INDEX

NAME: Sony Mahato STD.: _____ SEC.: _____ ROLL NO.: _____ SUB.: _____

ML Lab.

S. No.	Date	Title	Page No.	Teacher's Sign / Remarks
		INDEX		
	21/3/24	Import & Export Datasets pandas		
	4/4/24	End to End Project		4/4/24
	18/4/24	Linear & Multiple		
		Regression.		18/4/24
	25/4/24	Decision Tree Algorithm		
	9/5/24	Logistic Regression		
	9/5/24	SVM - IrisDataset, KNN		25/4/24
	23/5/24	ANN using BP		
	23/5/24	Random Forest Algorithm		9/5/24
	23/5/24	ADA Boost		
	30/5/24	K-means clustering		30/5/24
	30/5/24	PCA (Implement)		30/5/24

```
fetch_housing_data()
import pandas as pd
def load_housing_data(housing_path="H
    path):
    data_path = os.path.join(housing_path,"
        housing.csv")
    return pd.read_csv(data_path)
```

Create a test set

```
import numpy as np
def split_train_test(data, test_ratio=0
    shuffled_indices = np.random.permuta
        (len(data))
    test_set_size = int(len(data) * test_rat
```

Step-2 Discover & visualize

```
start_train_set.shape, start_test_set.
    shape.
start_test_set.reset_index().to_feather
    (fname="data/01/start_test_set.f")
housing = start_train_set.copy(); housing.
    shape
```

Visualize

```
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()
```

Step 3 Prepare the data for ML Algorithms

Data cleaning

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

Fine Tune

```
param_grid = [{'n_estimators': [3,10,30]
'max_features': [2,4,6,8]}]
forest_reg = RandomForestRegressor ()
grid_search = GridSearchCV(estimators
    forest_reg, param_grid = paramgr
```

grid_search_best_params_

grid_search.best_estimator_

cures = grid_search.cv_results_

ॐ॥॥॥

Week: 03

Implement linear and multiple linear regression algorithm.

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

def estimate_coef(x, y):
    n = np.size(x)
    m_x = np.mean(x)
    m_y = np.mean(y)

    SS_xy = np.sum(y*x) - n*m_y*m_x
    SS_xx = np.sum(x*x) - n*m_x*m_x

    b_1 = SS_xy / SS_xx
    b_0 = m_y - b_1*m_x
    return (b_0, b_1)

def plot_regression_line(x, y, b):
    plot.scatter(x, y, col='m', s=30)
    y_pred = b[0] + b[1]*x
```

```
plt.plot(x, y_pred, color='g')
plt.xlabel('x')
plt.ylabel('y')

def main():
x = array = [0,1,2,3,4,5,6,7,8,9,10
y=np.array([1,3,2,5,7,8,8,9,10,12
b= estimate_coff(x,y)
print("Estimated coefficient:\n b_0=
```

Multiple linear regression.

```
from sklear_model_selection import
train_text

import matplot_lib pyplot as plt.
df =pd.read_csv ({file_name})
x=np.nstock ([row_df.vol[:2:]]
y=row_df_volws[1:2:1]
return x_train, x_test, y_test=
```

Regression coefficient

```
print ("coefficients", reg-coeff).
print ("varier score : '{}'.format(reg.
        score(x-test - y-test))}.
```

Dy 28/4

Kaggle certification.

```
import pandas as pd
imported_data = pd.read_csv("file.csv")
print(imported_data):
```

Reading from URL

```
Url = "HHtps://archive.ics.uci.edu/ml/
        machine_learning_data/irs/iris.data"
col_names = ["sepal_dyth_in_cm",
        width_in_cm", "petal_length_in.cm",
    "petal_width_in_cm" ];
iris_data = pd.read_csv(url, names =
                                col.names)
iris.head():
```

Exporting data into csv-file

```
iris.data = to_csv("Cleaned_irs_data
                        csv"):
```

Week: 04

To Explore Decision Tree Algorithm

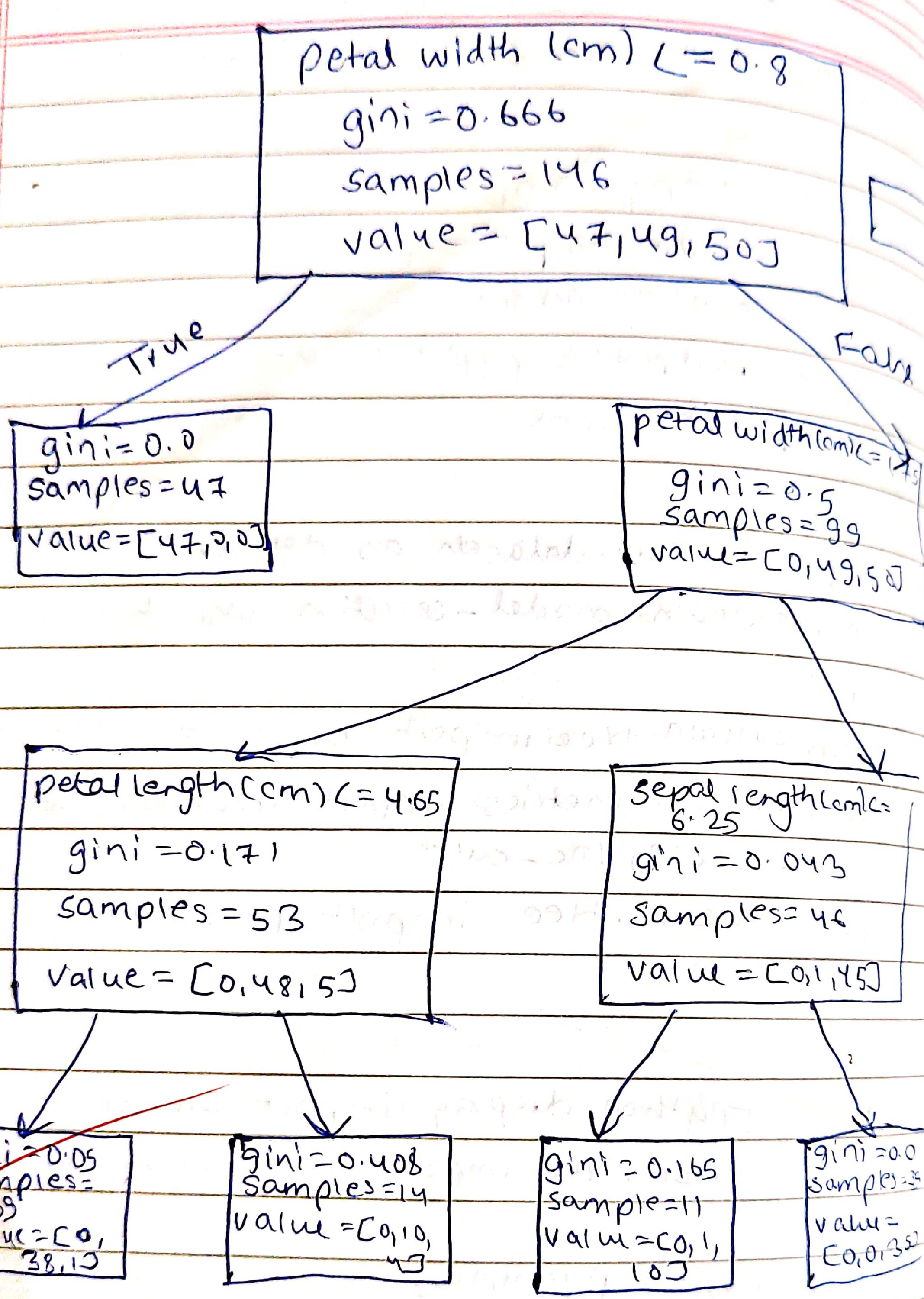
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline


import sklearn.datasets as datasets
from sklearn.model_selection import train_test_
split

from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, roc_
auc_score, roc_curve
from sklearn.tree import plot_tree


from IPython.display import Image
from sklearn.tree import export_graphviz

!pip install pydotplus
import pydotplus.
```


petal width (cm) <= 0.8
gini = 0.666
samples = 146
value = [47, 49, 50]
True
False
gini = 0.0
samples = 47
value = [47, 0, 0]
petal width (cm) <= 1.75
gini = 0.5
samples = 99
value = [0, 49, 50]
petal length (cm) <= 4.65
gini = 0.171
samples = 53
value = [0, 48, 5]
sepal length (cm) <= 6.25
gini = 0.043
samples = 46
value = [0, 1, 45]
gini = 0.05
samples = 39
value = [0, 38, 1]
gini = 0.408
samples = 14
value = [0, 10, 4]
gini = 0.165
sample = 11
value = [0, 1, 10]
gini = 0.0
samples = 35
value = [0, 0, 35]

Accuracy

```
print('Accuracy of Decision Tree-Train:',
  accuracy_score(y_pred_train, y_train))
print('Accuracy of Decision Tree-Test:',
  accuracy_score(y_pred, y_test))
```

Accuracy of Decision Tree-Train: 1.0
Accuracy of Decision Tree-Test: 0.9545454546

```
y_pred = dt_classifier.predict(X_test)

accuracy = accuracy_score(Y_test, Y_pred)
print(f"acuracy: {accuracy}")

new_sample = pd.DataFrame([[5.0, 3.5, 0.2]]
                          columns=X.columns)
predicted_class = dt_classifier.predict
                  (new_sample)
printf(f"Predicted class for new sample:
                  {predict_class[0]}").
```

18/4/24

Week: 05.

```
impolt pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline

from sklearn.metrics import accuracy_score
df = pd.read_csv("/content/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=0.6, random_state=0)

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)

y_predicted
array([1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0])

accuracy = accuracy_score(y_predicted, y_test)
print(accuracy)
    0.81818182

model.intercept_
array[-6.19091564])

import math
def sigmoid(x):
    return 1/(1+math.exp(-x))
age = 35
prediction_function(age)
    0.48500449899
age = 43
prediction_function(age)
0.56857705
```

age = 86

prediction_function(age)

0.8891413248300

10/25/4/24

Week: 06

May 9, 2024.

SVM - Iris Dataset

```
iris=pd.read_csv ('IRIS.csv')
sns.pairplot (data=iris, hue='species', palette='Set2')
from sklearn.model_selection import train_test_split

X=iris.iloc[:, :-1]
y=iris.loc[:, 4]
x_train, x_test, y_train, y_test=train_test_split
(x,y,test_size=0.30)
from sklearn.svm import SVC
model=SVC()
model.fit(x_train, y_train)
SVC()

model2=SVC (C=5, Kernel='rbf')
model2.fit (x_train, y_train)
pred=model2.predict(x_test)
from sklearn.metrics import classification_report,
confusion_matrix
print (confusion_matrix (y_test, pred)
print( classification_report (y_test, pred))
```

KNN

```
classifier = KNeighborsClassifier(n_neighbors=2)
classifier.fit(x_train, y_train)
y_pred = classifier.predict(x_test)

cm = confusion_matrix(y_test, y_pred)
cm

accuracy = accuracy_score(y_test, y_pred)
* 100
print('Accuracy of our model is equal' +
  str(round(accuracy,2)) + '%.')
```

Accuracy of our model = 93.33%.

Cross Validations.

```
k_list = list(range(1,50,2))
cv_scores = [ ]

for k in k_list:
knn = KNeighborsClassifier(n_neighbors=k
scores = cross_val_score(knn, x_train,
y_train, cv=10, scoring='accuracy')
```

```
MSE = [1 - x for x in cv_scores]
plt.figure(1)
```

~~M~~

```
best_k = k_list [MSE.index(min(MSE))]
print(best_k)
```

O/P:

Optimal no. of neighbors = 3

Accuracy = 97.33%

9/5/24

Week: 07

23/5/24

ANN using BP

```
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.max (x, axis=0)
y = y/100
epoch = 5000
lr = 0.1
inputlayer_neurons=2
hidolenlayer_neurons= 3
output_neurons = 1

wh=np. random.uniform(size=(input
layer_neurons, hiddenlayer_neurons))
bh= np.random. uniform (size= (1, hidden
layer_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):
    hinpl= np.dot(x,wh)

hinp =hinp+bh
    hlayer_ait = sigmoid(hinp)
    outinpl= np.dot(hlayer_act,wout)
    outinp = outinpl+bout
    output = sigmoid(outinp)

    hiddengrad= derivatives_sigmoid(hlayer_
        act)
    d_hiddenlayer = Eh* hiddengrad
    print("Input:\n"+ str(x))
    print("Actual output:\n",str(y))
    print("Predicted output\n" output)
```

OUTPUT

Input

[[0.66667 1]
[0.333 0.55556]
[1 0.66667]]

Actual output

[[0.92]
[0.86]
[0.89]

Predicted output

[[0.7841 8718]
[0.765 3993]
[0.78892728]

30/5/24

23/5/24

Week: 08

Random Forest Algorithm

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
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)
```

accuracy = 0.93333

```
classification_rep = classification_report (y_test, y_pred)
print("\n Classification Report\n", classification_rep)
```

OUTPUT

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
macroavg	0.94	0.93	0.93	75
weight avg	0.95	0.93	0.93	75

ADA BOOST Week: 09

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_split
from sklearn.metrics

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

x_train, x_test, y_train, y_test = train_test_split(x,y,size=0.4)
adaboost_clf = AdaBoostClassifier(n_estimators=30, learning_rate=1.0,
random_state=42)
adaboost_clf.fit(x_train y.train)

accuracy=accuracy_score(y.test, y.pred)
print("Accuracy":accuracy)
```

OUTPUT

Accuracy: 0.96666

30/5/24

30/5/24

Build k-means algorithm to cluster a set of data stored in a csv file

```
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']
```

K-means model

```
model = KMeans(n_clusters=3)
model.fit(x)
```

Visualizing

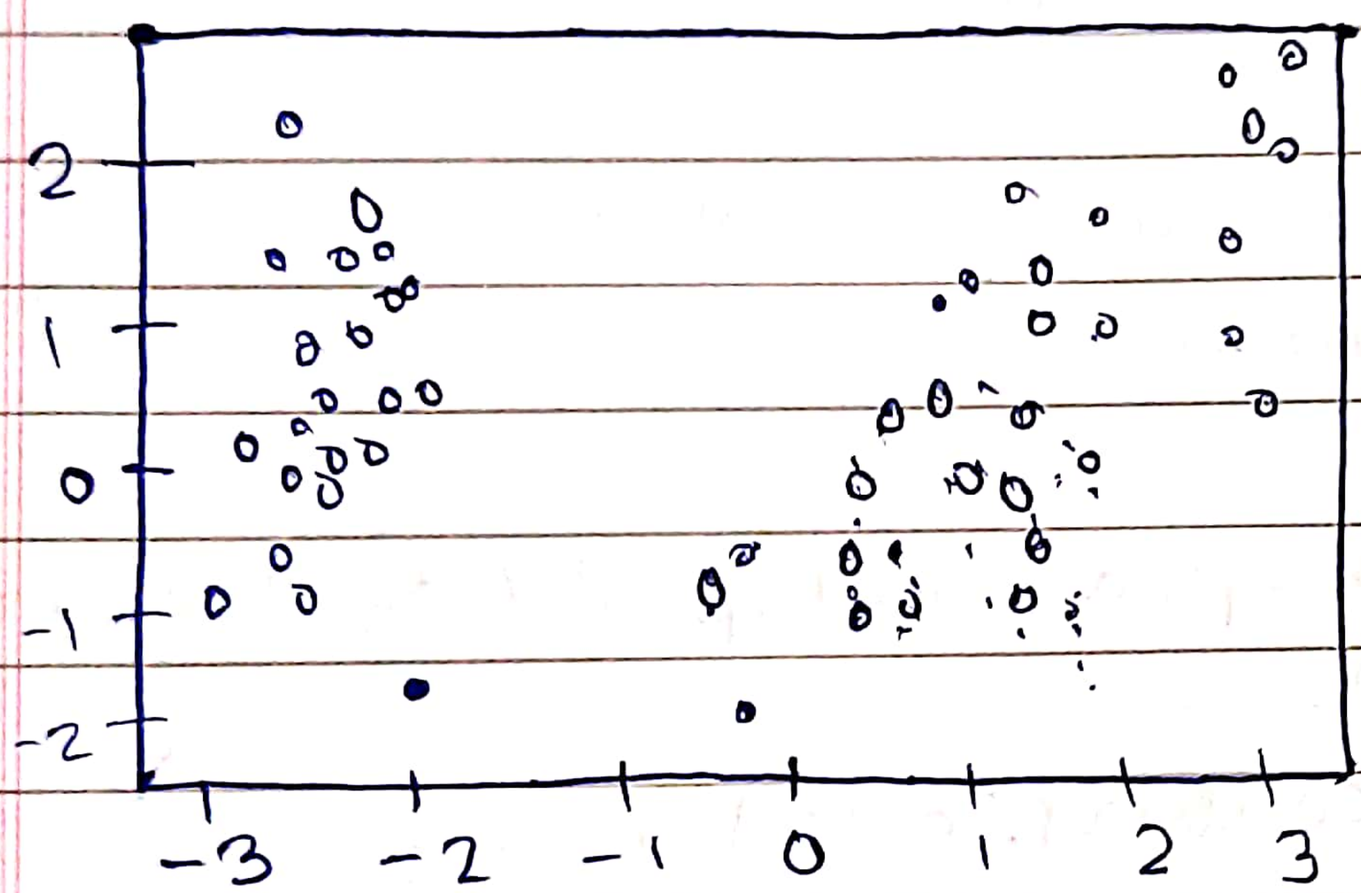
```
plt.figure(figsize=(19,14))
colormap = np.array(['red', 'lime', 'black'])
plt.subplot(2,2,1)
plt.scatter(X.Petal_Length, X.Petal_Width,
        c = colormap[y.Targets], s=40)
```

```
plt.title('Real clusters')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
```

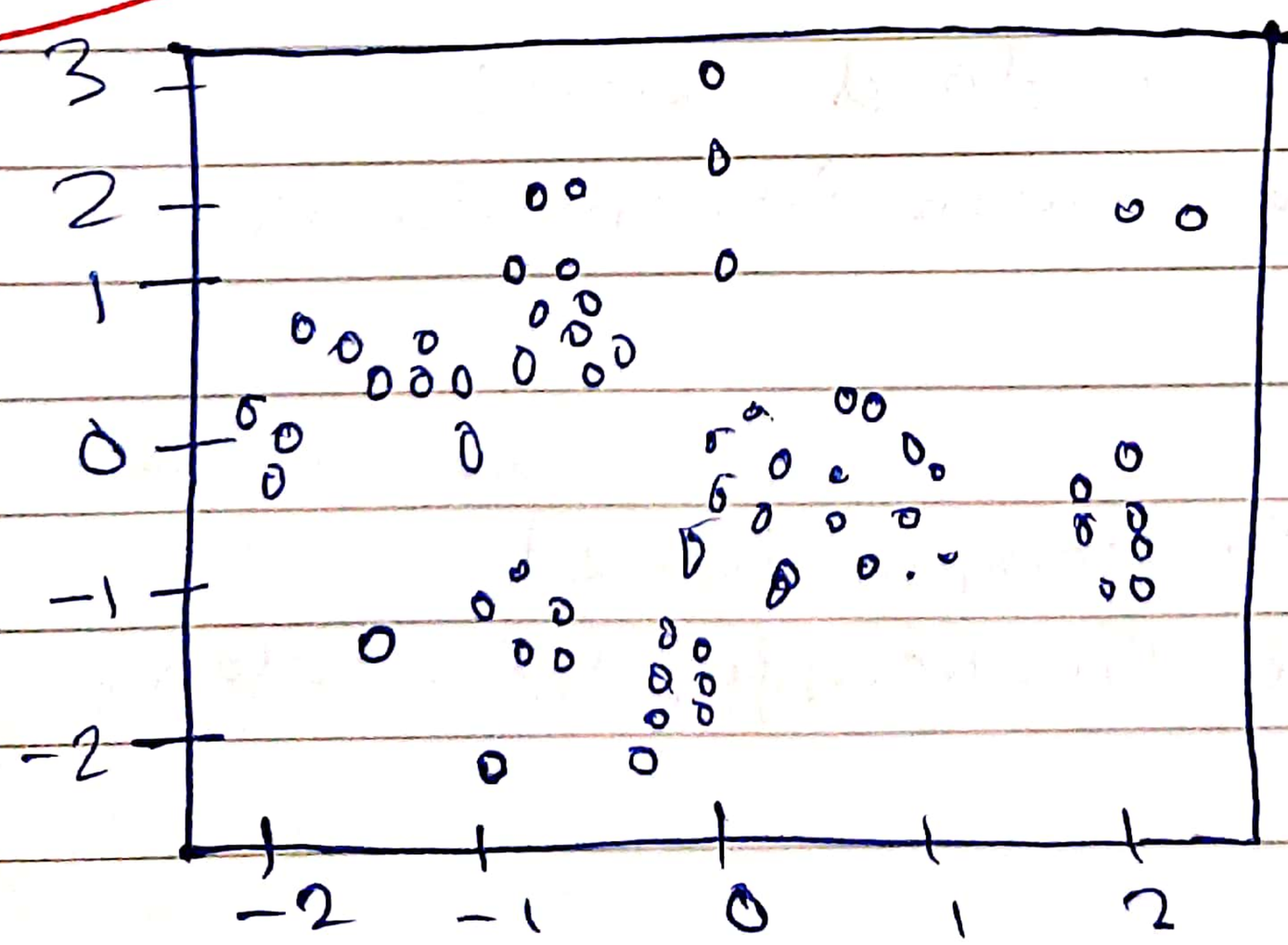
~~plt.subplot(2,2,2)~~

~~plt.scatter(~~

K-means clustering of Iris Dataset



K-means clustering of Iris Dataset



Implement Dimensionality reduction using Principle Component Analysis (PCA) method.

```
import pandas as pd
from sklearn.preprocessing import StandardScalar
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt


if list(df.columns) != columns:
    df.columns = columns

def perform_pca_and_plot(df, feature_cols, title):
    X = df[feature_cols].values
    y = df['species'].values

    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)

Apply PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
```

```
pca_df = pd.DataFrame(data=X_pca,columns
        =['Principal component 1', Principal
         component 2'])
pca_df['species']=y

plt.figure(figsize=(8,6))
colors=['r','g','b']
species_unique=df['species'].unique()
for target, color in zip(species_unique, colors):

plt.xlabel('Principal component 1')
plt.ylabel('Principal component 2')
plt.title(title)
plt.legend(species_unique)
plt.grid()
plt.show()
```

Explained ~~Experienced~~ variance for PCA of Sepal Dimensions: [0.55468462 0.44531538]

~~Experienced~~:
Explained variance for PCA of Petal Dimension: [0.98137855 0.01862145]

30/5/24