INDEX

NAME: SREEJA KATTA STD.: ____ SEC.: D ROLL NO.: 1BM21CS247 SUB.: ML LAB

S. No.	Date	Title	Page No.		Teacher's Sign / Remarks
1	21/03/24	Importing and Exporting Data	1	10	21/3/24
2	28/03/24	Hands-on-Machine Approach	2	10	28/3/24
3	04/04/24	Linear and Multiple Regression	5	10	4/4/24
4	18/4/24	Decision Tree	7	10	18/4/24
5	25/4/24	Logistic Regression	8	10	25/4/24
6	09/5/24	KNN Classification	10	10	9/5/24
7	09/5/24	SVM	11		
8	23/5/24	ANN	12	10	23/5/24
9	23/5/24	Random forest and AdaBoost	14		
10	30/5/24	K Means clustering	16	10	30/5/24
11	30/5/24	PCA	17		

Program-1: Import and Export using Pandas

Importing Data

```
import pandas as pd
housing_data = pd.read_csv("/content/sample_data/california_housing_test.csv")
housing_data.head()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population
0	-122.05	37.37	27.0	3885.0	661.0	1537.0
1	-118.30	34.26	43.0	1510.0	310.0	809.0
2	-117.81	33.78	27.0	3589.0	507.0	1484.0

Reading data from URL

```
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
col_names = ["sepal_length_in_cm", "sepal_width_in_cm", "petal_length_in_cm", "petal_width_in_cm",
             "class"]

iris_data = pd.read_csv(url, names=col_names)
iris_data.head()
```

	sepal_length_in_cm	sepal_width_in_cm	petal_length_in_cm	petal_width_in_cm	class
0	5.1	3.5	1.4	0.2	Iris-setosa

Dec
21/3/24

Step-1 : Get the data

Download the data

```
import os
import tarfile
import urllib

DOWNLOAD_ROOT = "https://raw.githubusercontent.com/ageron/handson-ml2/master/"
HOUSING_PATH = os.path.join("data", "01")
HOUSING_URL = DOWNLOAD_ROOT + "datasets/housing/housing.tgz"

def fetch_housing_data(housing_url=HOUSING_URL, housing_path=HOUSING_PATH):
    os.makedirs(name=housing_path, exist_ok=True)
    tgz_path = os.path.join(housing_path, "housing.tgz")
    urllib.request.urlretrieve(url=housing_url, filename=tgz_path)
    housing_tgz = tarfile.open(name=tgz_path)
    housing_tgz.extractall(path=housing_path)
    housing_tgz.close()

fetch_housing_data()

import pandas as pd

def load_housing_data(housing_path=HOUSING_PATH):
    data_path = os.path.join(housing_path, "housing.csv")
    return pd.read_csv(data_path)
```

Step 2: Discover & Visualize the Data to Gain Insights

```
strat_train_set.shape, strat_test_set.shape

strat_test_set.reset_index().to_feather(fname='data/01/strat_test_set.f')

housing = strat_train_set.copy(); housing.shape
```

Visualizing Geographical Data

```
housing.plot(kind='scatter', x='longitude', y='latitude')
plt.show()

housing.plot(kind='scatter', x='longitude', y='latitude', alpha=0.1)
plt.show()

housing[['population', 'median_house_value']].corr()
```

3. Prepare the Data for Machine Learning Algorithms

Data Cleaning

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy='median')
housing_num = housing.drop("ocean_proximity", axis=1)
```

Custom Transformation Pipelines

```
num_pipeline = Pipeline([
        ('imputer', SimpleImputer(strategy='median')),
        ('attribs_adder', CombinedAttributesAdder()),
        ('std_scaler', StandardScaler())
    ])

housing_num_tr = num_pipeline.fit_transform(housing_num)
housing_num_tr.shape
```

4. Select and Train a Model

```
lin_reg = LinearRegression()
lin_reg.fit(X=housing_prepared, y=housing_labels)

some_data = housing.iloc[:5]
some_labels = housing_labels.iloc[:5]
some_data_prepared = full_pipeline.transform(some_data)
```

~~some_data_prep~~

```
housing_predictions = lin_reg.predict(housing_prepared)
lin_mse = mean_squared_error(housing_labels, housing_predictions)
lin_rmse = np.sqrt(lin_mse)
lin_rmse
```

5. Fine-Tune Your Model

```
param_grid = [
  {'n_estimators': [3, 10, 30], 'max_features': [2, 4, 6, 8]},
  {'bootstrap': [False], 'n_estimators': [3, 10], 'max_features': [2, 3, 4]}
]

forest_reg = RandomForestRegressor()
grid_search = GridSearchCV(estimator=forest_reg, param_grid=param_grid,
                           scoring='neg_mean_squared_error', cv=5, return_train_score=True,
                           n_jobs=-1)

grid_search.fit(X=housing_prepared, y=housing_labels)

grid_search.best_params_
grid_search.best_estimator_
```

Sneha 28/3/24

Linear Regression

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from pandas.core.common import random_state
from sklearn.linear_model import LinearRegression

df_sal = pd.read_csv('/content/sample_data/salary_data.csv')
df_sal.head()
df_sal.describe()

plt.title('Salary Distribution Plot')
sns.distplot(df_sal['Salary'])
plt.show()

plt.scatter(df_sal['Years Experience'], df_sal['Salary'], color='lightcoral')
plt.title('Salary vs Experience')
plt.box(False)
plt.show()

X = df_sal.iloc[:, :1]
y = df_sal.iloc[:, 1:]

regressor = LinearRegression()
regressor.fit(X_train, y_train)

y_pred_test = regressor.predit(X_test)
y_pred_train = regressor.predict(X_train)

plt.scatter(X_test, y_test, color='lightcoral')
plt.plot(X_train, y_pred_train, color='firebrick')
plt.title('Salary vs Experience')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.box(False)
plt.show()

print(f'Coefficient : {regressor.coef_}')
print(f'Intercept : {regressor.intercept_}')
```

Multiple Regression

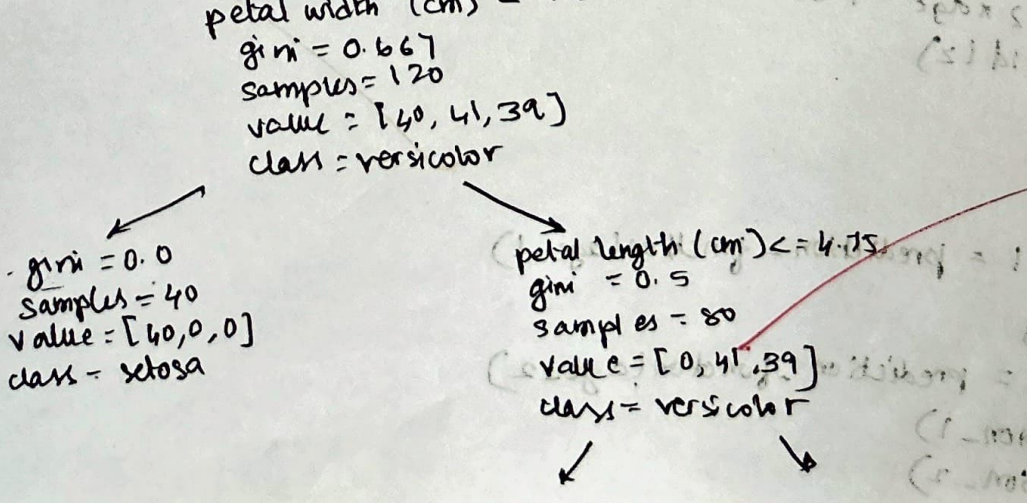
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.linear_model import LinearRegression

df_start = pd.read_csv('/content/sample_data/50_startups.csv')
df_start.head()
df_start.describe()

plt.title('Profit Distribution Plot')
sns.distplot(df_start['Profit'])
plt.show()

plt.scatter(df_start['R&D Spend'], df_start['Profit'], color='lightcoral')
plt.xlabel('R&D Spend')
plt.ylabel('Profit')
plt.box(False)
plt.show()

X = df_start.iloc[:, :-1].values
y = df_start.iloc[:, -1].values

X = np.array(ct.fit_transform(X))

regressor = LinearRegression()
regressor.fit(X_train, y_train)

y_pred = regressor.predict(X_test)

np.set_printoptions(precision=2)
result = np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),
                                                                       1)),1)

result
```

4/14/24

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree

#dataset
iris_data = load_iris()
iris_df = pd.DataFrame(data= iris_data.data, columns = iris_data.feature_names)
iris_df['target'] = iris_data.target
print(iris_df.head())

X= iris_df.drop('target', axis=1)
y = iris_df['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.23, random_state=42)

clf = DecisionTreeClassifier(criterion="gini", random_state=100, max_depth=4, min_samples_leaf=8)
clf.fit(X_train, y_train)

accuracy = clf.score(X_test,y_test)
print("Accuracy:", accuracy)

plt.figure(figsize=(12,8))
tree.plot_tree(clf, feature_names=iris_data.feature_names, class_names=iris_data.target_names,
               filled=True)

plt.show()
```

OUTPUT

Accuracy: 80.98



Sneha 18/4/24

Program 5: Logistic Regression 25/04/24

Predicting if a person would buy life insurance based on his age using logistic regression

```
import pandas as pd
import matplotlib import pyplot as plt
%. matplotlib inline

df = pd.read_csv("/content/sample_data/insurance_data.csv")
df.head()
plt.scatter(df.age, df.bought_insurance, marker='+', color='red')

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df[['age']], df.bought_insurance,
                                                    train_size=0.8)

from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, y_train)

y_predicted = model.predict(X_test)
model.predict_proba(X_test)
model.score(X_test, y_test)

print("Coefficient: ", model.coef_)
print("Intercept: ", model.intercept_)

import math

def sigmoid(x)
return 1/(1+math.exp(-x))

def prediction_function(age):
z = 0.042 * age - 1.53
y = sigmoid(z)
return y

age_1 = 35
prediction_1 = prediction_function(age_1)

age_2 = 43
prediction_2 = prediction_function(age_2)
print(prediction_1)
print(prediction_2)
```

OUTPUT

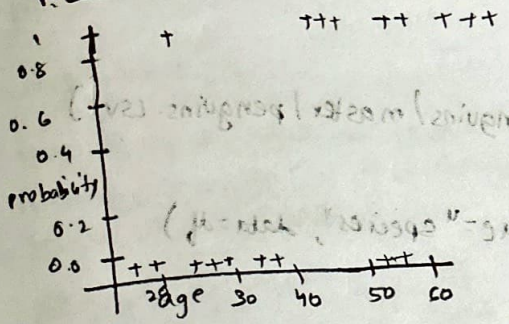
accuracy = 0.6666666666

coefficient : [[0.16265891]]

Intercept : [-6.81930256]

Prediction 1 is 0.4850044983

Prediction 2 is 0.5685652990

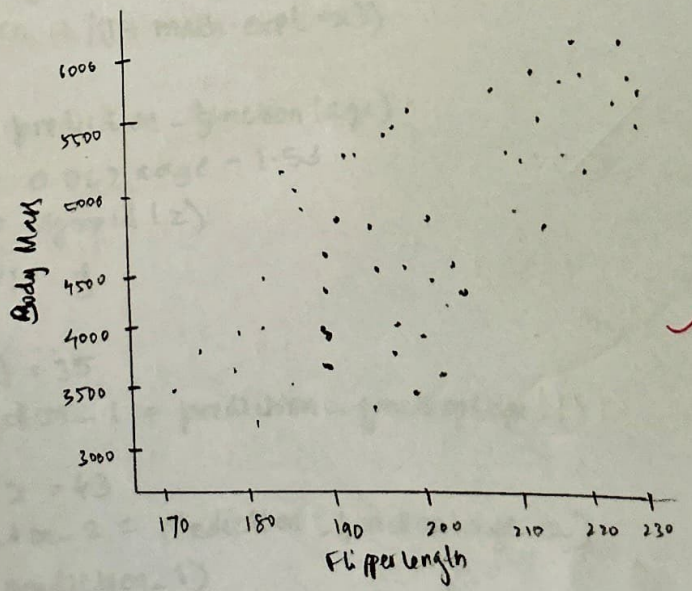


25/4/24

09/05/24

Program 6: KNN classification model

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.neighbours import KNeighborsClassifier
from sklearn.metrics import accuracy_score

df = pd.read_csv("https://raw.githubusercontent.com/palmerpenguins/master/penguins.csv")
print(df.head())
sns.scatterplot(x="flipper_length_mm", y="body_mass_g", hue="species", data=df)
plt.title("Penguin Species Classification")
plt.xlabel("Flipper Length (mm)")
plt.ylabel("Body Mass (g)")
plt.show()

df.dropna(inplace=True)

X = df[['flipper_length_mm', 'body_mass_g']]
y = df['species']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
knn = KNeighborsClassifier(n_neighbours=3)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

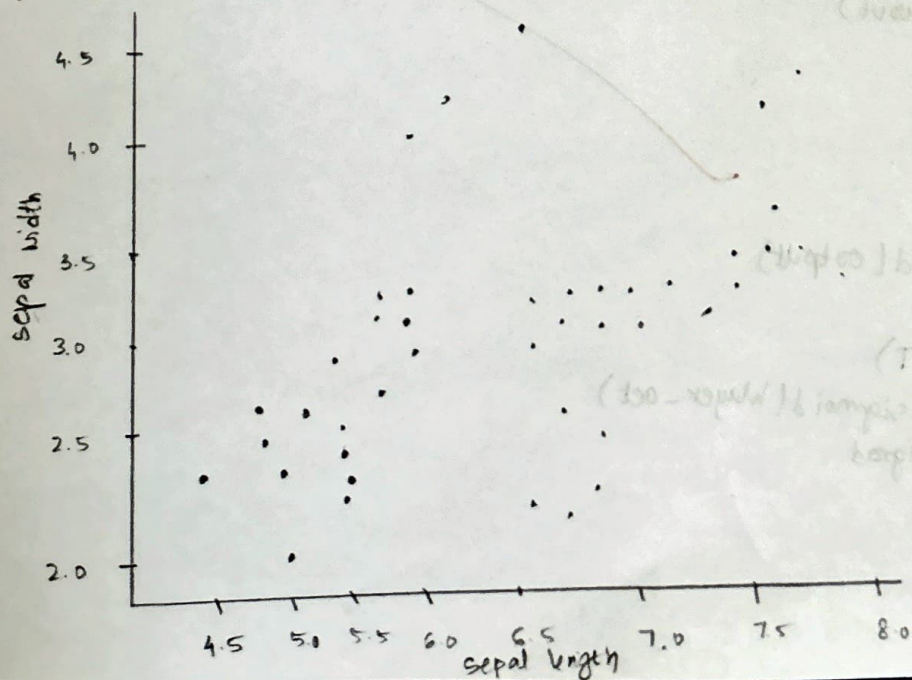


Accuracy: 0.6865671641

Program 7 : Support Vector Machine model

09/05/24

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.dataset import load_iris
from sklearn.model_selection import train_test_split
from sklearn.svm import svc
from sklearn.metrics import accuracy_score

iris = load_iris()

iris_df = pd.DataFrame(data = iris_data, columns = iris.features_names)
iris_df['target'] = iris.target
print(iris_df.head())

plt.figure(fig size = (10, 6))
plt.scatter(iris_df['sepal length (cm)'], iris_df['sepal width (cm)'],
            c = iris_df['target'], cmap='viridis')

plt.xlabel('sepal length (cm)')
plt.ylabel('sepal width (cm)')
plt.colorbar(label = 'species')
plt.show()

X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, test_size=0.6,
                                                    random_state = 32)

svm_classifier = svc(kernel = 'linear')
svm_classifier.fit(X_train, y_train)

y_pred = svm_classifier.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print("Accuracy of SVM classifier:", acuracy)
```



Sneha 9/5/24

Acuracy : 0.977777

Program 8 : Artificial Neural Networks


```
import numpy as np
X = np.array([[2,9],[1,5],[3,6]], dtype=float)
y = np.array([[92],[86],[89]], dtype=float)
X = X/np.amax(X, axis=0)
y = y/100
epoch = 5000
lr = 0.1
inputlayer_neurons = 2
hiddenlayer_neurons = 3
output_neurons = 1

wh = np.random.uniform(size=(inputlayer_neurons, hiddenlayer_neurons))
bh = np.random.uniform(size=(1, hiddenlayer_neurons))
wout = np.random.uniform(size=(hiddenlayer_neurons, output_neurons))
bout = np.random.uniform(size=(1, output_neurons))

def sigmoid(x):
    return 1/(1+np.exp(-x))

def derivatives_sigmoid(x):
    return x*(1-x)

for i in range(epoch):
    hinp1 = np.dot(X, wh)
    hinp = hinp1 + bh
    hlayer_act = sigmoid(hinp)
    outinp1 = np.dot(hlayer_act, wout)
    outinp = outinp1 + bout
    output = sigmoid(outinp)

    EO = y - output
    outgrad = derivatives_sigmoid(output)
    d_output = EO * outgrad
    EH = d_output.dot(wout.T)
    hiddengrad = derivatives_sigmoid(hlayer_act)
    d_hiddenlayer = EH * hiddengrad
```

```
wout += hlayer_act.T.dot(d_output) * lr
wh += X.T.dot(d_hiddenlayer) * lr

print("Input : \n" + str(X))
print("Actual Output: \n" + str(y))
print("Predicted output: \n",output)
```

Input :

```
[[0.666667  1.      ]
 [0.333333  0.555556]
 [1.        0.666667]]
```

Actual output :

```
[[0.92]
 [0.86]
 [0.89]]
```

Predicted output :

```
[[0.76787301]
 [0.75248028]
 [0.77506492]]
```

23/05/24

Program 9(a): Random Forest Algorithm

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

iris = load_iris()
X = iris.data
y = iris.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=32)
rf_classifier = RandomForestClassifier()
rf_classifier.fit(X_train, y_train)
y_pred = rf_classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

OUTPUT

Accuracy: 0.933333

Classification Report

	precision	recall	f1-score	support
0	1.00	1.00	1.00	30
1	0.81	1.00	0.89	21
2	1.00	0.79	0.88	24
accuracy			0.93	75
macro avg	0.94	0.93	0.93	75
weighted avg	0.95	0.93	0.93	75

Program 9(b) : AdaBoost


```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score

iris = load_iris()
X = iris.data
y = iris.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=32)
adaboost_clf = AdaBoostClassifier(n_estimators=30, learning_rate=1.0, random_state=42)
adaboost_clf.fit(X_train, y_train)
y_pred = adaboost_clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

OUTPUT

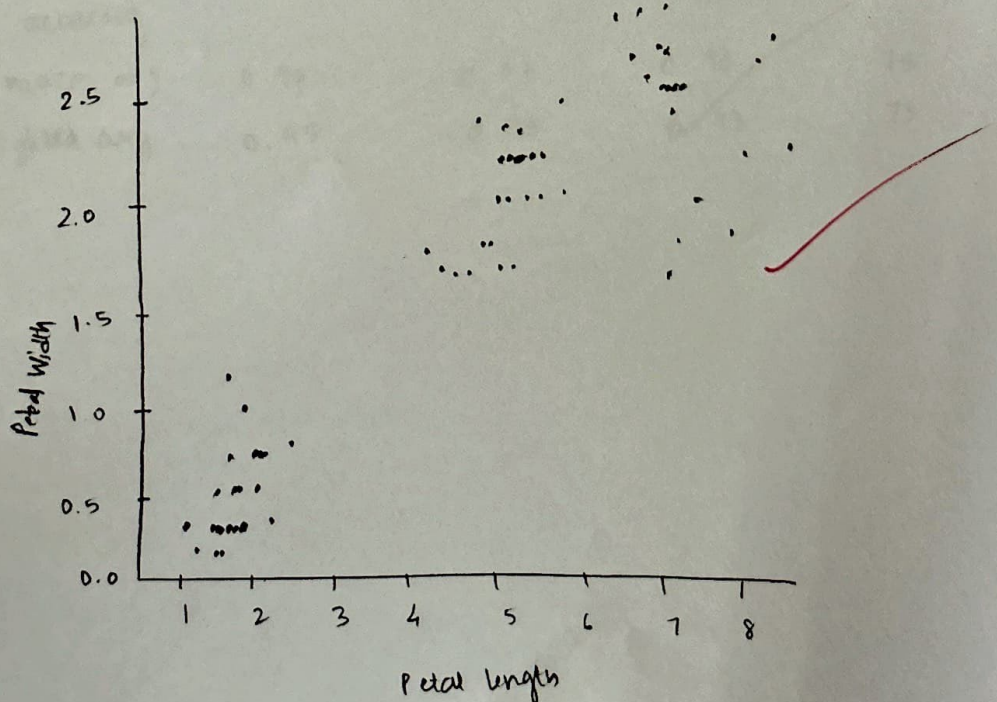
```
Accuracy: 0.966667
```

Lucky
23/5/24

30/05/24

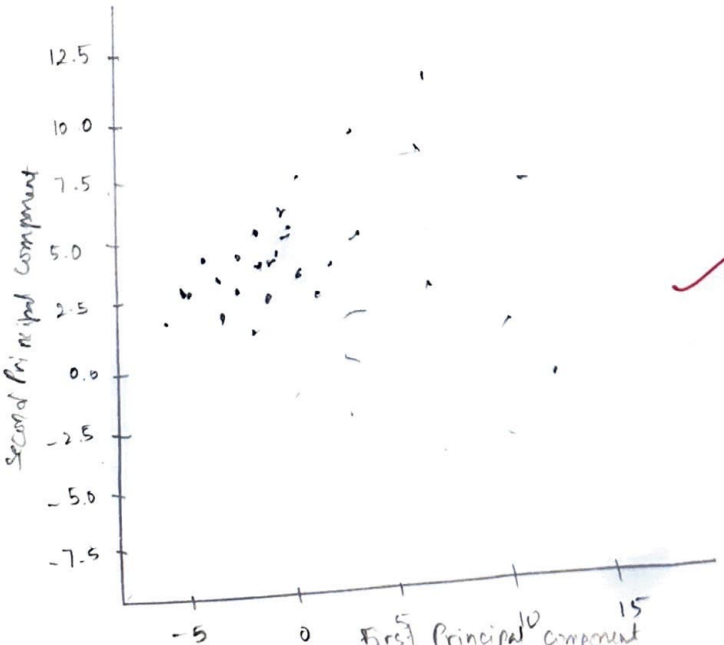
Program 10: K Means clustering

```
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.cluster import KMeans
import pandas as pd
import numpy as np

iris = datasets.load_iris()
X = pd.DataFrame(iris.data)
X.columns = ['Sepal_Length', 'Sepal_Width', 'Petal_Length', 'Petal_Width']
y = pd.DataFrame(iris.target)
y.columns = ['Targets']

model = KMeans(n_clusters=3)
model.fit(X)

plt.figure(figsize=(14,14))
colormap = np.array(['red', 'lime', 'black'])
plt.subplot(2,2,1)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[model.labels_], s=40)
plt.title('K-Means clustering')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
```



Program 11: Principal Component Analysis (PCA)


```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
%matplotlib inline

from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()
cancer.keys()
print(cancer['DESCR'])
df = pd.DataFrame(cancer['data'], columns = cancer['feature_names'])
df.head()

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(df)
scaled_data = scaler.transform(df)

from sklearn.decomposition import PCA
pca = PCA(n_components = 2)
pca.fit(scaled_data)
x_pca = pca.transform(scaled_data)
scaled_data.shape

x_pca.shape
plt.figure(figsize = (8, 6))
plt.scatter(x_pca[:,0], x_pca[:,1], c= cancer['target'], cmap = 'plasma')
plt.xlabel('First Principal Component')
plt.ylabel('Second Principal component')
```



30/5/24