INDEX

NAME: Yashasirni.M STD.: 6 SEC.: D ROLL NO.: 2 52 SUB.: ML

S. No.	Date	Title	Marks / Page No.	Teacher's Sign / Remarks
1	21-03-2024	Import & export	10	21/3/24
2	28-03-2024	End-to-End ML Project	10	28/3/24
3	04-04-2024	Linear Regression	9	18/4/24
4	18-08-2024	Decision Tree	10	
5	25-04-2024	Logistic Regression	10	25/4/24
6	09-05-2024	KNN ~~and SVM~~	10	9-5
7	09-05-2024	SVM	10	9-5
8	23-05-2024	ANN	10	23/5/24
9.a	23-05-2024	Random Forest		
9.b	23-05-2024	Ada Boost		
10	30-005-2024	K-Means	10	
11	30-05-2024	PCA	10	

GKC0223951

21-03-2024

Week-1

Importing and Exporting

Using 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()
```

OUTPUT

	sepal-length-in-cm	sepal-width-in-cm	petal-length-in-cm
0	5.1	3.5	1.4

petal-width-in-cm	class
0.2	Iris-setosa

```
# to export
df.to_csv('cleaned_iris_data.csv')
# import without url
df = pd.read_csv("C:/Users/Admin/Downloads/Iris.csv",
names=["sepal_length-in-cm", "sepal-width-in-cm",
"petal_length-in-cm", "petal-width-in-cm", "class"])
```

OUTPUT

	sepal-length-in-cm	sepal-width-in-cm	petal-length-in-cm
0	5.1	3.5	1.4

petal-width-in-cm	class
0.2	Iris-setosa

21/3/24

28-03-2024

End-to-End Machine Learning Project

1. Frame the problem, select the performance measure
   - Root mean square error

2. Get the Data

```
# download data
imports pandas as pd
data_path = os.path.join(housing_path, "housing.csv")
pd.read_csv(data_path)

# checking data structure
housing.head()
housing.info() # gives information of each attribute about datatype

# all are numerical except for ocean_proximity.

# housing.describe()
# gives information like mean, count, std etc for
  attributes with numerical values

# create histogram

import matplotlib.pyplot as plt
import seaborn as sns

housing.hist(bins=50, figsize=(20,15))
plt.show() # gives in form of graphs.

# splitting train test data

from sklearn.model_selection import
train_test_split

# 80% - train set, 20% test set.

train_set, test_set = train_test_split(housing, test_size=0.2,
  random_state=42)

train_set.shape; test_set.shape
```

3. Discover & visualize the Data to gain Insights

Visualize geographical Data.

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

to estimate densities, set alpha = 0.1

housing.plot(kind:"scatter", x="longitude", y:"latitude",
alpha = 0.1)

correlation using scatter matrix.

from pandas.plotting import scatter_matrix

attributes = ['median_house_value', 'median_income',
'total_rooms', 'housing_median_age']

scatter_matrix(frame: housing[attributes], figsize=(12,8))
plt.show()

4. Prepare data for machine learning algorithm.

Data cleaning.

from sklearn.impute import SimpleImputer.

imputer = SimpleImputer(strategy: "median")

25/3/24

Handling text & categorical attributes.

from sklearn.preprocessing import OrdinalEncoder
ordinal_encoder = OrdinalEncoder()
housing_cat_encoded = ordinal_encoder.fit_transform(housing_cat.values)
housing_cat_encoded.shape

one attribute is 1 if category is equal to <1HOCEAN
otherwise zero. Similarly for others.

from sklearn.preprocessing import OneHotEncoder.

one_hot_encoder = OneHotEncoder()

housing_cat_1hot = one_hot_encoder.fit_transform(housing_cat.res
hape
gives 1 or 0

Feature Scaling

there is Min Max Scaling and standardization
that is done using transformation pipeline

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

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

5. Select & Train a Model

```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X = housing_prepared, y = housing_labels)

from sklearn.tree import DecisionTreeRegressor
tree_reg = DecisionTreeRegressor()
```

Cross validation

```
from sklearn.model_selection import cross_val_score
score = cross_val_score(estimator = tree_reg X=housing_prepa,
                        y=housing_labels, scoring=neg_mean_squared_error,
                        cv=10)
```

using random forest for ~~latest tree~~ ensemble learning.

6. Fine - Tune

grid search

```
from sklearn.model_selection import GridSearchCV
param_grid = [{'n_estimators': [3,10,30], 'max_features':
                [2,4,6,8]},
    {'bootstrap': [False], 'n_estimators': [3,10], 'max_features':
                [2,3,4]}
```

```
grid_search = GridSearchCV( estimator = forest_reg , param_grid =
        param_grid , scoring = 'neg_mean_squared_error',
                                cv = 5 )
```

04-04224

Week-3

Simple Linear Regression

```
import numpy as np
import matplotlib.pyplot as plt

~~def estimate_coef~~
import pandas as pd
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/Salary_Data.csv')
df_sal.head()

# Analyze
df_sal.describe()
plt.title('Salary Distribution Plot')
sns.distplot(df_sal['Salary'])
plt.show()
plt.scatter(df_sal['YearsExperience'], df_sal['Salary'],
    color='lightcoral')
plt.title('Salary vs Experience')
plt.xlabel('Years of experience')
plt.ylabel('Salary')
plt.box(False)
plt.show()
# split data .
X = df_sal.iloc[:, :1]

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

```
X_train , x_test , y_train , y_test = train_test_split ( X , y,
                         test_size = 0.2 , random_state = 0 )

# train model

  regressor = LinearRegression ( )
  regressor . fit ( X train , y_trai )

# predict

 y_pred_test = regressor . predict ( x_test )
 y_pred_train - regressor . predict ( x_train )

  plt . scatter ( X_train , y_train , color = 'lightcoral' )
  plt . plot ( x_train , y_pred_train , color = 'firebrick' )
  plt . title ( 'Salary vs Experience ( Training Set )' )
  plt . xlabel ( 'Years of experience' )
  plt . ylable ( 'Salary' )
  plt . legend ( [ 'X_train/Pred (y_test' , 'X_train/y_train' ],
    title = 'Sal/Exp' , loc = 'best' , facecolor = 'white' )
    plt . box ( False )
  plt . show ( )

plt . scatter ( X_test , y_test , color = 'lightcoral' )
plt . plot ( X_train , y_pred_train , color = 'firebrick' )
plt . title ( 'Salary vs Experince ( Training Set )' )
plt . xlabel ( 'Years of experience' )
 plt . ylable ( 'Salary' )
  plt . legend ( [ 'X_train/Pred(y_test)' } , 'X_train/y_train' ],
  title = 'Sal / Exp' , loc = 'best' , facecolor = 'white' )
   plt . box(False)
 plt . show ( )

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

Multiple 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 sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.linear_model import LinearRegression

df_start = pd.read_csv('/content/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.title('Profit vs R&D Spend')
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
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),
    [3])], remainder='passthrough')
X = np.array(ct.fit_transform(X))

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 0.2, random_state = 0)
```

```
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)
```

15-04-24

Week-4
Decision Tree

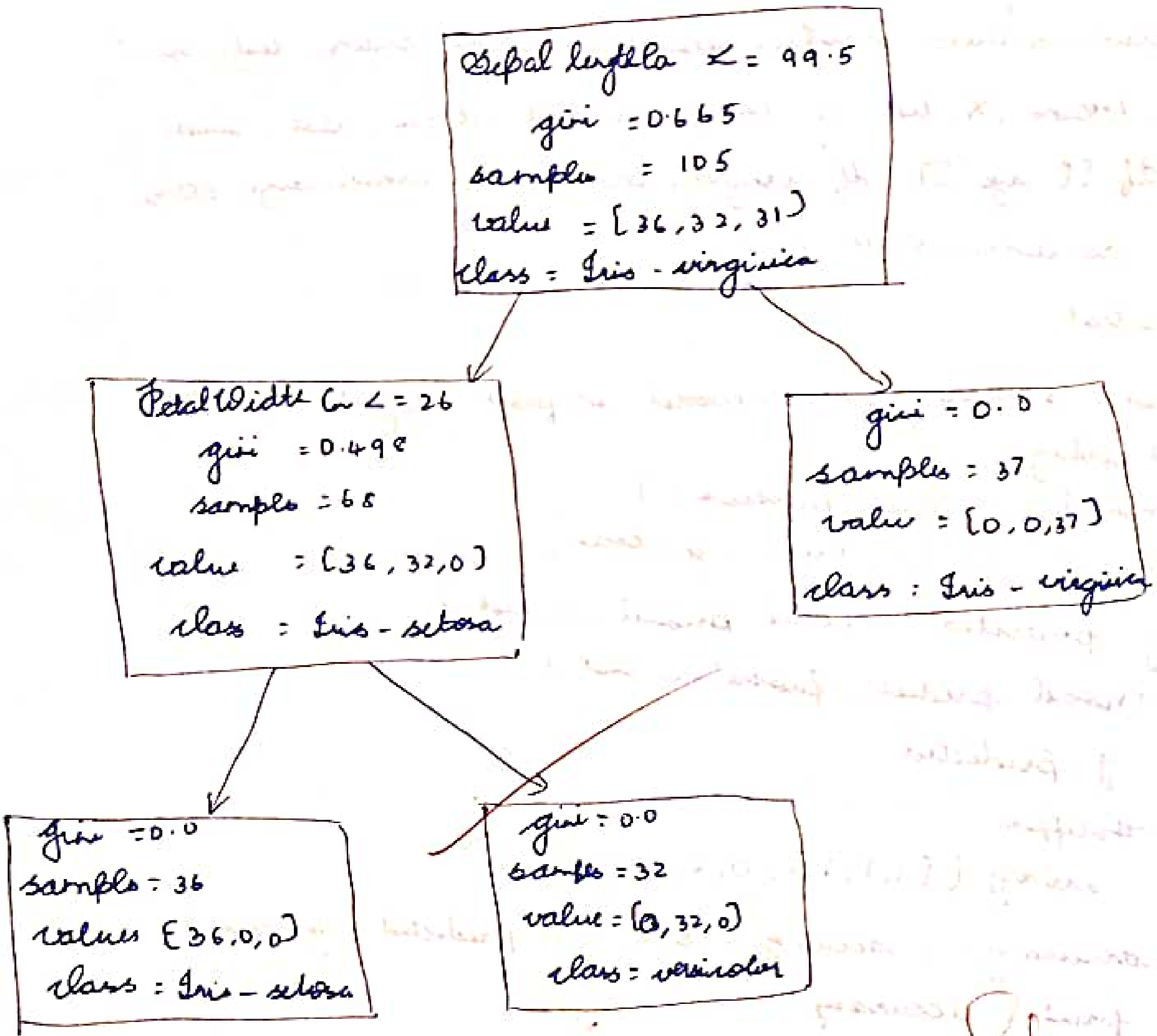
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn import datasets

iris = datasets.load_iris()
iris_df = pd.DataFrame(data = iris.data, columns = iris.
    feature_names)
iris_df['species'] = iris.target
iris_df['species'] = iris_df['species'].map({0: 'setosa', 1:'versicolor',
        2:'virginica'})
X = iris_df.drop('species', axis=1)
y = iris_df["species"]
X_train, X_test, y_train, y_test = train_test_split(X, y,
      test_size = 0.33, random_state = 42)
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier(criterion = 'gini',
   random_state = 100, max_depth = 5, min_samples_leaf=2)
model.fit(X_train, y_train)
from sklearn.metrics import accuracy_score
y_pred = model.predict(X_test)
print('Accuracy of Decision Tree: ', accuracy_score(y_pred,
                                                     y_test))
```

OUTPUT :

Accuracy of Decision Tree : 0.98

```
from sklearn.tree import plot_tree
plt.figure(figsize=(8,6))
plot_tree(model, feature_names=['SepalLength Cm',
'SepalWidth Cm', 'PetalLength Cm', 'PetalWidth Cm'],
class_names=['setosa', 'versicolor', 'virginica'], filled=True,
rounded=True)

model2 = DecisionTreeClassifier()
model2.fit(X_train, y_train)
y_pred = model2.predict(X_test)
```

Accuracy : 0.98



18/4/24

25.04.24

Week - 5
Logistic Regression

```
import pandas as pd
from matplotlib import pyplot as plt
% matplotlib inline
from sklearn.metric import accuracy_score
df = pd.read_csv("/content/insurance data.csv")
df.head()
# plotting points
plt.scatter(df.age, df.brought_insurance, marker='+', color='red')

# splitting
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
df[['age']], df.baught_insurance, train_size=0.6,
random_state=0)
X_test

from sklearn.linear_model import LogisticRegression.
# fitting
model = LogisticRegression()
model.fit(X_train, y_train)
y_predicted = model.predict(X_test)
model.predict_proba(X_test)
y_predicted
# output
array([1,1,1,1,0,0,0,0,0,0])
accuracy = accuracy_score(y_predicted, y_test)
print(accuracy)
0.818181
```

#with sigmoid function

```
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=35
prediction_function(age)
0.48500

age = 86
prediction_function(age)
0.8891
```

Sreta 25/4/24

09-05-24

Week. 6

Lab. 6 : K Nearest Neighbors and
7 SVM

importing modules

```
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
```

getting dataset

```
df = pd.read_csv("/content/Iris.csv")
df.head()
```

seperating independent and dependent variable.

```
X = df.drop("Species", axis=1)
y = df["Species"]
```

plotting scatterplots for variables.

```
plt.figure(figsize=(10,6))
sns.scatterplot(data=df, x=df["PetalLengthCm"],
  y=df["SepalWidthCm"], legend=False, hue=df["Species"])

plt.xlabel('Petal Length')
plt.ylabel('Sepal Width')
plt.title('Scatter Plot of Petal length vs Sepal
   Width')

plt.show()
```

```
# splitting to train and test.
X_train, X_test, y_train, y_test = train_test_split
        (X, y, train_size = 0.6, random_state = 0)

~~from~~ .
# KNN model
 neigh = KNeighborsClassifier(n_neighbors=2)
 neigh.fit(X, y)

 y_predicted = neigh.predict(X_test)
 accuracy = accuracy_score(y_predicted, y_test)
 print(accuracy)

  1.0
```

Program - 7

```
# SVM model

  model = SVC()

  model.fit(X, y)

  y_predicted : mod . predict(x_test)
  accuracy = accuracy_score(y_predicted, y_test)
  print(accuracy)
  1.0
```

9.5

4.5.24

23-05-24

Week-7

ANN
(Artificial Neural Network)

```
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

# initialize variables
epoch = 5000
lr = 0.1
input layer_neurons = 2
hiddenlayer_neurons = 3
output_neurons = 1

# weight and bias
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 derivative_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)
dhiddenlayer = 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:

```
[[0.66   1.  ]
 [0.33   0.55]
 [1.     0.66]]
```

Actual Output:

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

Predicted Output:

```
[[0.8455]
 [0.8406]
 [0.846]]
```

23-05-24

Random Forest

```
from sklearn.ensemble import RandomForestClassifier
import pandas as pd
from sklearn.metrics import accuracy_score

df = pd.read_csv("/content/drive/My Drive/melb_data.csv
melbourne_data = df.dropna(axis=0)
y = melbourne_data.Price
melbourne_feature = ['Rooms', 'Bathroom', 'Latitude']

X = melbourne_data(melbourne_features)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
                    train_size=0.8, random_state=0)
model = RandomForestClassifier()
model.fit(X y)

y_predicted = model.predict(X_test)
accuracy = accuracy_score(y_predicted, y_test)
```

0.9935

Ada Boost

```
df = pd.read_csv ("/content/drive/My Drive/Iris.csv")
X = df.drop("Species", axis=1)
y = df["Species"]
X_train, X_test, y_train, y_test = train_test_split
    (X, y, train_size = 0.8, random_state=0)
from sklearn.ensemble import AdaBoostClassifier
model = AdaBoostClassifier(n_estimators = 50)
model.fit (X, y)
y_predicted = model.predict (X_test)
accuracy = accuracy_score(y_predicted, y_test)

1.0

# With logistic regression.
from sklearn.linear_model import LogisticRegression

log = LogisticRegression()
ada = AdaBoostClassifier(n_estimators = 150, base_estimator=
                                               log,
                           learning_rate = 1)

model = ada.fit (X, y)

y_pred = model.predict (X_test)
print ("Accuracy:", accuracy_score (y_test, y_pred))
Accuracy : 1.0
```

23/5/24

30-05-2024

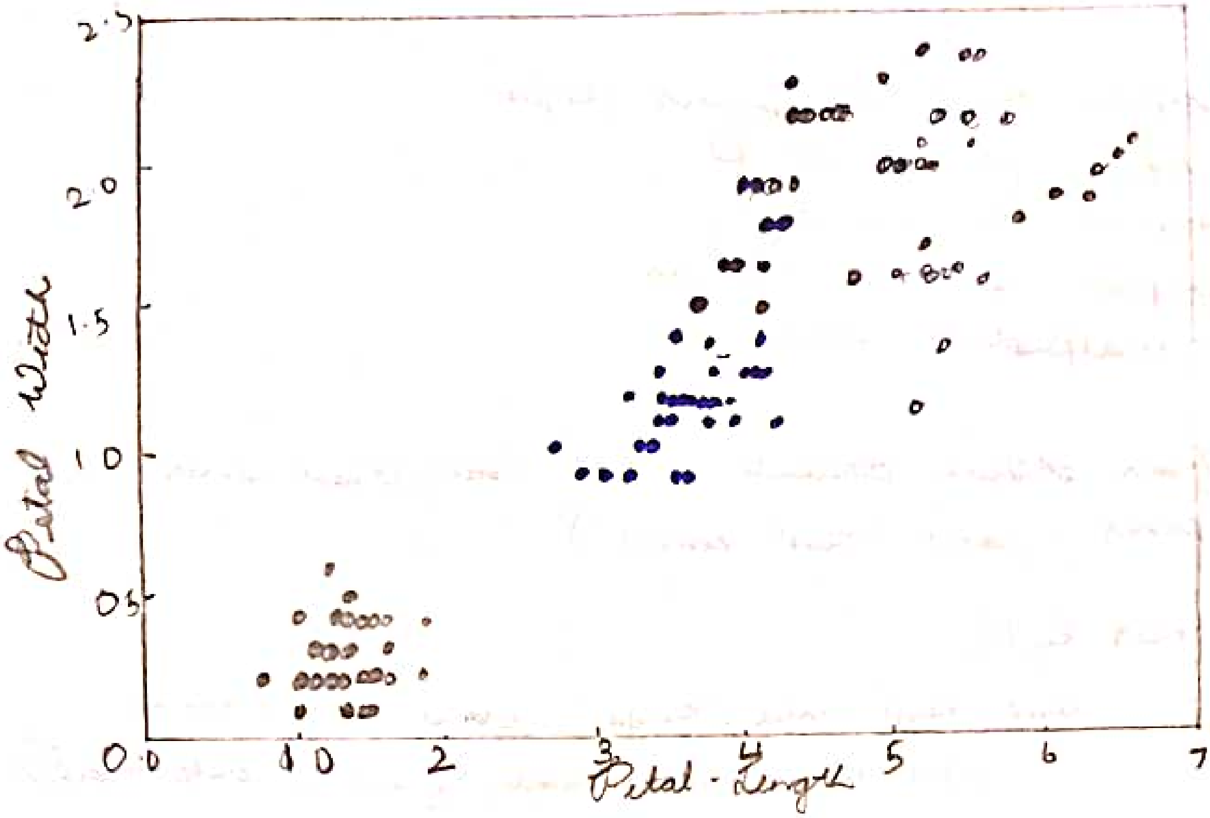
Week-8

K-Means

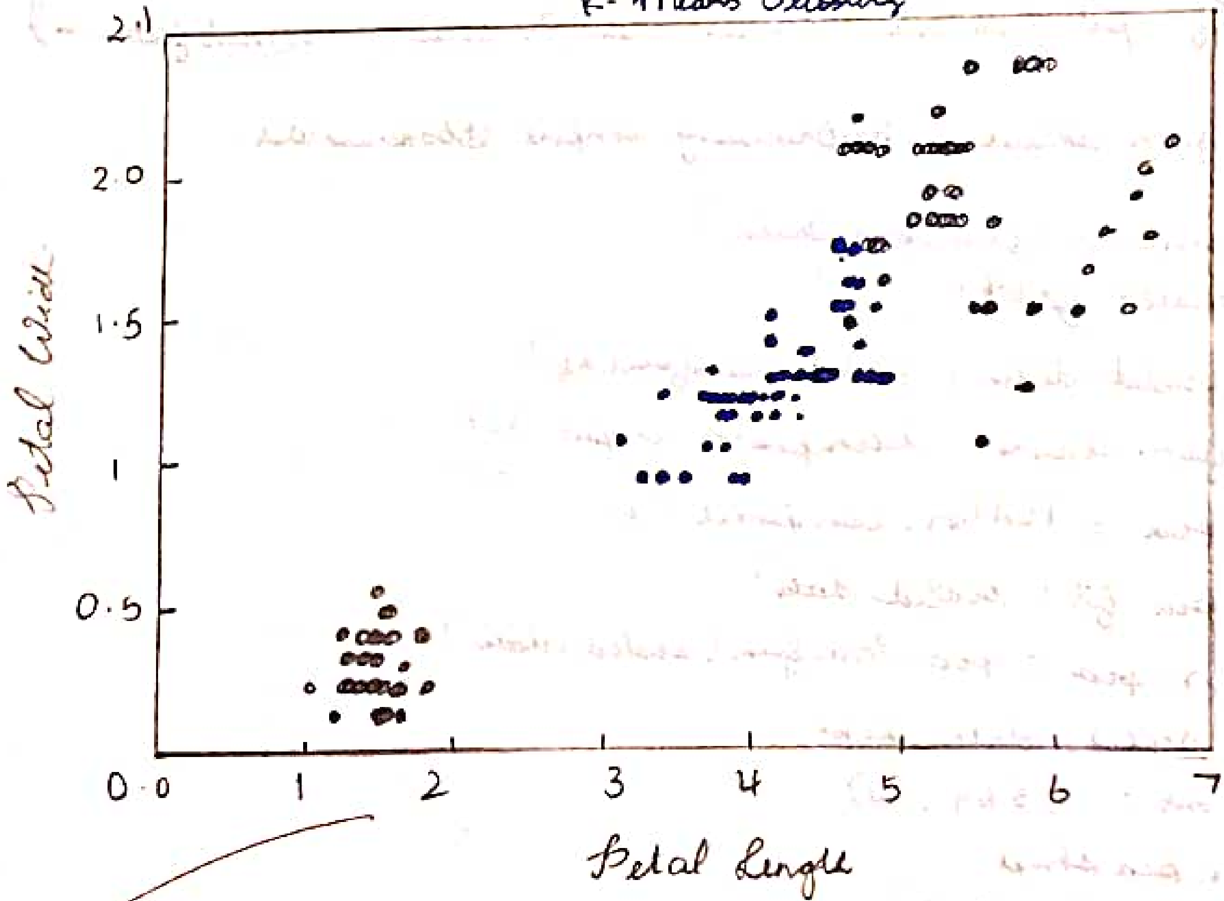
```
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.cluster import KMeans
import pandas as pd
import numpy as np

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[y.Targets], s=40)
plt.title('Real Clusters')
plt.xlabel('Petal Width')
plt.ylabel('Petal Length')
```

~~plt.title(~~

```
plt.subplot(2,2,2)
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')
```



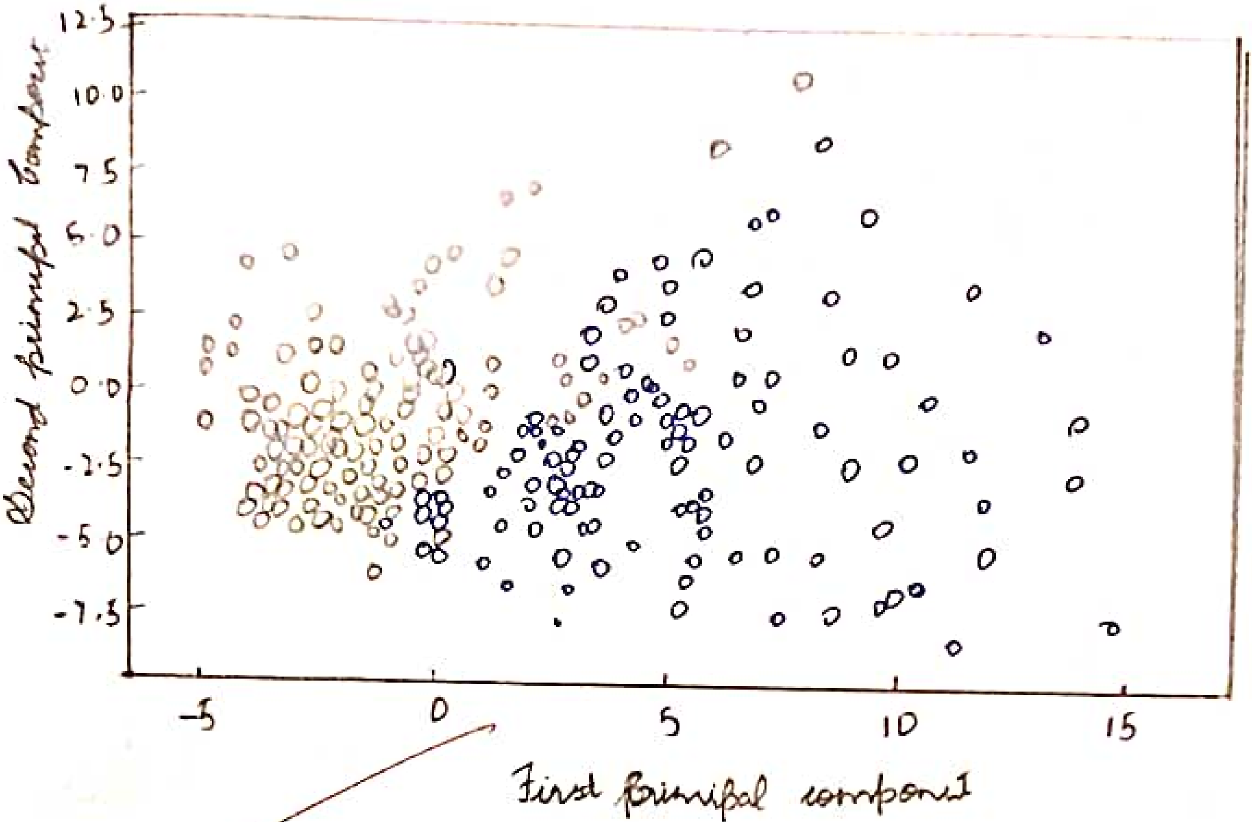
K- Means Clustering



30-05-2024

PCA

```
impor 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.key()
OUT : dict_key(['data', 'target', 'frame', 'target_names',
        'DESCR', 'feature_names', 'filename', 'data_module'])
print (cancer['DESCR')

df = pd.DataFrame(cancer['data'], columns = cancer['feature_n'])

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(df)

scaled_data = scaler.transform(df)
from sklearn.decomposition import PCA
pca = PCA(n_component = 2)
pca.fit( scaled_data)
x_pca = pca.transform(scaled_data)
scaled_data.shape
out : (569, 30)
x_pca.shape
out : (569, 2)
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')
```



12.5
10.0
7.5
5.0
2.5
0.0
-2.5
-5.0
-7.5
-5
0
5
10
15
Second principal compon
First principal component

Dd

20/5/02