt.plot(x,B, color='y', label='Blood pressure')
plt.legend()
plt.tight_layout()
plt.show()
```



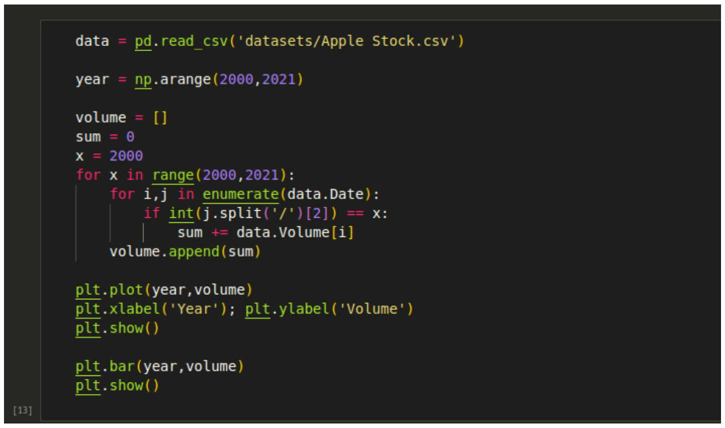
Apple Dataset is loaded and stored in variable ‘data’.

Initialize an empty list to store the annual trading volumes. Initialize a variable sum to accumulate the trading volume.

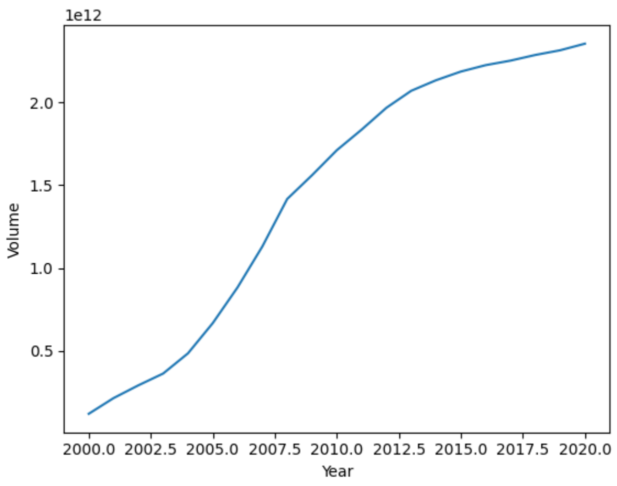
Iterate over each year from 2000 to 2020. For each year, iterate over the Date column in the dataset.

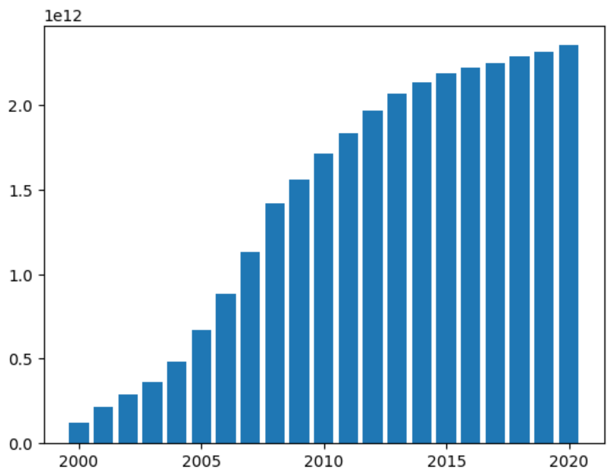
If the year part of the date matches the current year, add the corresponding Volume to sum. Append the accumulated volume for the year to the volume list.

Visualized the volume of models for every year by a line plot and bar graph.

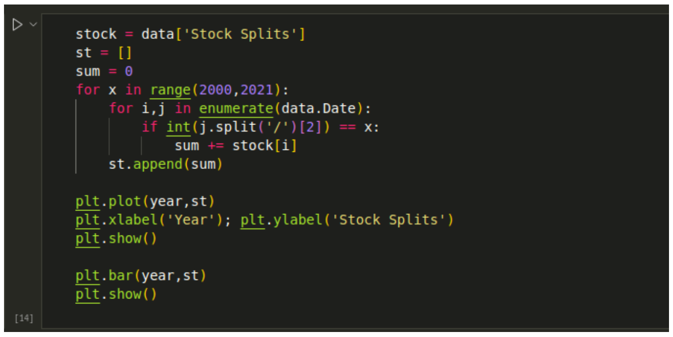


```
data = pd.read_csv('datasets/Apple Stock.csv')

year = np.arange(2000,2021)

volume = []
sum = 0
x = 2000
for x in range(2000,2021):
    for i,j in enumerate(data.Date):
        if int(j.split('/')[2]) == x:
            sum += data.Volume[i]
    volume.append(sum)

plt.plot(year,volume)
plt.xlabel('Year'); plt.ylabel('Volume')
plt.show()

plt.bar(year,volume)
plt.show()
```

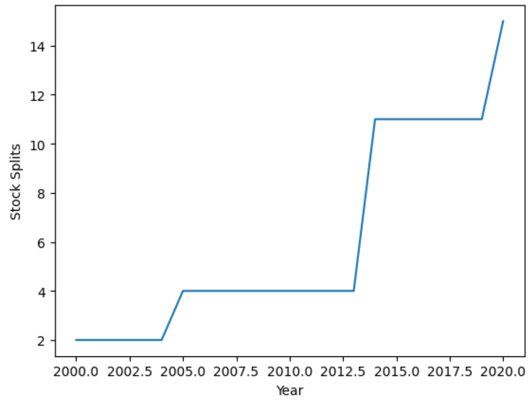


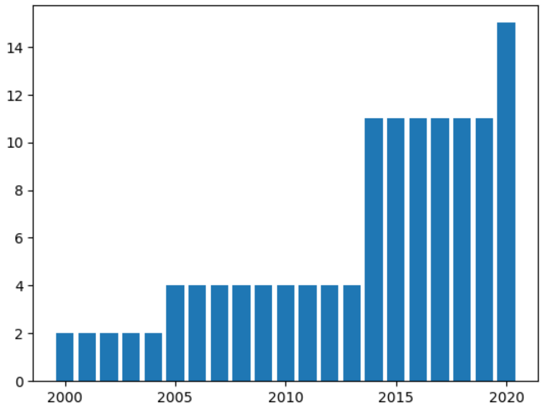


Visualized the volume of models for every year by a line plot and bar graph.



```
stock = data['Stock Splits']
st = []
sum = 0
for x in range(2000,2021):
    for i,j in enumerate(data.Date):
        if int(j.split('/')[2]) == x:
            sum += stock[i]
    st.append(sum)

plt.plot(year,st)
plt.xlabel('Year'); plt.ylabel('Stock Splits')
plt.show()

plt.bar(year,st)
plt.show()
```





**<u>RESULT</u>** :

The results showcase successful execution of various data preprocessing and plotting techniques, providing a comprehensive understanding of the dataset's structure, relationships, and distributions.

This thorough analysis is foundational for building effective data models and deriving meaningful insights from the data.

**<u>LEARNING OUTCOMES</u>** :

Gained proficiency in loading datasets into pandas DataFrames and extracting specific columns for focused analysis.

Acquired the ability to create various types of plots using matplotlib, including scatter plots, bar graphs, overlay plots.

Enhanced the ability to perform a comprehensive data analysis by combining multiple preprocessing and visualization techniques to gain deeper insights into the dataset.

Understood how to visualize relationships between different features, which is essential for exploratory data analysis and identifying patterns or trends in the data.