INDEX

Name Zayd Ahmed Std Sec 6D

Roll No. Subject ML Lab School/College BMSCE

School/College Tel. No. Parents Tel. No.

Sl. No.	Date	Title	Page No.	Teacher Sign / Remarks
01.	21/3/24	Import/Export using Pandas	10	21/3/24
02	28/3/24	Week 2	9	
03.	4/4/24	Week 3	10	
04.	18/4/24	Week 4	10	25/4/24
05.	25/4/24	Week 5	10	
06.	9/5/24	Week 6	10	9/5/24
07.	9/5/24	Week 7		
08	23/5/24	Week 8	10	23/5/24
09	23/5/24	Week 9		
10	30/5/24	Week 10	10	30/5/24
11	30/5/24	Week 11		

Thursday, 21/3/24

Week 1

Q) Python Program to import and Export data using Pandas library functions

```
import pandas as pd
sdata = pd.read_csv("dataset.csv")
sdata.head()
```

O/P

	Year	levels	code	name	units	variableCode
0	2021	level1	9999	Napco	USD	H01
1	2022	level2	8888	NZSC	INR	H40

—x—

```
url = "https://archive.ics.uci.edu/ml/machine-database/iris.data"
col_names = ["sepal_length_cm", "sepal_width_cm", "petal_length
    _cm", "petal_width_cm", "class"]
iris_data = pd.read_csv(url, names = col_names)
iris_data.head()
```

O/P

	sepal_length_cm	sepal_width_cm	petal_length	petal_width	class
0	5.1	3.5	1.4	0.2	setosa

—x—

21/3/24

Week 2

28/3/24

End-end ML project

1. Get 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()
```

Now download the data

```
fetch_housing_data()
```

2. Discover and visualize the data to gain insights

latitude / longitude info can be represented using .plot()

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

to estimate densities, use 'alph' parameter

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

to check the relations between two or more data info 'corr' method is used

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

This generates a weak correlation b/w the specified parameters

for a better correlation, we need to do

```
corr_matrix = housing.corr()
corr_matrix['median_house_value'].sort_values(ascending=
                                              False)
```

3. Prepare data for ML Algorithms

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy='median')

# this can handle only numbers
housing_num = housing.drop("ocean_proximity", axis=1)

# now we fit imputer over our data
# fitting is basically training
imputer.fit(housing_num)

# now we can use this to transform the numbers by
replacing their missing values with corresponding medians
X = imputer.transform(housing_num)

housing_tr = pd.DataFrame(data=X, index=housing_num.index,
                          columns=housing_num.columns)

housing_tr.head()
```

4. select and train a Model

```
from sklearn.linear_model import LinearRegression
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)
print("predictions:", lin_reg.predict(some_data_prepared))
```

```
from sklearn.metrics import mean_squared_error

housing_predictions = lin_reg.predict(housing_prepared)

lin_mse = mean_squared_error(housing_labels, housing_
                             predictions)
```

~~Week 3~~

~~Linear Regression~~

~~import numpy as np~~

~~pd~~

Week 3

4/9/24

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.linear_model import LinearRegression
```

#Import

```
df_start = pd.read_csv('/content/50_Startups.csv')
df_start.head()
```

analyze

```
df_start.describe()
```

relationship b/w profit & RND spend

```
plt.scatter(df_start['R&D Spend'], df_start['Profit'],
            color='lightcoral')
plt.title('Profit vs R&D Spend')
plt.ylabel('Profit')
```

```
plt.xlabel('R&D spend')
plt.box(False)
plt.show()
```

split into independent / dependent variables

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

one-hot encoding

```
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),
                                      [3])], remainder='passthrough')
X = np.array(ct.fit_transform(X))
```

train Model

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

predict results

```
y_pred = regressor.predict(X_test)
```

compare predictions

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

Sneha
4/4/24

Week 4

18/4/24

import Libraries

```
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
```

Loading 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())
```

classify X & y

```
X = iris_df.drop('target', axis=1)
Y = iris_df['target']
```

split data into testing and training

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

construct tree classifier

```
clf = DecisionTreeClassifier(criterion='gini', random_state=100,
                             max_depth=5, min_sample_leaf=8)
clf.fit(X_train, y_train)
```

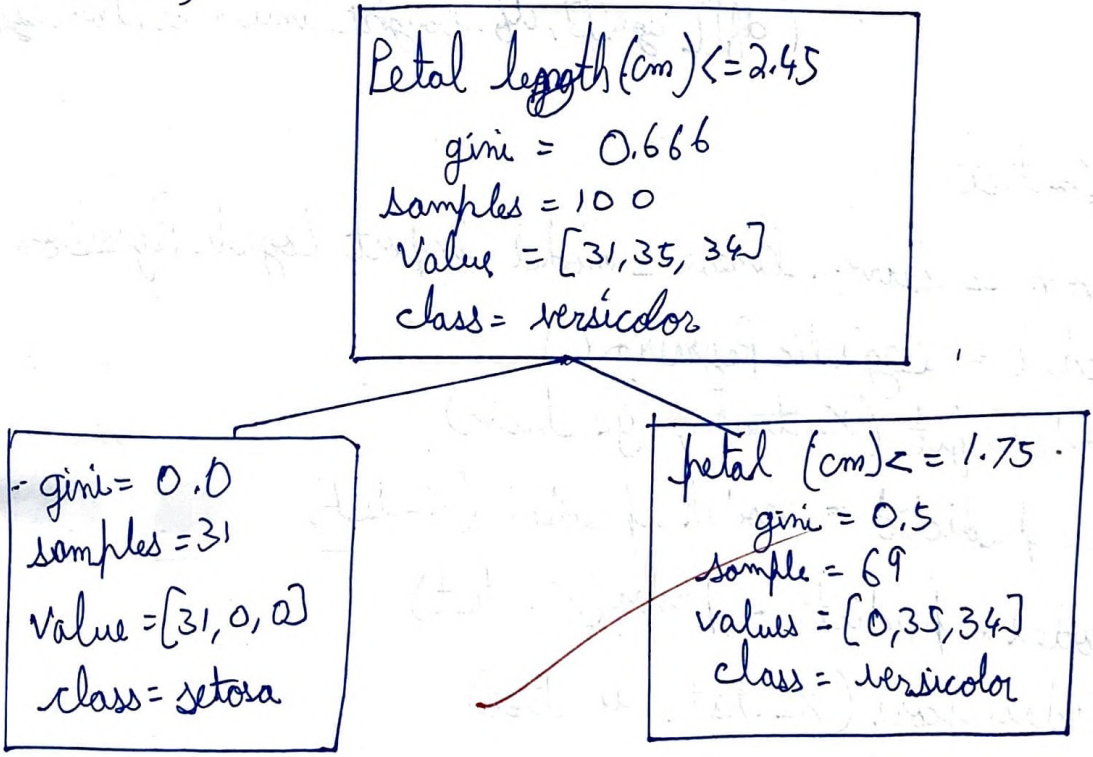
predict accuracy

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

accuracy : 0.98

Plot decision tree

```
plt.figure(fig.size=(12,8))
tree.plot_tree(clf, feature_names=iris_data.feature_names, class_name=
               iris_data.target_names, filled=true)
plt.show()
```



— X —

18/4/24

Week 5

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

df = pd.read_csv("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)

X_test

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)

y_predicted

X_test
```

model.coef_

0.1547

model.intercept_

-6.4737

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

def prediction_function(age):
    z = 0.1547 * age - 6.4737
    y = sigmoid(z)
    return y

age = 35
prediction_function(age)

# 0.25747          // not buying

age = 43
# 0.544            // buying
```

Smita
-25/4/24

Week - 7

SVM

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.datasets import load_iris

iris = load_iris()
data = pd.DataFrame(data = iris.data, column = iris.feature_names)
data['Species'] = iris.target

X = data.drop('species', axis = 1)
Y = data['species']
print(data.head())

plt.scatter(X['sepal length (cm)'], X['sepal width (cm)']
            c=y, cmap='viridis')

plt.xlabel('sepal Length (cm)')
plt.ylabel('sepal width (cm)')
plt.title('Scatter Plot of Iris Dataset')
plt.show()

X_train, X_test, y_train, y_test = train_test
                    split(x, y, test_size = 0.25, random_state=42)

svm_classifier = SVC(kernel='linear', random_state =42)
svm_classifier.fit(X_train, y_train)
y_pred = svm_classifier.predict(X_test)
```

```
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy:', accuracy)
```

Accuracy: 1.0

Week -6

KNN

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifiers
from sklearn.metrics import accuracy_score
import numpy as np

data = pd.read_csv('spam_ham_dataset.csv')
data.head()
data.drop(['label','unnamed:0'], axis1, inplace = True)
data.head()

X = data['text']
y = data['label_num']
vectorizer = Tfid.fvectorizer()
X = vectorizer.fit_transform(X)

X_train, X_test, y_Train, y_test = train_test_split
                    (X,y, test_size=0.3, random_state=42)

k=6
knn_classifier = KNeighborsClassifier(n_neighbors = k)
knn_classifier.fit( X_train, y_train)
y_prod = knn_classifer.predict(X_test)
```

accuracy = accuracy_score (y_test, y_pred)
print (accuracy)

0.935567

—— × ——

9/5/24

Week 8

ANN

```
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))
bh = np.random.uniform(size=(1, hiddenlayer_neurons))
wout = np.random.uniform(size=(hiddenlayer_neurons))
bout = np.random.uniform(size=(1, output_neurons))
def sigmoid(x):
    return 1/1(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)
    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)
```

—x—

O/P

Input

```
[[0.6667  1.    ]
 [0.3333  0.5556]
 [1.      0.6667]]
```

Actual OP

[0.93] [0.86] [0.89]

Predicted OP

```
[0.8956]
[0.8794]
[0.8941]
```

—x—

9.1 Random Forest Algorithm

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

iris_data = load_iris()
iris_df = pd.DataFrame(data = iris_data.data, col = iris_data.
                                                    feature_names)

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 = 0.3, state = 42)

clf = RandomForestClassifiers(n_estimators=50)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

accuracy = metrics.accuracy_score(y_test, y_pred)
print(accuracy)
```

O/P

0.983333

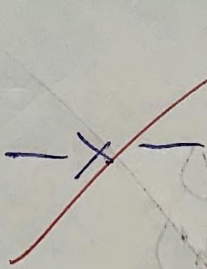
—X—

9.2 AdaBoost

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn import datasets
from sklearn.model_selection import train_test_split

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

X_train, X_test, y_train, y_test = train_test_split
                                   (X, y, test_size=0.25)

abc = AdaBoostClassifier(n_estimators=50, learning_rate=1)
model = abc.fit(X_train, y_train)
y_pred = model.predict(X_test)
print('Accuracy:', metrics.accuracy_score(y_test, y_pred))
```

O/P

0.975623



Sneha
23/5/24

K. means

Week 10

```
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.cluster import Kmeans
import numpy as np

iris = datasets.load_iris()
X = pd.DataFrame(iris.data)
X.columns = ['Sepal_length', 'Sepal_width', 'Petal_length']
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.
            s=40)
plt.title('Real clusters')
plt.xlable('Petal Lengths')
plt.ylable('Petal Widths')
plt.subplot(2, 2, 2)
plt.scatter(X.Petal_length, X.Petal_Width, c=colors,
            s=40)
plt.title('K-means Clustering')
plt.xlabel('Petal Length')
plt.ylabel('Petal width')
```

PCA Week 10

```
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)
StandardScaler(copy = True, with_mean = True, with_std = True)
scaled_data = scaler.transform(df)
from sklearn.decomposition import PCA
pca = PCA(n_components = 2)
pca.fit(scaled_data)
PCA(copy = True, n_components = 2, whiten = False)
X_pca = pca.transform(scaled_data)
scaled_data.shape
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