INDEX

Name: Shashank. M. M   Std   Sec

Roll No.   Subject: ML lab   School/College

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

Sl. No.	Date	Title	Page No.	Teacher Sign / Remarks
01	21/3/24	Iris. Dataset (1st lab program)	10	21/3/24
02.	28/3/24	Week 2	10	28/3/24
03.	04-04-24	Week-03.	10	4/4/24
04	18-04-24	Week-04	10	18-4-24
05	25-04-24	Week-05	10	9-5-24
06	09-05-24	Week-06.	10	9-5-24
07.	09-05-24	Week-07	10	9-5-24
08.	23-05-24	8, 9a, 9b	10	23-5-24
09	30-05-24	Lab. 10, 11	10	30/5/24

21/03/24

Lab 1.

Iris data set

1. import pandas as pd

url = " https://archive.ics.uci.edu/ml/machine-learning-database/iris/iris.data "

col names = [" Segal-length-in-cm",
" segal-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 length in cm	petal length in cm
5.1	~~3.5~~	3.5 1.4

petal width in cm	class
0.2	ires-setosa

iris_data.to_csv ("cleaned_iris_data...

Data set is done in CSV

① Fetch

-> To load

dif

pat

os.

retu

-> housi

Nam

-> housi

Sho

-> %

impo

housi

plt.

-> (

Week-02

① Fetch the Data of housing:

→ To load the data:

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

→ housing["Ocean_proximity"].value_counts()

Name: ocean_proximity, dtype: int64

→ housing.describe()

Shows all the output

→
```
%matplotlib inline
import matplotlib.pyplot as plt
housing.hist(bins=50, figsize=(20,15))
plt.show.
```

Output → histogram for each numerical attribute.

→ Create a test case:

```
housing["income_cat"] = pd.cut(housing
                ["medium_income"],
        bins = [0, 1.5, 3.0, 4.5, 6., np.inf],
        labels[1, 2, 3, 4, 5])
```

→ histogram of income categories.

→ strat_test_set["income_cat"].value_counts() / len(strat_test_set).

⇒ income_cat, dtype: float64

→ To create a copy of main dataset

housing = strat_train_set.copy()

→ Visualizing Geographical Data.

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

Correlations.

corr_matrix = housing.corr().

→ corr_matrix["median_house_value"].sort_values(ascending = False)

Name: median_house_value, dtype: float64

Scatter Matrix

housing.plot(kind = "scatter", x = "median_income", y = median_housing_value", alpha = 0.1)

→ Median income versus median house value.

Analyze the Best Models and their Errors

→ feature_importance = grid_search.best_estimator_.feature_importances

. feature_importance

array([7.33, 6.29-02 ... 2.8564-03])

4/4/24

Week-03

part-02: Linear and Multiple Regression.

import\ mv

→ Salary file

① Import pandas as pd.
d = pd.read_read_csv ('salary.csv')
df.head()

② import librarries.

③ df_sal_describe()

④ plt.title ('Salary Distrubution Plot')
sns.distplot (df_sal['Salary'])
plt.show()

→ plot of Salary Distrubtion Plot v/s Salary.

⑤ plt.scatter (df_sal['Years Experience'], df_sal['Salary'], colour = 'light corol')
plt.title ("Salary VS Experience")
plt.xlable('Year of Experience')
plt.ylable ('Salary')
plt.box (False)
plt.show()

→ Sal

⑥ x = df_sal
y = df_sal

⑦ x_train, x_
split (x,

⑧ regressor =
regressor.(

//

Start w

① Impor

② Impor

③ df-

④ plt.
sns.
plt.

⑤ plot

⑥

⑦

⑧

⑨

(6) x = df_sal. iloc [:, :1]
y = df_sal. iloc [:, 1:]

(7) x train, x_test, y train, y_test = train_test_split (x, y, test_size = 0.2, random_state = 0)

(8) regressor = Linear Regression()
regressor.fit (x_train, y_train).

Start up file

(1) Import file.

(2) Import libraries.

(3) df_start. describe()

(4) plt. title ('profit Distrubution Plot')
sns. distplot(df_start ['profit'])
plt. show()

(5) plot Profit VS R and D spend graph.

(6) x = df_start. iloc [:, :-1], values.
y = df_start .iloc [:, :-1]. values

(7) x_train, X_test, y_train, y_test = train_test_split (x, y, test_size = 0.2, random_state = 0)

(8) regressor = Linear Regression()
regressor.fit (x_train, y train)
regressor.predict (x_test)

(9) y_pred = regressor.predict (x_test)

using Dataset

sing and

inear

nance.

Week 4. Decision Tree

→ import pandas as pd
import mathplotlib.pyplot as plt
from sklearn.dataset import load_iris
import seaborn as sns.

-from sklearn.datasets import load_iris.
iris_data = load_iris()
print (iris_data.DESCR)
X = iris_data.data.
Y = iris_data.target.
print ("Shape of X:", X.shape)
print ("Shape of Y:", Y.shape)

from sklearn.model_selection import train_test_split.
X_train, X_test, Y_train, Y_test = train_test_split (X, Y, test_size = 0.33, random_state = 42)

from sklearn.tree import Decision Tree Classifier.
treemodel = Decission Tree Classifier()
treemodel.fit (X_train, Y_train)
Decision Tree Classifier()

from sklearn.tree import Decision Tree Classifier, plot tree
import mathplot. pyplot as plt.

- Importing Libraries
- Loading the Boston Housing Dataset
- Preparing Data
- Splitting Data into training and Testing Sets.
- Creating and Training the Linear Regression Model.
- Evaluating Model performance.
- Plotting the error.

Wee

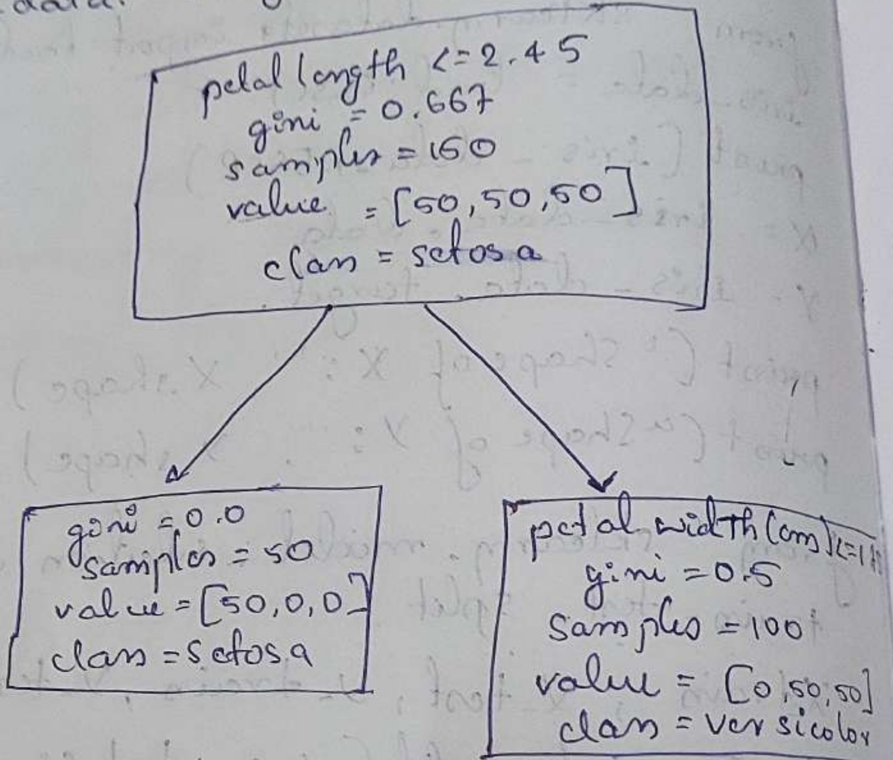
→ import p
import
from sk
import s

- from
iris_da
print (
X = ir
y = iri
print
print

from
trai
X_tr
fr
ran

fro
Cla
f
tre
D
f
w

```
clf = DecisionTreeClassifier()
clf.fit(x,y)
plt.figure(figsize=(12,8))
plot_tree(clf, filled=True, feature_names=
    iris_data.feature_names, class_names=
    iris_data.target_names)
```




```
import   from sklearn.model_selection
            import cross_val_score
         scores = cross_val_score(clf, x, y,
                                  cv=5)

         accuracy = scores.mean()
         print("Mean Accuracy", accuracy)
```

10/4

$\boxed{\text{Mean Accuracy} = 0.96000}$

Week - 05 25-09-24

① import pandas as pd.
from mathlib import pyplot as plt.
% matplotlib inline

② read the csv file.

③ plt. scatter (df.age, df. bought.
insurance, marker = '+': color 'red')
from sklearn.model - selection import
train- test - split.
x train , x- test y-train = train - test
split (df [['age']], df. bought -
insurance ,train - size = 0.8)

print (x-test)
from sklearn . learn - model
import logistic regression
model = logistic Regression ()
model .. fit (x-train . y-train)

~~print (x- test)~~

y predicted = model. predict (x- test)
model. predict . proba (x-test)
model. score (x test, y-test)

```
print (y predicted )
print ( x. test )
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 )
```

y :- 0. 48504498380589

age : 43 .

```
prediction function (age )
```

y :- 0.56856524977 05

Week 6 KNN 09-5-24

sklearn KNN classifiers

(1) importing libraries

(2) printing Insurance Dataset

(3) plt. scatter (df.iloc [:, 0], df.iloc [:1],
c= df.i loc [:, -1], cmap= 'viridis')
plt. xlable ('feature')
plt. ylable ('feature 2')
plt. title (' scatterplot of the dataset')
plt.show().

(4) X = df. iloc [:, :-1]
y = df. iloc [:, -1]

(5) X_train, X_test, y_train, y_test =
train_test_split (X, y, test size = 0.3,
random_stat = 42)

(6) Knn = KNeighborsClassifier (n_neighbors =3)

(7) knn. fit (X_train, y_train)
=> KNeighborsClassifier(n_neighbours=3)

(8) y_pred = knn. predict (X_test)

(9) accuracy = accuracy_score (y_test, y_pred)
print ("Accuracy :", accuracy)

Accuracy = 0.888888858

Week7: SVM. sklearn.svm

① import libraries.

② plt scatter (df[dF['target']==0]['sepal length (cm)'], df[df['target']==0]
['sepal width (cm)'], label = 'setosa')

df['target']==1, df target == 2

label = 'Versicolor',

lable = 'Virginica'

plt. xlable ('sepal length (cm)')

plt. ylable ('sepal width (cm)')

plt. title ('sepal Width vs length')

plt. legend()

plt. show()

X_train, X_test, y_train, y_test =

train_test_split(iris_data, iris_target,

test_size = 0.2, random_state = 42)

svm_classifier = SVC(kernal = 'linear')

svm_classifier.fit (X_train, y_train)

SVC(kernal = 'linear')

svm

]==0]['sepal
['target']==0]
label='setosa')
arget==2
'color'
nica'
gth (cm)')
(cm)')
s length')

nt =
, iris.target,
_state=42)
l='linear')
y_train)
)

```
y_pred = svm_classifier.predict(x_test)
accuracy = accuracy(y_test, y_pred)
print("Accuracy:", accuracy)
```

Accuracy = 1.0.

Week 8

ANN using Backpropagation

```
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,
        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)

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

Input:

[[0.66667 1.0.33333 0.55556

1. 0.666667]]

AO:

[[92.] [86.] [89.]]

PO: [[0.8428407]

[0.8345443]

[0.841814]]

13-5-24

Week 9a

Random Forest Algorithm.

```
Import pandas as pd.
from sklearn.model selection import
                train_test split.
from sklearn.ensemble import
                Random Forest Classifier.

iris = load_iris()
data = pd.DataFrame(data = iris.data,
       column = Iris_feature_name)
data['target'] = iris.target.
X = data.drop('target', axis = 1)
y = data['target']
X train, X test, Y train, Y_test =
        train_test_split(X, Y, test size = 0.2,
                random state = 42)
rf classifier = Random Forest Classifier()
rf classifier fit (X_train, y train)
y_pred = rf classifier predict(X_test)
accuray = accuray_score(Y_test, Y_pred)
print ("Accuray", accuray)
```

Accurray = 1.0

Week a6. ADA boost.

import libraries.

import sklearn.ensemble import
AdaBoostClassifier.

df = read_csv('iris.csv')

df.head()

x = df.drop('species', axis = 1)

y = df ['species'].

x_train, x_test, y_train, y_test =
train_test_split (x, y, testsize=0.1
random_state=56).

x_train.shape, x_test.shape,

y_train.shape, y.test.shape

((135,5), (15,5), (135,1), (15,1))

mylogregmodel = LogisticRegression()

adbar = AdaBoostClassifier(n_estimators= 100,
base_estimators = mylogregmodel,
learning rate = 1)

model = adabc.fit (x_train, y_train)

y_pred = model.predict (x_test)

print ('Accuracy score', accuracy score(y_test,
y_pred)

Acc = 1.0

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.xlable('Petal Length')
plt.ylable('Petal Width)
plt.subplot(2, 2, 2)
plt.scatter(X.petal_length, X.petal_Width,
    c = colormap[model.labels_], s=40)
```

plt.title
plt. xlable
plt. ylable

```
plt.title('K-Means clustering')
plt.xlable('Petal Length')
plt.ylable('Petal width')
```

11. Princip... Analysis

```
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 .key()
dict_keys ([ 'DESCR', 'data',
     'feature_names', 'target_names',
   target']])
print (cancer['DESCR'])
df = pd. Dataframe (cancer [ 'data'],
     coloumns = cancer ['feature_names'])
df.head ()
from sklearn.preprocessing import
    standard scalar.
scalar = Standard Scalar()
scalar. fit (df).
Standard Scalar (copy = True, width_mean
    True, width_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

(569, 30)

x_pca.shape

(569, 12)

plt.figure(figsize = (8, 6))
plt.scatter(x_pca[:, 0], x_pca[:, 1],
        c= cancer['target'], cmap='plasma')

plt.xlable('First principal component')
plt.ylable('Second principal component')
```