ML LAB.

INDEX

NAME: Sheeram STD ____ SEC: 6D ROLL NO: 1602487

S.No.	Date	Title	Page No.	Teacher's Sign / Remarks
1.	21-3	Import & export data using Pandas library functions		25/4
2.	28-03	Demonstrate various data preprocessing techniques of dataset		25/4
3.	4-4	Implement linear and Multilinear regression algo using appropriate dataset		25/4
4.	18-4	Use an appropriate data set for building the decision tree (ID3) and apply this knowledge to classify a new		25/4

Lab-1 - Week-1

1. Write a python program to import and export data using Pandas library functions.

→ import pandas as pd

```
airbnb_data = pd.read_csv("listing-austin.csv")
airbnb_data.head()
```

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

Output:

①

	id	name	host_id	host_name	neighbourhood	price
0	2265	Zen-East-	2466	Paddy	78702	179
1	5245	Eco-friendly	2466	Paddy	78702	114

②

	sepal_length_in_cm	sepal_width	petal_length	petal_width	class
0	5.1	3.5	1.4	0.2	iris-setosa
1	4.9	3.0	1.4	0.2	iris-setosa

week 9

2. Demonstrate various data pre-processing techniques for a given dataset.

→ Get the data:

Download the data:

```
import os
import tarfile
import urllib
```

```
Download_ROOT = "https://raw.githubusercontent.com/ageron/handson-ml2/master/"
HOUSING_PATH = os.path.join("datasets", "housing")
HOUSING_URL = DOWNLOAD_ROOT + "datasets/housing/housing.tgz"
def fetch_housing_data(housing_url, housing_path):
    os.makedirs(housing_path, exist_ok=True)
    tgz_path = os.path.join(housing_path, "housing.tgz")
    urllib.request.urlretrieve(housing_url, tgz_path)
    housing_tgz = tarfile.open(tgz_path)
    housing_tgz.extractall(path=housing_path)
    housing_tgz.close()
fetch_housing_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 = load_housing_data()
housing.head()
housing.info()
```

3> Discover and visualize the Data to gain Insights.

```
start_train_set.shape, strat_test_set.shape
strat_test_set.reset_index().to_feather("data/o1/strat_test_set.f")
housing = strat_train_set.copy(); housing.shape
```

* Now for visualizing geographical Data

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

Now for correlation

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

```
from pandas.plotting import scatter_matrix
attributes = ["median_house_value", "median_
              income"]
scatter_matrix(frame=housing[attributes], figsize=(12,8))
plt.show()
housing["rooms_per_household"] = housing["totalrooms"]/
                                 housing["households"]
```

4.) Prepare Data for Machine learning Algorithms

```
from sklearn.input import simple.Imputer
    imputer = simple Imputer(strategy="median")
housing_num = housing.drop("ocean_proximity", axis=1)
imputer.fit(housing_num)
imputer.statistics_ housing_num.median().values
```

```
from sklearn.preprocessing import ordinal Encoder
    ordinal_encoder = ordinal Encoder()
    ordinal_encoder categories.
attr_addr = combined Attribut Addr(add_bedroom),
            per_room = False)
from sklearn.compose import column transformation
num_attributes = housing_num.columns.tolist()
```

5) Select and train a model

```
from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()
lin_reg.fit(X=housing_prepared,
            y=housing_labels)

housing_predictions = lin_reg.predict
                      (housing_prepared)

lin_mse = mean_squared_error(housing_labels,
                             housing_predictions)

lin_rmse = np.sqrt(lin_mse)
lin_rmse

tree_reg = DecisionTreeRegressor()
tree_reg.fit(X=housing_prepared, y=housing_labels)
forest_reg = RandomForestRegressor()
forest_reg.fit(X=housing_prepared,
               y=housing_labels)
```

6) Fine Tune the models

```
grid_search.best_params
grid_search.best_estimator.cat_encoder =
        full_pipeline.named_transformers["cat"]
final_model = grid_search.best_estimator
X_test_prepared = full_pipeline.transform
                  (X=X_test)
final_rmse = np.sqrt(final_mse)
final_rmse
squared_errors = (y_test - final_predictions) ** 2
```

LAB-3

Simple Linear Regression and 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 pandas.core.common import random_state
from sklearn.linear_model import LinearRegression

df_sal = pd.read_csv(r"C:\salary_Data.csv")
df_sal.head()
df_sal.describe()

plt.title("salary Distribution Plot")
sns.distplot(df_sal["salary"])
plt.show()
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)]

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

y_pred_test = regressor.predict(X_test)
y_pred_test = regressor.predict(X_train)
```

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

df_start = pd.read_csv(r'C:\50_Startups.csv')
df_start.head()
df_start.describe()

sns.distplot(df_start['profit'])
plt.show()

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)

result.
```

25-4-23

Week-4

1. Use an appropriate data set for building the decision tree (ID3) and apply this knowledge to classify a new sample.

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
import seaborn as sns

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=0.37,
                                    random_state=42)

from sklearn.tree import DecisionTreeClassifier
treemodel = DecisionTreeClassifier()
treemodel.fit(X_train, y_train)
DecisionTreeClassifier().
```

```
from sklearn.model_selection import cross_val_
                                    score

scores = cross_val_score(clf, x, y, cv=5)
~~accuracy~~ =
accuracy = scores.mean()
print("Mean ~~Accuracy~~ Accuracy:", accuracy)
```

Output:-

Mean Accuracy: 0.96666

25-4-24

Lab 5

→ Build Logistic Regression model for a given dataset.

```
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

df = pd.read_csv("insurance_data.csv")
print(df.head(1))

X_train, X_test, y_train, y_test = Train_test_split(df[["age"]], df.bought_insurance,
                                   test_size=0.2)

print("X_test:")
print(X_Test)

model = LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print(y_pred)
print(model.predict_proba(X_test))
print(model.score(X_test, y_test))
```

Output:-

Accuracy = 0.5

```
import math
def sigmoid(z)
    return 1/(1+Math.exp(-z))

def predi(age):
    z = 0.042 * age - 1.[illegible]
    y = sigmoid(z)
```

```
print(predi(35))
print( pred (43))
```

Output:-

prediction : array (1,0,1,0,0,0,0,1,0))
score : 0.8888
linear Reg score : 0.5843
predictions : 0.485
0.5685.

9-5-24

Lab-6 and Lab-7

KNN classification Model

```
import pandas as pd.
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbours import KNeighborsClassifier
from sklearn.metrics import accuracy_score.
import matplotlib.pyplot as plt
import seaborn as sn.


df = pd.read_csv ("iris.csv")
df.head()

classes = df["species"].unique()
colors = ['r', 'g', 'b']

for i, cls in enumerate(classes):
    class_data = df[df["species"] == cls]
    plt.scatter(class_data["sepallength(cm)"], class_data
                ["petallength"], c = colors[i], label = cls)

plt.xlabel("Sepal Length [cm]")
plt.ylabel("Petal Length [cm]")
plt.legend()
plt.show()

for i, cls in enumerate(classes):
    class_data = df[df['species"] == cls]
    plt.scatter(class_data["sepalwidth"], class_data["petalwidth"
                c = colours[i], label = cls)
```

```
plt.xlabel("Sepal width[@cm]")
plt.ylabel("Relax cidth[cm]")
plt.legend()
plt.show()

y = df["Specie"]
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, train_size,
                                   random_state=0)

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, Y_train)
y_pred = knn.predict(X_test)
y2 = knn.predict(X_train)

from sklearn.metrics imports accuracy_score
score = accuracy_score(y_pred, y_test)
print(f"Training Accuracy {score}")
score2 = accuracy_score(y2, y_train)
print(f"Training Accuracy : {score}")
```

Output:-

Training Accuray = 1.0

Accuracy = 1.0

8-6-24

Lab-6

```
from sklearn.svm import SVC
model = SVC(kernel='linear', random_state=0, C=1.0)
model.fit(X_train, y_train)
y_2 = model.predict(X_train, y_train)
```

```
from sklearn.metrics import accuracy_score
score1 = accuracy_score(y_pred, y_test)
print(f"Testing Accuracy : {score}")
score2 = accuracy_score(y2, y_train)
print(f"Training Accuracy : {score2}")
```

Output = Training Accuracy = 1.0

Training Accuracy = 1.0

Lab-8

Build Artificial Neural Network model with back propogation on a given dataset.

⇒

```
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
hiddenlayers_neurons = 3
output_neurons = 1

wh = np.random.uniform(size=(inputlayer_neurons, hiddenlayer_neurons))
bh = np.random.uniform(size=(1, hiddenlayer_neurons))
wout = np.random.uniform(size=(hiddenlayers_neurons, output_neurons)),
bout = bh = 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(hlayers_act, wout)
    outinp = outinp1+bout
    output = sigmoid(out.inp)
    EO = y-output
    outgrad = derivatives_sigmoid(output)
    d_output = EO * outgrad
    EH = d_output.dot(wout.T)

hiddngrad = derivatives_sigmoid(hlayer_act)
d_hidden layers = EH * hiddngrad
wout += hlayer_act.T.dot(d_output)*lr
wh += X T.dot(d_hidden.layers)*lr
print("Input : \n", str(X))
print("Actual output: \n" + str(y))
print("Predicted output : \n", output)
```

Output :

Input : [[0.66667 1]
[0.33333 0.66666]
[1 0.666667]]

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

Predicted output : [[0.8913061]
[0.88519821]
[0.88962179]]

Output :-

Testing Accuracy : 1.0

Traing Accuracy : 1.0

(b)

```
from sklearn.ensemble import AdaBoost classifier
adb = AdaBoostClassifier()
adb_model = adb.fit(X_train, y_train)
y_pred = adb.predict(X_test)
y2 = adb.predict(X_train)

from sklearn.metrics import accuracy_score
score = accuracy_score(y_pred, y_test)
print(f"Testing Accuracy : {score}")
score2 = accuracy_score(y2, y_train)
print(f"Traing Accuracy : {score2}")
```

Output :- Testing Accuracy : 0.977777

Traing Accuracy : 1.00.

```
from sklearn.metrics import accuracy_score
score = accuracy_score(y_pred, y_test)
print(f"Testing Accuracy = {score}")
score2 = accuracy_score(y2, y_train)
print(f"Training Accuracy = {score2}")
```

Outputs:-

Testing Accuracy : 1.0

Training Accuracy : 1.0

Lab - 10

1a Build a k-means Algo to cluster a set of data stored in .csv file.

→

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

iris = datasets.load_iris()
x = pd.DataFrame(iris.data, columns=["sepal_length",
        sepal width", "petal_length", "petal width"])
y = pd.DataFrame(iris.target, columns=['Target'])
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['Target']], s=40)

plt.title("Real clusters")
plt.xlabel("petal length")
plt.ylabel("petal width")

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")
plt.show()
```

Lab 11

Implement Dimensionality reduction using PCA method

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

```
from sklearn.preprocessing import StandardScaler
scalar = StandardScaler()
scalar.fit(df)
scaled_data = scaler.transform(df)
```

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pca.fit(scaled_data)
x_pca = pca.transform(scaled_data)
x_pca.shape
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')
```