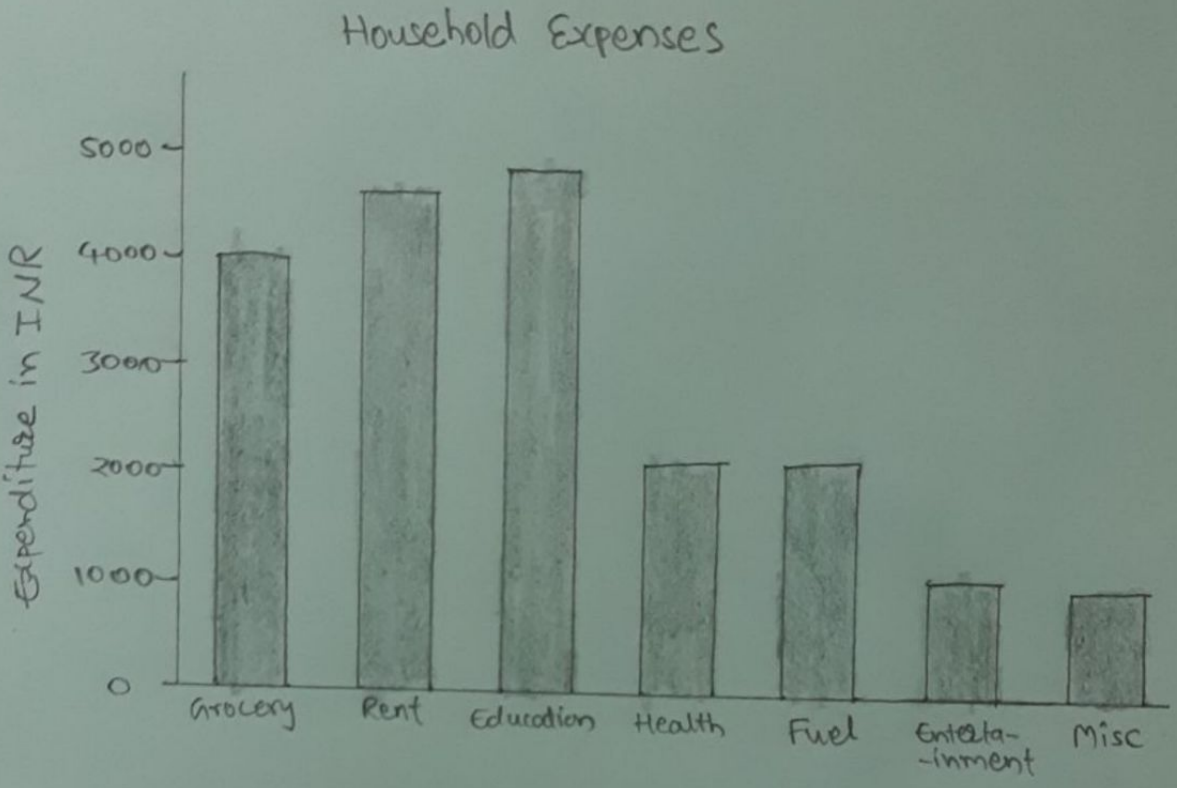
Output -

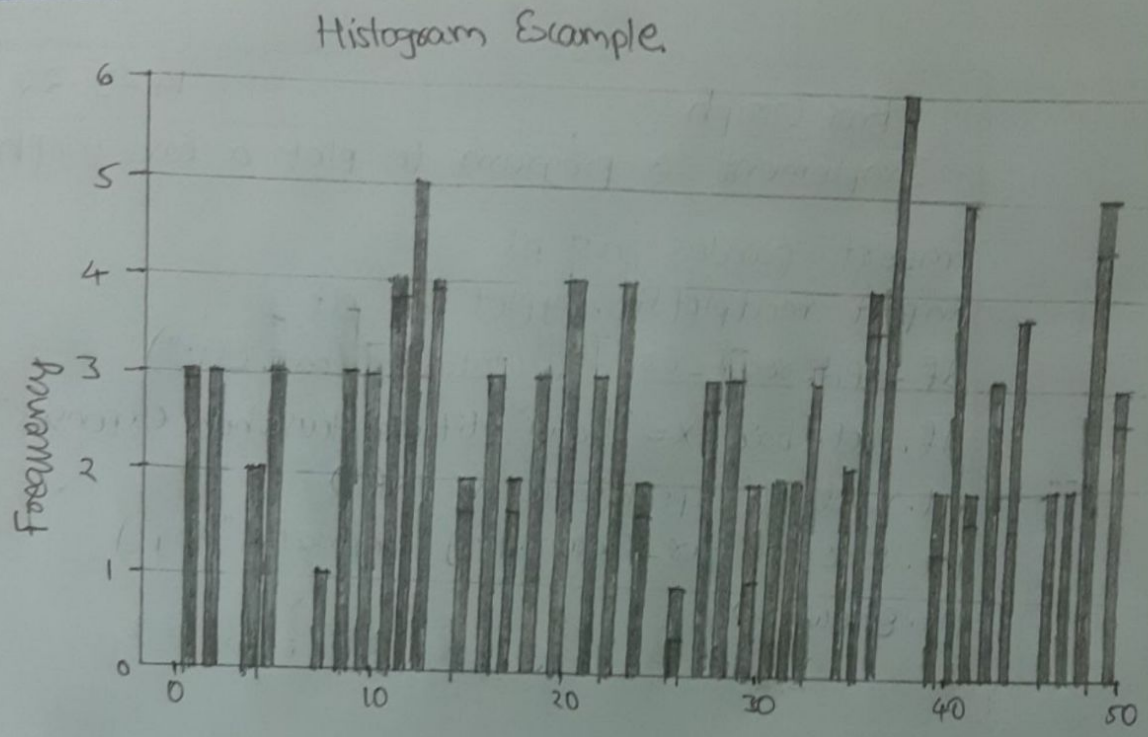

Household Expenses
Expenditure in INR
5000
4000
3000
2000
1000
0
Grocery
Rent
Education
Health
Fuel
Entertainment
Misc

TITLE: Bar Graph | DATE: 14-3-22

Aim: Implement a program to plot a bar graph

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv("../datasets/bar.csv")
df.plot.bar(x="Head", title="Household Expenses")
plt.ylabel("Expenditure in INR")
plt.savefig("bar_chart.png", format="png")
plt.show()
```

Output -

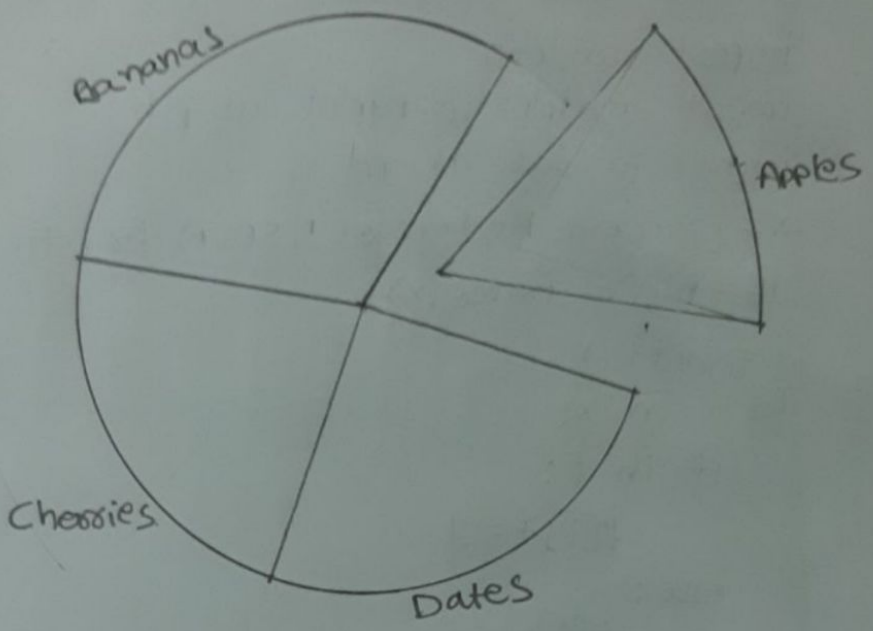


TITLE: Histogram | DATE: 14-3-22

Aim: Implement a program to plot a histogram

```
import random
import matplotlib.pyplot as plt
import pandas as pd
x=[random.randrange(1,50,1) for i in range(100)]
df=pd.DataFrame(x)
f=dict()
for i in x:
    if in f:
        f[i]+=1
    else:
        f[i]=1
plt.bar(f.keys(),f.values(),color="green")
plt.ylabel("Frequency")
plt.title("Histogram Example")
plt.savefig("Histogram.png",format="png")
plt.show()
```

Output -



TITLE: Pie Chart | DATE: 14-3-22

Aim: Implement a program to plot a Pie Chart

```
import matplotlib.pyplot as plt
import numpy as np
lst = np.random.random_sample(4)
y = np.array(lst)
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]
myexplode = [0.2, 0, 0, 0]
plt.pie(y, labels= mylabels, explode = myexplode,
                                    startangle = 0)

plt.savefig("pie.png", format ="png")
plt.show()
```

Output -

```
/home/varun/AI_ML/lab/manual
Index(['LOCATION', 'Country', 'INDICATOR', 'Indicator', 'MEASURE',
       'Measure', 'INEQUALITY', 'Inequality', 'Unit Code', 'Unit',
       'PowerCode Code', 'PowerCode', 'Reference Period Code',
       'Reference Period', 'Value', 'Flag Codes', 'Flags'],
      dtype='object').
(3292, 17)
```

	LOCATION	Country	INDICATOR		Flags
0	AUS	Australia	HO_BASE		Estimated Value
1	AUT	Austria	HO_BASE		NaN
2	BEL	Belgium	HO_BASE		NaN
3	CAN	Canada	HO_BASE		NaN
4	CZE	Czech Republic	HO_BASE		NaN

(5 rows x 17 columns)

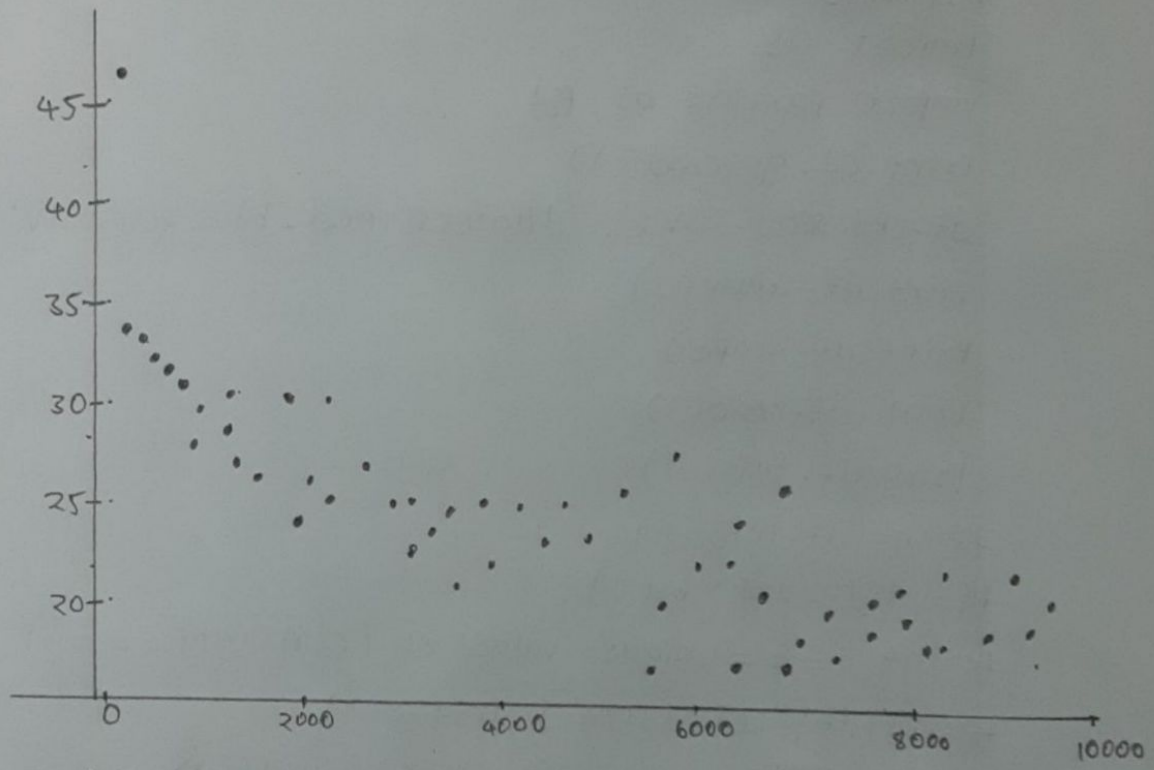
	LOCATION	Country	INDICATOR		Flags
3287	EST	Estonia	WL_TNOW		NaN
3288	ISR	Israel	WL_TNOW		Estimated Value
3289	RUS	Russia	WL_TNOW		Estimated Value
3290	SVN	Slovenia	WL_TNOW		NaN
3291	OECD	OECD-Total	WL_TNOW		NaN

TITLE: Data Cleaning - MetaData | DATE: 21-3-22

Aim: Implement a program for reading and cleaning Datasets

```
import os
import pandas as pd
print(os.getcwd())
df = pd.read_csv("../datasets/oecd_bli_2015.csv")
print(df.columns)
print(df.shape)
print(df.head())
print(df.tail())
#print(df.info())
#print(df.describe())
#print("=== unique values of LOCATION ===")
#print(df["LOCATION"].unique())
#print(df["LOCATION"].value_counts())
#print(df["Unit"].dropna())
#print(df.sort_values("Value"))
#print(df.groupby("LOCATION"))
ndf = df[(df["Inequality"]=="Total") &
         (df["INDICATOR"] == "HO-BASE")]
#print(ndf)
```

Output –



Number of prime numbers in step of 200 from 0 to 10.000

TITLE: Data Cleaning - Prime Numbers	DATE: 28-3-22

Aim: write a program to find number of prime numbers using file handling techniques.

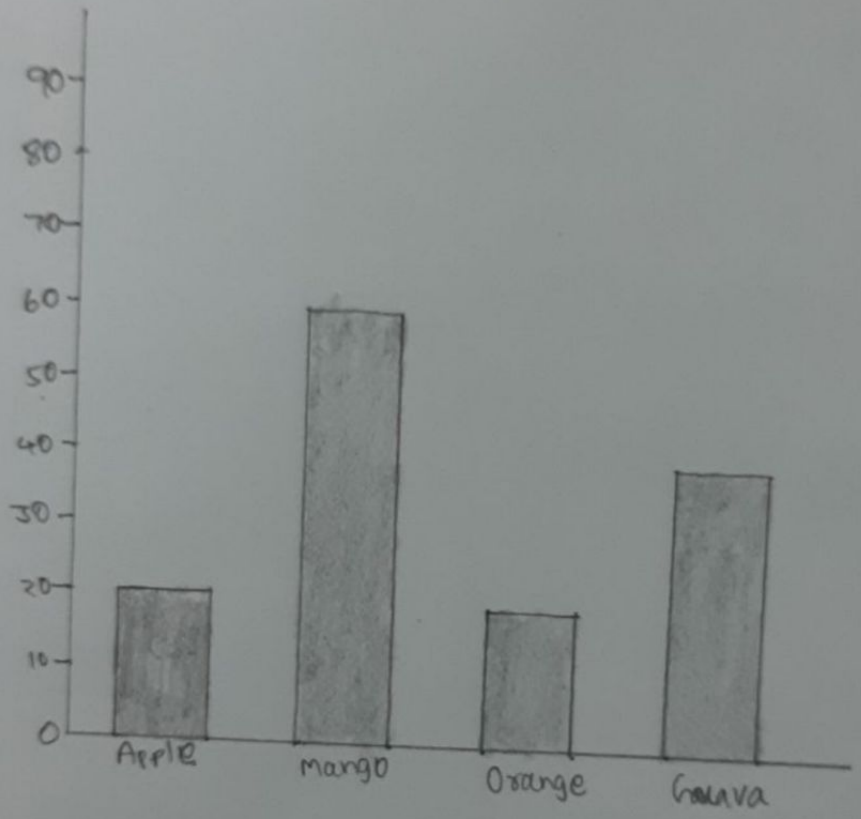
```
import os
import matplotlib.pyplot as plt
import numpy as np
END=10000
LEN=50
STEP=int(END/LEN)
def generate_numbers():
    with open("numbers.txt", "w") as lfile:
        lfile.write(str(2)+"\n")
        for i in range(3,END,2):
            lfile.write(str(i)+"\n")
def is_prime(lnumber):
    for i in range (2, lnumber):
        if lnumber % i==0:
            return False
    return True
def make-data():
    if os.path.exists("numbers.txt"):
        os.remove("numbers.txt")
    generate_numbers()
    if os.path.exists("primes.txt"):
        os.remove("primes.txt")
```

```
with open("numbers.txt", 'r') as lfile:
    for lline in lfile:
        lnumber = int(lline.strip("\n"))
        if is_prime(lnumber):
            with open("primes.txt","a") as primes:
                primes.writelines(lline)
if __name__ = "__main__":
    make_data()
    prime_range = [STEP*i for i in range(1,(LEN+1))]
    prime_count = [0 for i in range(1,(LEN+1))]
    COUNT=0
    primes=0
    with open("primes.txt","r") as file:
        LIMIT=STEP
        for line in file:
            primes+=1
            number = int(line.strip("\n"))
            if number <= LIMIT:
                prime_count[COUNT]+=1
            else:
                LIMIT+=STEP
                COUNT+=1
                prime_count[COUNT]+=1
    Print(primes)
    plt.scatter(prime_range, prime_count)
    plt.savefig("wlb-plot.png", format="png")
    plt.xticks(prime_range, rotation=45)
```

```
plt.xlabel("numbers")
plt.ylabel("# prime numbers")
plt.show()
```

Output -

Frequency of Apple : 20
Frequency of Mango : 60
Frequency of Orange : 20
Frequency of Guava : 40



TITLE: Word Frequency | DATE: 4-4-22

Aim: Write a program to find the word frequency of each individual word in the given text.

```
import pandas as pd
import matplotlib.pyplot as plt
def myfun( ):
    words_dict={ }
    with open("/content/words.txt","r") as f:
        for line in f:
            words=line.strip("\n").split(" ")
            for word in words:
                if word in words_dict:
                    words_dict[word]+=1
                else:
                    words_dict[word]=1
    return words_dict
if __name__=="__main__":
    wd=myfun( )
    for allkeys in wd:
        print("Frequency of ",allkeys, end=" ")
        print(":",end=" ")
        print(wd[allkeys], end=" ")
        print( )
    plt.bar(list(wd.keys()), list(wd.values()))
    plt.show( )
```

Output –

['age', 'sex', 'bmi', 'bp', 's1', 's2', 's3', 's4', 's5', 's6']

Coefficient : [938.23786125]

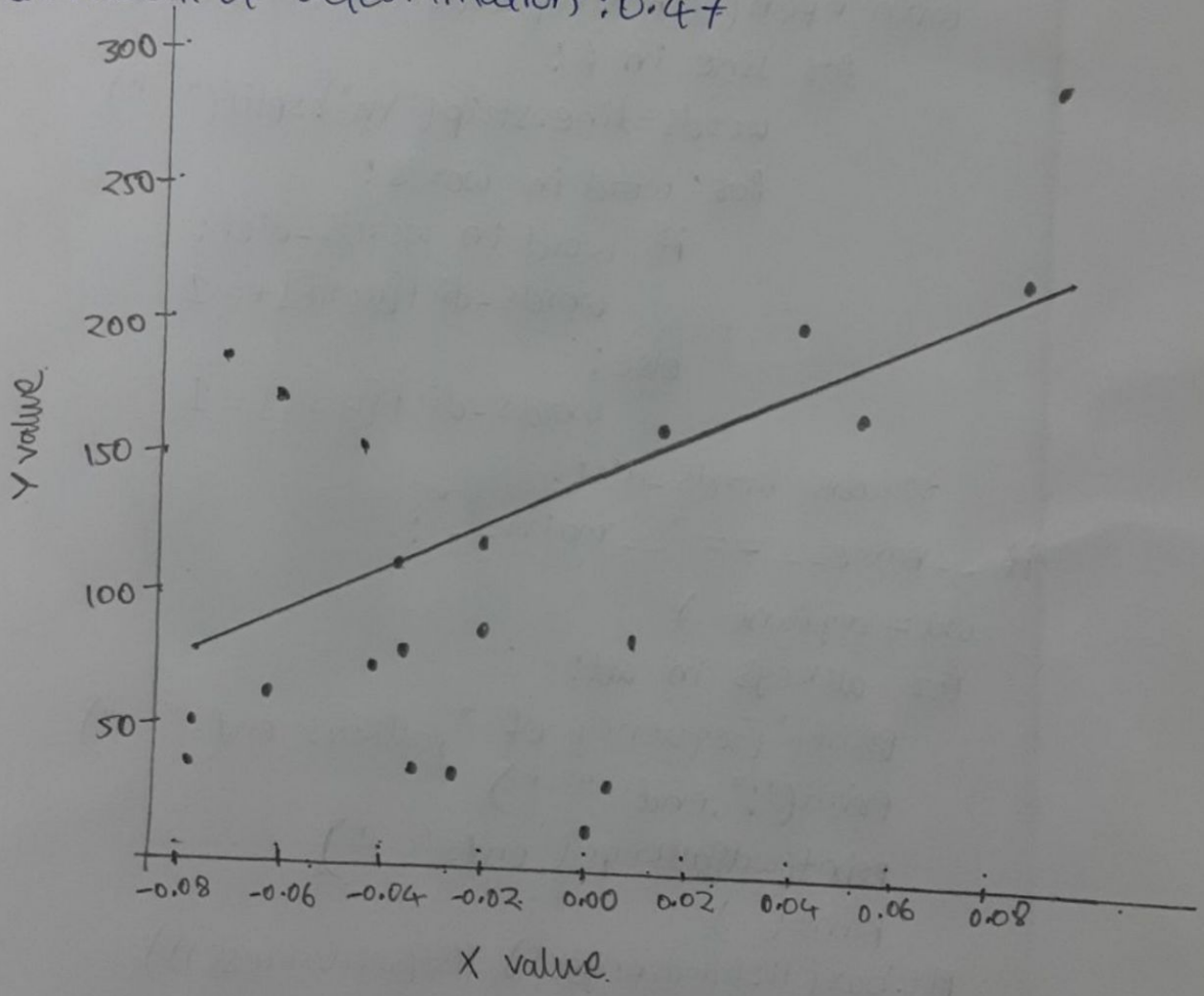
Constant : 152.91886182 6

Mean squared log error : 0.22

Mean squared error : 2548.07

Mean absolute error : 41.23

Coefficient of determination : 0.47



TITLE: Linear Regression-1 | DATE: 11-4-22

Aim: Implement a program for Linear Regression with a sklearn dataset

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets, linear_model
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error,
    r2_score, mean_absolute_error, mean_squared_log_error
diabetes_x, diabetes_y = datasets.load_diabetes(return_X_y
                                                =True)
data = datasets.load_diabetes(as_frame=True)
df = pd.DataFrame(data=data.data, columns=data.
                                    feature_names)
print(data.feature_names)
diabetes_x = diabetes_x[:, np.newaxis, 2]
diabetes_x_train = diabetes_x[:-20]
diabetes_x_test = diabetes_x[-20:]
diabetes_y_train = diabetes_y[:-20]
diabetes_y_test = diabetes_y[-20:]
regr = linear_model.LinearRegression()
regr.fit(diabetes_x_train, diabetes_y_train)
diabetes_y_pred = regr.predict(diabetes_x_test)
print("Coefficient :", regr.coef_)
```

```
Print("Const:", regr.intercept_)
Print("Mean squared log error: %.2f" %.mean_squared_
      log_error(diabetes_y_test, diabetes_y_pred))
Print("Mean squared error: %.2f" %.mean_squared_
      error(diabetes_y_test, diabetes_y_pred))
Print("Mean Absolute error: %.2f" %.mean_absolute_
      error(diabetes_y_test, diabetes_y_pred))
print("Coefficient of determination: %.2f"
    %.82_score(diabetes_y_test, diabetes_y_pred))
plt.scatter(diabetes_x_test, diabetes_y_test,
                                color="black")
plt.plot(diabetes_x_test, diabetes_y_pred, color="blue",
        linewidth=2)
plt.xlabel("x value")
plt.ylabel("Y Value")
plt.xticks()
plt.yticks()
plt.savefig("W2a-plot.png", format="png")
plt.show()
```

Output -

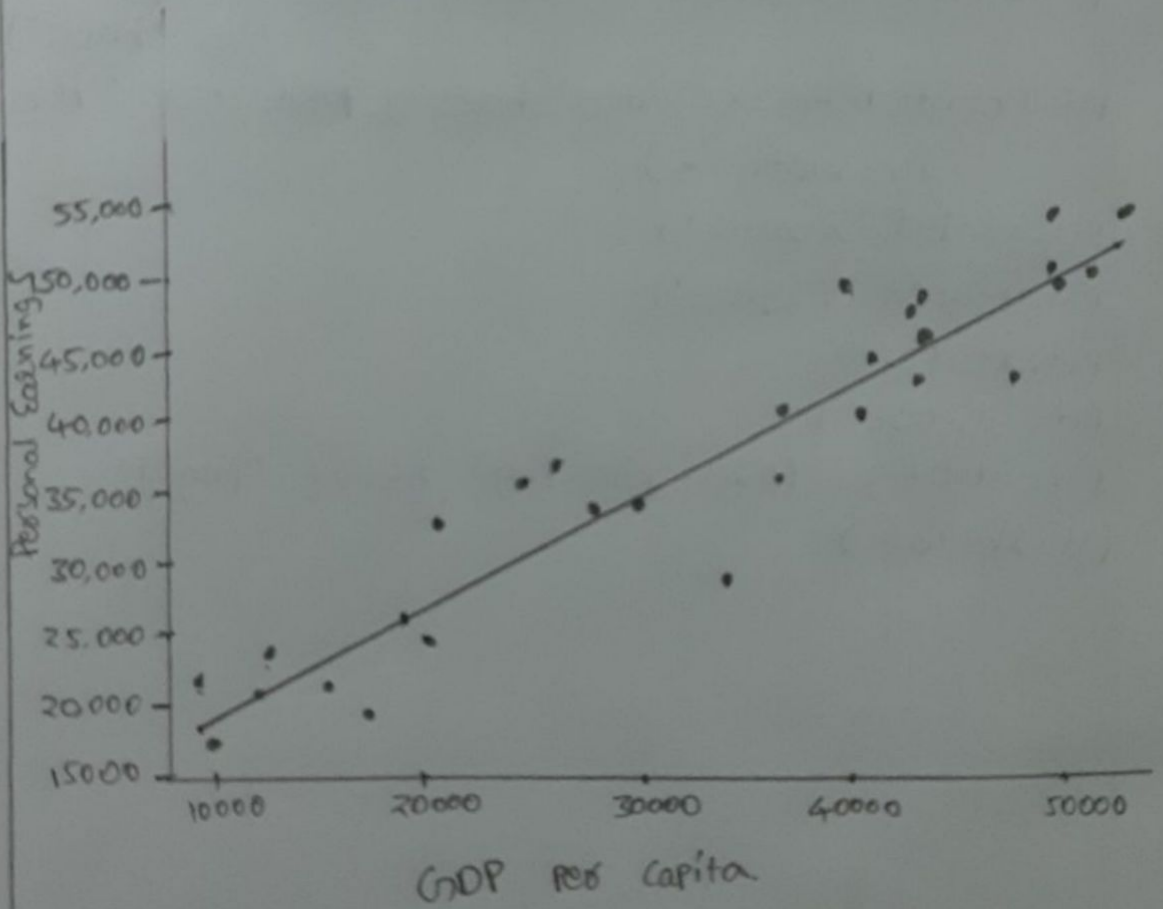
[[0.75362795]]

LinearRegression()

[9054.91] [20885.] [18265.6293]

[9437.37] [16919.] [18553.8618]

[55805.2] [56340.] [53497.9547]



TITLE: Linear Regression-2 | DATE: 18-4-22

Aim: Implement a program for Linear Regression by reading csv files

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sklearn.linear_model
def Prepare_country_stats(oecd_bli, gdp_per_capita):
    oecd_bli = oecd_bli[oecd_bli["INEQUALITY"]=="TOT"]
    oecd_bli = oecd_bli.pivot(index="Country",
                columns="Indicator", values="value")
    gdp_per_capita.rename(columns={"2015":"GDP per
    capita"}, inplace=True)
    gdp_per_capita.set_index("Country", inplace=True)
    full_country_stats = pd.merge(left=oecd_bli,
                right=gdp_per_capita, left_index=True,
                right_index=True).
    full_country_stats.sort_values(by="GDP per capita",
    inplace=True).
    remove_indices = [0,1,6,8,33,34,35]
    keep_indices = list(set(range(36)) - set(remove_indices))
    return full_country_stats[["GDP per capita", "Housing
    expenditure"]].iloc[keep_indices]
oecd_bli = pd.read_csv("../datasets/oecd_bli_2015.csv",
                thousands=',')
```

```
gdp_per_capita = pd.read_csv("../datasets/gdp_per_capita.csv",
                             thousands=',')
country_stats = prepare_country_stats(oecd_bli,
                                      gdp_per_capita)
country_stats.to_csv("gdp.txt",sep=";")
X = np.c_[country_stats["GDP per capita"]]
y = np.c_[country_stats["Housing expenditure"]]
model = sklearn.linear_model.LinearRegression()
model.fit(X,y)
X_new = [[22587]]
print(model.coef_)
print(model)
yp = model.predict(X)
for i in range(X.size):
    print(X[i], y[i], yp[i])
plt.scatter(X,y)
plt.xlabel("GDP per capita")
plt.ylabel("Housing expenditure")
plt.plot(X, yp)
plt.savefig("../datasets/gdp_per_capita_personal_earnings
                                        .png")
plt.show()
```

Output -

Coefficients:

[[0.000732]
 [-0.064531]
  .
  .
  .
 [-0.047356]]

Constant:

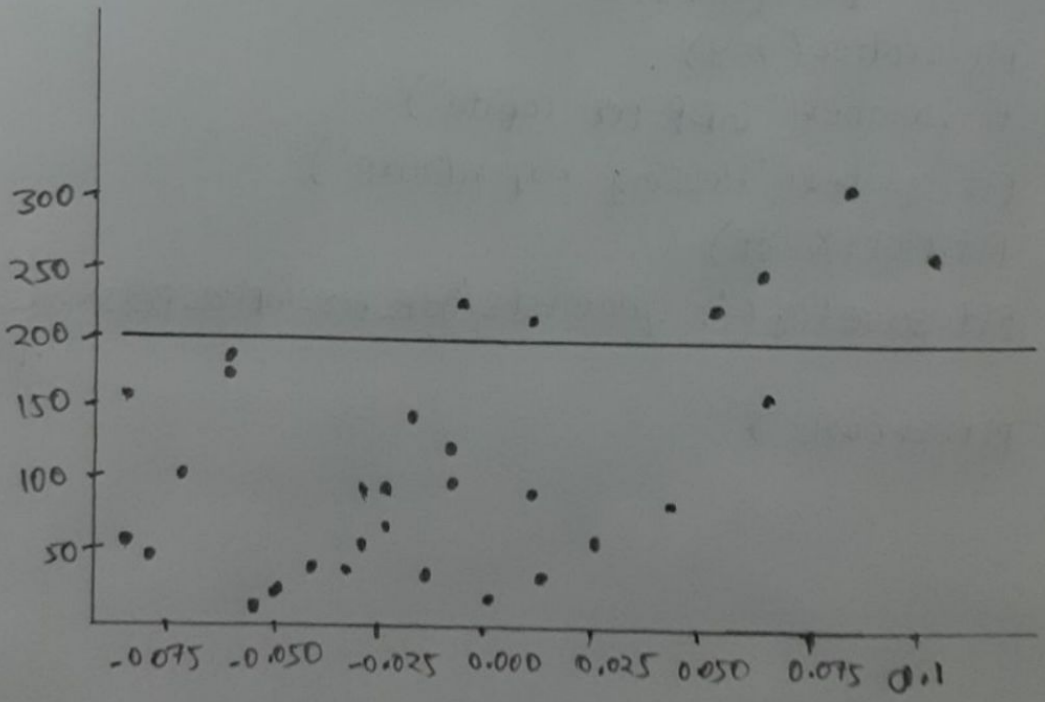
[-0.550179 -0.550153 _ _ _ .
 . . . _ _ _ _ _ -0.550159]

Mean Squared error: 9601.1

Coefficient of determination: -0.86

[261. 113. 131. _ _ _ .
 _ _ _ _ 200. 200.]

[[0 0 0 0 0 0 0 0 1 0 0 0 _ _ _ .]
 _ _ _ _ _ _
 _ _ _ _ _ _
 0 0 0 0 0 0 0 0 1 0 0 0 _ _ _]]



TITLE: Logistic Regression-1	DATE: 25-4-22

Aim: Implement a program for Logistic Regression with any sklearn dataset

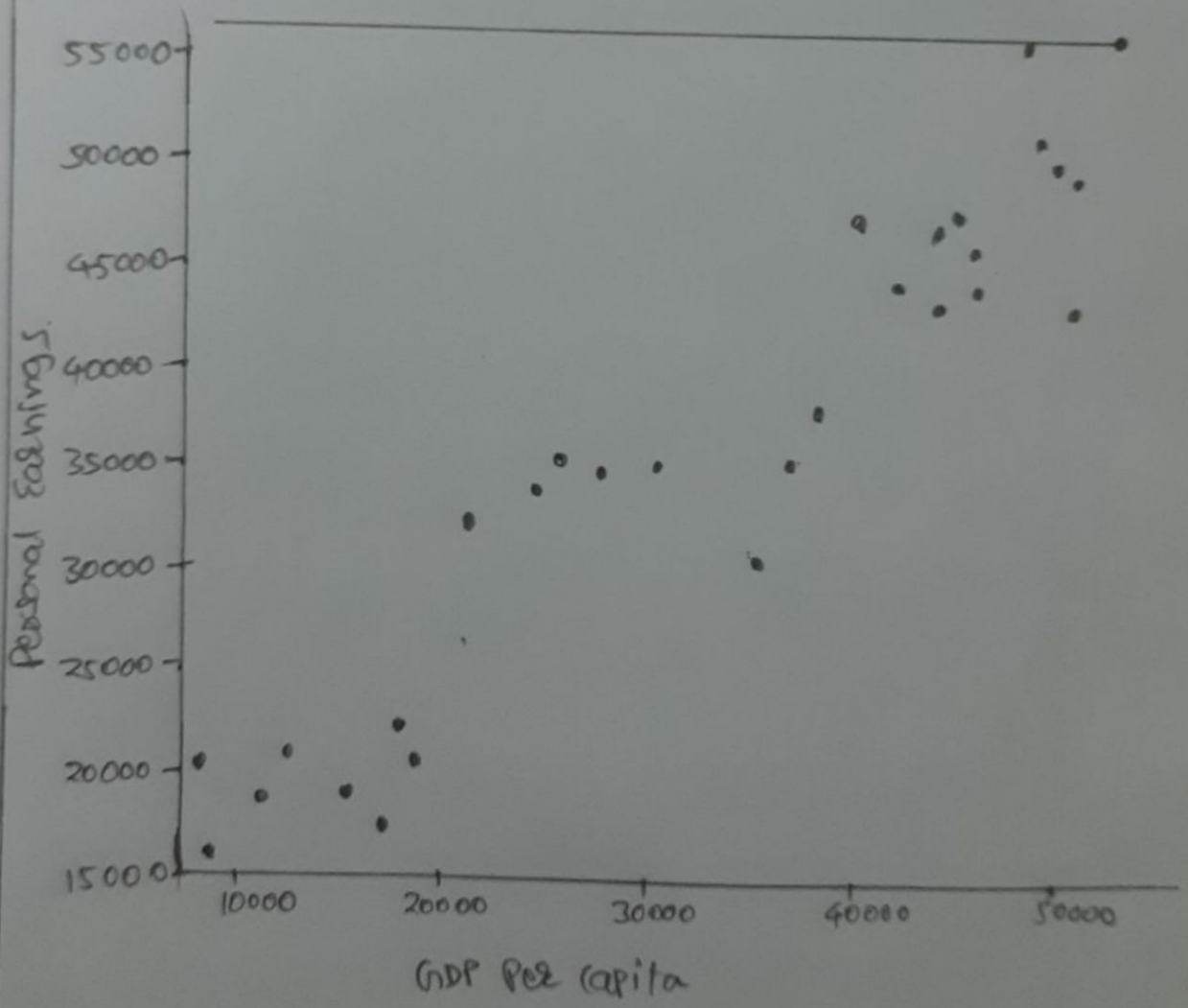
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error,
        r2_score, confusion_matrix
diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y
                                                =True)

print(type(diabetes_X))
diabetes_X = diabetes_X[:, np.newaxis,3]
diabetes_X_train = diabetes_X[:-30]
diabetes_X_test = diabetes_X[-30:]
diabetes_y_train = diabetes_y[:-30]
diabetes_y_test = diabetes_y[-30:]
regr = linear_model.LogisticRegression()
regr.fit(diabetes_X_train, diabetes_y_train)
diabetes_y_pred = regr.predict(diabetes_X_test)
print("Coefficients:\n", regr.coef_)
print("Constant:\n", regr.intercept_)
print("Mean Squared Error: %.2f" % mean_squared_error(
    diabetes_y_test, diabetes_y_pred))
```

```
Print("Coefficient of Determination: %.2f" % r2_score(
    diabetes_y_test, diabetes_y_pred))
plt.scatter(diabetes_X_test, diabetes_y_test, color="black")
plt.plot(diabetes_X_test, diabetes_y_pred, color="blue")
plt.xticks()
plt.yticks()
print(diabetes_y_test, diabetes_y_pred)
print(confusion_matrix(diabetes_y_test, diabetes_y_pred))
plt.show()
```

Output -

```
[[-3.10421 3e-05]
 ,
 ,
 [1.6358112e-05]]
LogisticRegression( )
[9054.91] [20885.] [56340.0
[9437.37] [16919.]  56340.0
,
|
.
[55805.2] [56340.] 56340.0.
```



TITLE: Logistic Regression-2 | DATE: 30-5-22

Aim: Implement a program for Logistic Regression by reading csv files

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import sklearn.linear_model
def prepare_country_stats(oecd_bli, gdp_per_capita):
    oecd_bli = oecd_bli[oecd_bli["INEQUALITY"]=="TOT"]
    oecd_bli = oecd_bli.pivot(index="Country",
                              columns="Indicator", values="Value")
    gdp_per_capita.rename(columns={"2015": "GDP per capita"},
                          inplace=True)
    gdp_per_capita.set_index("Country", inplace=True)
    full_country_stats = pd.merge(left=oecd_bli,
                                  right=gdp_per_capita,
                                  left_index=True, right_index=True)
    remove_indices = [0,1,6,8,33,34,35]
    keep_indices = list(set(range(36)) - set(remove_indices))
    return full_country_stats[["GDP per capita",
                               'personal earnings']].iloc[keep_indices]
oecd_bli = pd.read_csv("../datasets/oecd_bli_2015.csv",
                       thousands=',')
```

```
gdp_per_capita = pd.read_csv("../datasets/gdp_per_capita
                 .csv", thousands=',')
country_stats = prepare_country_stats(oecd_bli, gdp_per_
                capita)
country_stats.to_csv("gdp.txt", sep=";")
X=np.c_[country_stats["GDP per Capita"]]
y=np.c_[country_stats["Personal Earnings"]]
print(y.size)
y = y.reshape(y.size,)
model = sklearn.linear_model.LogisticRegression()
model.fit(X,y)
X_new=[[22587]]
print(model.coef_)
print(model._estimator_type)
yp=model.predict(X)
for i in range(X.size):
    print(X[i],y[i], yp[i])
plt.scatter(X, y)
plt.xlabel("GDP Per Capita")
plt.ylabel("Personal Earnings")
plt.plot(X,yp)
plt.savefig("../datasets/gdp_per_capita_personal_earnings
            .png", dpi=600)
plt.show()
```

Output -

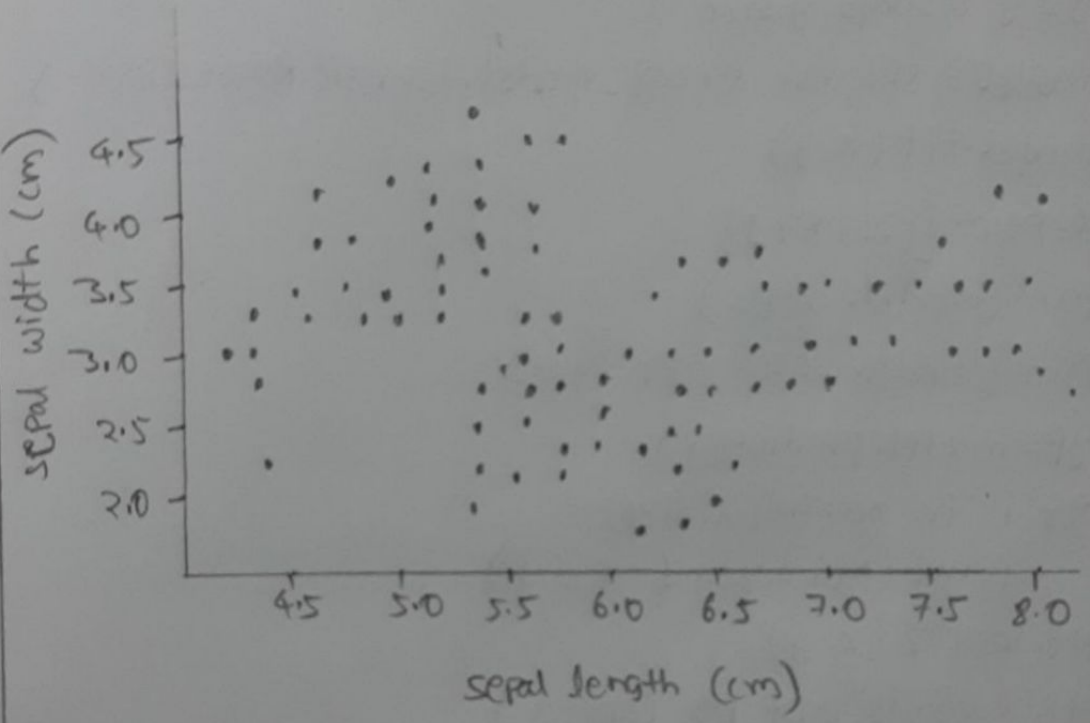
$y = 4.942 + -2.277 \times x_0 + 2.250 \times x_1$

$y = 6.179 + -2.069 \times x_0 + 1.723 \times x_1$

$y = 12.355 + -1.601 \times x_0 + -0.796 \times x_1$

Support Vectors

```
[[4.9  3. ]
 [5.7  3.8]
  .
  .
  .
 [5.9  3. ]]
```



TITLE: SVM-1	DATE: 6-6-22

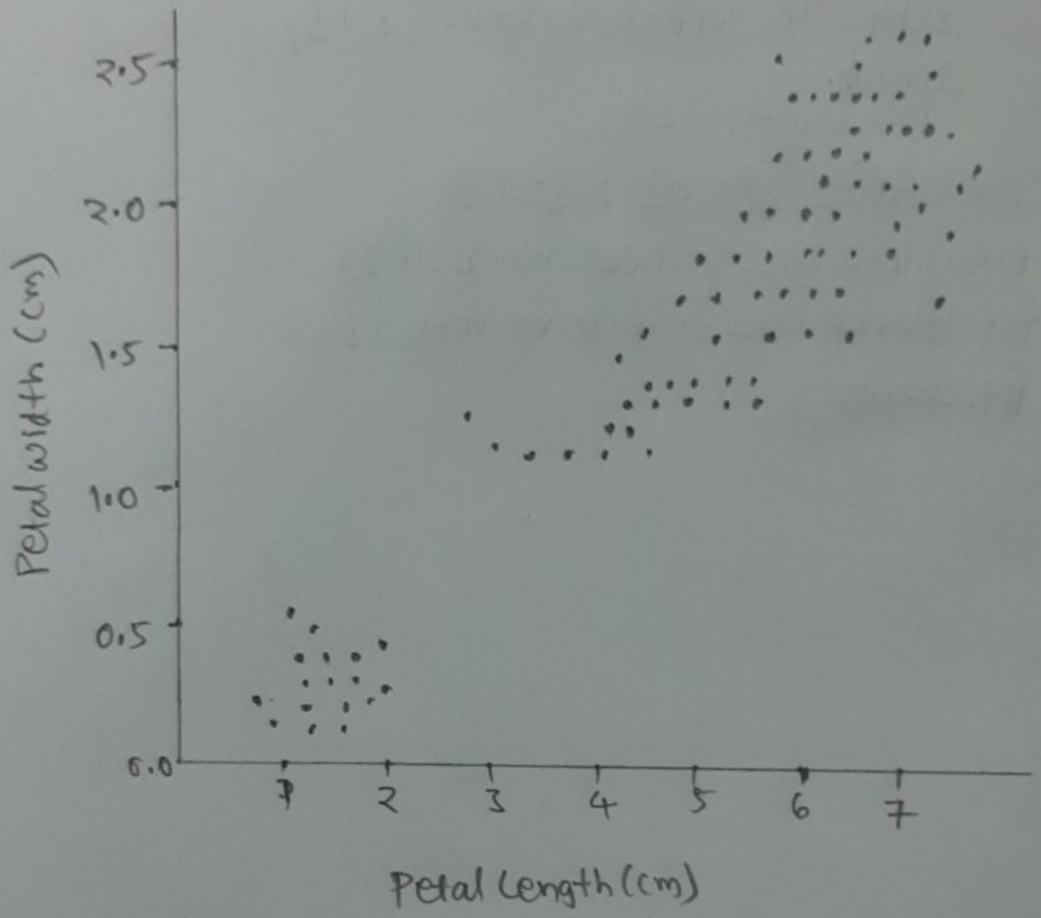
Aim: Write a program to construct a SVM classifier with a sklearn dataset

```
import numpy as np
from sklearn import datasets
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import LinearSVC
from sklearn import svm
import matplotlib.pyplot as plt
import pandas as pd
import sklearn.metrics as skm.
iris = datasets.load_iris()
x = iris["data"][:,(0,1)]
y = (iris["target"]).astype(np.float64)
svm_clf = svm.SVC(kernel="linear")
svm_clf.fit(x,y)
for (intercept, coef) in zip(svm_clf.intercept, svm_clf.coef):
    s="y={0:.3f}".format(intercept)
    for (i,c) in enumerate(coef):
        s+="+{0:.3f}*x{1}".format(c,i)
    print(s)
print(x)
```

```
print("Support Vectors")
print(svm_clf.support_vectors_)
plt.scatter(
    svm_clf.support_vectors_[:,0],
    svm_clf.support_vectors_[:,1],
    linear
    linewidth=1)
plt.scatter(X[:,0], X[:,1])
plt.xlabel(iris.feature_names[0])
plt.ylabel(iris.feature_names[1])
plt.show()
```

Output -

array([1])



TITLE: SVM-2 | DATE: 13-6-22

Aim: Write a program to construct a svm classifier

```
import numpy as np
from sklearn import datasets
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import LinearSVC
import matplotlib.pyplot as plt
import pandas as pd
iris = datasets.load_iris()
irisdf = pd.DataFrame(data=np.c_[iris['data'], iris['target']],
                      columns=iris['feature_names']+['target'])
irisdf.plot.scatter(x="petal length (cm)",
                    y="petal width (cm)")
svm_clf = Pipeline([("scaler", StandardScaler()),
        ("linear_svc", LinearSVC(C=1, loss="hinge")),])
svm_clf.fit(X, y)
print(svm_clf.predict([[5.5, 1.7]]))
x1, y1 = [1, 7], [1, 0.75]
x2, y2 = [2.5, 2], [3, 0]
x3, y3 = [1.5, 3], [2.5, 0]
```

```
plt.plot(x1, y1, x2, y2, x3, y3, marker = 'o')
plt.scatter(x[:,0], x[:,1])
plt.xlabel(iris.feature_names[2])
plt.ylabel(iris.feature_names[3])
plt.show()
```

Output -

array([1])

Training Accuracy: 0.969230

Testing Accuracy: 0.9

2	2
2	2
2	2
2	1
2	1
2	2
2	2
2	2
2	2
2	2
2	2
2	2
2	2
2	2
2	2
2	2
2	2
2	2
2	2
2	2

TITLE: Decission Tree	DATE: 20-6-22

Aim: Write a program to demonstrate working of Decision Tree

```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
iris = load_iris()
X_train = iris.data[:-20, 2:] #Petal length and width
y_train = iris.target[:-20]
X_test = iris.data[-20:, 2:]
y_test = iris.target[-20:]
tree_clf = DecisionTreeClassifier(max_depth=2, random_state=42)
tree_clf.fit(X_train, y_train)
tree_clf.predi
print(tree_clf.predict([[5, 1.5]]))
print("Training Accuracy:", tree_clf.score(X_train, y_train))
print("Testing Accuracy:", tree_clf.score(X_test, y_test))
y_pred = tree_clf.predict(X_test)
for i in range(0,20):
    print(y_test[i], "  ", y_pred[i])
```