INDEX

Name spoorthi. J

Standard Section Roll No.

Subject MachineLearning LAB.

School / College

S. No.	Date	Title	Page No.	Teacher's Sign
1	21/3/24	Import and Export data -		
		Pandas Library		4/4/24
2	28/3/24	End-to-End Project		18/4/24
3	4/4/24	Linear Regression.		18/4/24
4	18/4/24	Decision Tree		25/4
5	25/4/24	Logistic Regression		
6	9/05/24	KNN,		
7	9/05/24	SVM		9/5/24
8	23/5/24	Random Forest (9a)		
		AdaBoost (9b)		
9	23/5/24	ANN (8)		23/5/24
10	30/5/24	K-Means Clustering (10)		
11	30/5/24	Principle Component (11)		30/5/24
		Analysis.		

LAB-01 21/3/24.

① Write a python program to import and export data using pandas library function.

```
import pandas as pd
airbnb_data = pd.read_csv("data/listings_austin.csv")
airbnb_data.head()
```

Read 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()
```

Exporting dataframe to csv file

```
iris_data.to_csv("cleaned_iris_data.csv")
```

Output:

	sepal_length_in_cm	sepal_width_in_cm	petal_length_in_cm	petal_width_in
0	5.1	3.5	1.4	0.2

class
Iris-setosa.

21/3/24

28/08/24 WEEK - 2

Step 1: Performance Measure

Step 2: Get the data

Download the data

```
import os
import tarfile
import urllib
DOWNLODA_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):
    """ Create 'HOUSING-PATH', Downloads & Extracts the
    contents of 'HOUSING-URL' into '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()
```

Create a Test set

```
import numpy as np
def split_train_test(data, test_ratio=0.2):
    """ splits a dataset into train/test using a 'test_ratio'
    """
```

shuffled_indices = np.random.permutation(len(data))
test_set_size = int(len(data) * test_ratio)
test_indices = shuffled_indices[: test_set_size]
train_indices = shuffled_indices[test_set_size :]
return data.iloc[train_indices], data.iloc[test_indices]

train_set, test_set = split_train_test(data=housing)
len(train_set), len(test_set)

3. Discover & Visualize the Data to Gain Insights.

~~strat_test_set['income_cat'].value_counts()/len(strat_test~~_set).

3.1
start_train_set.shape, start_test_set.shape
start_test_set.reset_index().to_feather(fname='data/oi/strat_test_set.f')
housing = start_train_set.copy(); housing.shape
~~housing.plot~~

3.2
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()

3.3
Looking for correlations

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

Experimenting with Attribute Combinations

```
housing['rooms_per_household'] = housing['total_rooms']/
                                   housing['households']

housing['bedrooms_per_room'] = housing['total_bedrooms']
                                 /housing['total_rooms']
```

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

28/3/24

① Prepare the data for Machine Learning

Datacleaning

```
imputer = SimpleImputer(strategy='median')
housing_num = housing.drop("ocean_proximity", axis=1)
imputer.fit(housing_num).
```

Handling Text and Categorical Attributes

```
housing_cat = housing[['ocean_proximity']]
housing_cat.head(10)
housing_cat['ocean_proximity'].value_counts()
```

Custom Transformers

```
class combined AttributesAdder(BaseEstimator, TransformerMixin):
    def __init__(self, add_bedrooms_per_room = True):
        self.add_bedrooms_per_room = add_bedrooms_per_room

    def fit(self, x, y=None):
        return self

    def transformer(self, x, y=None):
        rooms_per_household = x[:, rooms_ix] / x[:, households_ix]
        population_per_household = x[:, population_ix] / x[:,
                                     households_ix]
```

```
if self.add_bedrooms_per_room:
        bedrooms_per_room = x[:, bedrooms_ix]/x[:, rooms_ix]
        return np.c_[x, rooms_per_household, population_per_
        household, bedrooms_per_room]
else:
    return np.c_[x, rooms_per_household, population_per_household]
```

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
```

(5) Select and Train a Model

```
def display_scores(scores):
        print("Scores:", scores)
        print("Mean:", scores.mean())
        print("Standard Deviation:", scores.std())
```

(6) Fine Tune 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]}
```

Evaluate your system on the test set

```
final model = grid_search.best_estimator_
x_test = start_test_set.drop(labels='median_house_value', axis=1)
y_test = start_test_set['median_house_value'].copy()

x_test_prepared = full_pipeline.transform(x=x_test)

final_predictions = final_model.predict(x=x_test_prepared)

final_mse = mean_squared_error(y_true=y_test, y_pred=final_predictions)
final_rmse = np.sqrt(final_mse)
final_rmse
```

WEEK-03

4/4/24

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

df_sal = pd.read_csv('/content/drive/MyDrive/salary_Data - SalaryData.csv')
df_sal.head()

plt.title('salary Distribution Plot')
sns.distfplot(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()

# splitting variables
x = df_sal.iloc[:, :1]
y = df_sal.iloc[:, 1:]

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_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.ylabel('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 Experience (Test set)')
plt.xlabel('Years of Experience')
plt.ylabel('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_}')
```

Output

Coefficient: [[9312.57512673]]
Intercept: [26780.09915063]

18/4

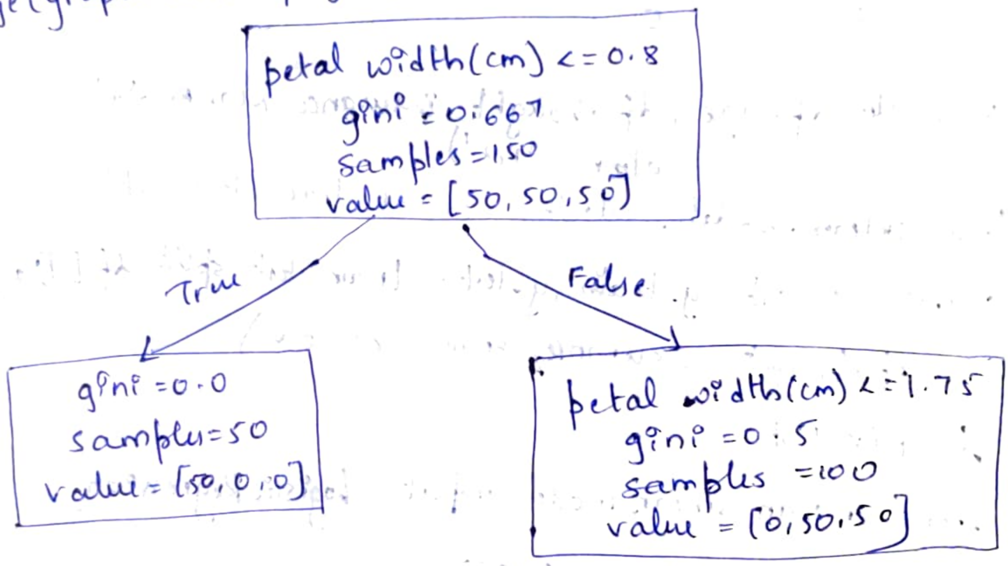
18/4/24

LAB-04

DECISION TREE

```
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.mode_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 sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

url = "https://archive.ics.uci.edu/ml/machine-learning-datasets/
        iris/iris.data"
df = pd.read_csv(url, header = None, name = ['sepal length (cm)',
    'sepal width (cm)', 'petal_length (cm)', 'petal width (cm)',
    'Species'])

df.head()

df.info()

x = df.drop("Species", axis=1)
y = df["species"]
x_train, x_test, y_train, y_test = train_test_split(x, y,
        test_size=0.3, random_state=1)

dt = DecisionTreeClassifier(max_depth=3, min_samples_leaf=10,
        random_state=1)

dt.fit(x, y)
```

```
from Ipython.display import Image
from sklearn.tree import export_graphviz
! pip install pydotplus
import pydotplus
features = X.columns
dot_dat = export_graphviz(dt, out_file=None, featurename=
            features)
graph = pydotplus.graph_from_ddta_(dot_data)
Image(graph.create_png())
```




```
dt = DecisionTreeClassifier(random_state=1)
dt.fit(x_train, y_train)
y_pred_train = dt.predict(x_train)
y_pred = dt.predict(x_test)
y_prob = dt.predict_proba(x_test)

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 : 0.955

18/4/24

25/4/24

WEEK - 05

```
import pandas as pd
df = pd.read_csv('/content/drive/MyDrive/insurance_data.csv')
from matplotlib import pyplot as plt
%matplotlib line
df.head()
```

	age	bought_insurance
0	22	0
1	25	0

```
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)

print(X_test)
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train; y_train)

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

model.coef_
model.intercept_

from sklearn.linear_model. import LinearRegression
model = LinearRegression()
model.fit(X_train, y_train)
y_predicted = model.predict(X_test)
Model.score(X_test, y_test)
```

```
import math
def sigmoid(x):
    return 1/(1+math.exp(-x))
def predicted_function(age):
        z = 0.042 *age - 1.53   # 0.04150733 ~ 0.042 and
                                  -1.52726963 ~ -1.53

    y = sigmoid(z)
    return y

age = 35
prediction_function(age)
age = 43
prediction_function(age)
```

Output =

~~Prediction = array([1, 0, 1, 0, 0, 0, 0, 1, 0])~~

~~score = 0.8333~~

Linear Reg Score : 0.584321

Predictions = 0.485
0.568

25/4/24

Week-06 9/05/24

KNN

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

df = make_blobs(n_samples=500, n_features = 2, centers = 4,
        cluster_std = 1.5, random_state = 4)

~~plt.style.use('seab~~
X_train, X_test, y_train, y_test = train_test_split(X, y,
        random_state = 0)

knn5 = KNeighborsClassifier(n_neighbors = 5)
knn1 = KNeighborsClassifier(n_neighbors = 1)
y_pred_5 = knn5.predict(X_test)
y_pred_1 = knn1.predict(X_test)
from sklearn.metrics import accuracy_score
print("Accuracy with k=5", accuracy_score(y_test, y_pred_5)*100)
print("Accuracy with k=1", accuracy_score(y_test, y_pred_1)*100)
```

Accuracy with k=5 . 93.60
Accuracy with k=1 90.4

SVM

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

iris = pd.read_csv('/content/drive/MyDrive/Iris.csv')
iris.head()

from sklearn.model_selection import train_test_split
x = iris.iloc[:, :-1]
y = iris.iloc[:, 5]
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()

pred = model.predict(x_test)
from sklearn.metrics import classification_report, confusion_matrix

print(confusion_matrix(y_test, pred))

[[18  0  0
   0 15  0
   0  1 11]]

print(classification_report(y_test, pred))
```

	precision	recall	f1-score	suport
Iris-setosa	1.00	1.00	1.00	18
Iris-versicolor	0.94	1.00	0.97	15
Iris-virginica	1.00	0.92	0.96	12
accuracy			0.98	45
macro avg	0.98	0.97	0.97	45
weighted avg	0.98	0.98	0.98	45

9/5/24

Week-06 23/5/24

Random Forest And Adaboost

```
import pandas as pd
import numpy as np
data = pd.read_csv('/content/drive/MyDrive/food-ingredients_
                   and_allergens.csv')

data.head()
y = df['species']
x = df.drop(["species"], axis=1)
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y,
                   test_size=0.3, random_state=0)

from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(n_estimators=100)
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
from sklearn.metrics import accuracy_score
score = accuracy_score[y_pred, y_test]
print(f"Accuracy: {score}")
```

Output:

```
Accuracy: 1.0
```

AdaBoost With Default Parameters

```
from sklearn.ensemble import AdaBoostClassfier
adb = AdaBoostClassifier()
adb=model = adb.fit (x-train, y-train)
y-pred = adb-model.predict (x-test)
score = accuracy-score(y-pred, y-test)
print (f"Accuracy: {score}")
```

Accuracy = 0.977

AdaBoost (with Hyper Parameter)

```
from sklearn.linear-model import Losistic Regression
b-model = logistic Regression ()
adbhp= AdaBoostClassifier (n-estimators =150, estimator = bmodel,
                            learning-rate =1)
model = ~~adb.fit~~ adbhp.fit (x-train, y-train)
y-pred = model.predict (x-test)
score = accuracy - score (y-pred, y-test)
print (f" Accuracy : {score}")
```

Output :-

Accuracy : 1.0

Implementation of ANN using Back Propogation for given values.

```
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.array(x, axis=0)
y = y/100
~~epot~~ epoch = 5000
lr = 0.1
input_layer_neurons = 2
hiddenlayer_neurons = 3
output_neurons = 1

wh = np.random.uniform(size = (inputlayer_neurons,
                        ~~hig~~ 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)
    hink = hink + bh
    hlayer_act = sigmoid(hink)
    outinp1 = np.dot(player_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_hidden_layer = EH * hiddengrad

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

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

<u>Output</u>

Input:

[[0.6667 1
0.3334 0.556
1. 0.6667]]

Actual ~~Length~~ output: [[0.92]
[0.86]
[0.89]]

Predicted Output:

[[0.9 35]
[0.9 23]
[0.9 339]]

23/5/24

K-Means Clustering

Week-07 30/5/24

```
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.fix(x)
plt.figure(figsize=(14,4))
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.xlablel('Petal Length')
plt.ylabel('Petal Width')
plt.subplot(2,2,2)
plt.scatter(x.PetalLength, x.Petal_Width,
            c=colormap[model.labels_], s=40)
plt.title('K-Means Clustering')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
```

Principle Component Analysis

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

from sklearn.datasets import load_breast_cancer
cancer= load_breash_cancer()

cancer.keys()
print(cancer['DESCR'])

df =pd.DataFrame (cancer ['data'], columns= cancer['feature_
                  names'])
df.head()


from sklearn.preprocessing import Standard scaler
Scaler =standard Scaler();
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
x_pca.shape

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

30/5/24