INDEX

ML lab

Name Udita Singh 1BM21CS260 Std Sec

Roll No. Subject School/College

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

Sl. No.	Date	Title	Page No.	Teacher Sign / Remarks
01.	21-03-24	Import & Export	10	21/3/24
02.	28/3/24	End to End Project	10	28/3/24
03.	4/04/24	Linear Regression	10	4/4/24
04.	18/04/24	Decision Tree	10	18/4/24
05.	25/04/24	Age Prediction using regression	10	25/4/24

Week-1 Import & Export Pandas Library

Import csv

```
import pandas as pd
air_bnb_data = pd.read_csv("sample_data/california_housing_test.csv")
air_bnb_data.head
```

output -

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
0	-122.05	37.37	27.0	3885.0	661.0	1537.0	606.0	6.6085	344700.0

Import 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()
```

Output

	sepal_length_in_cm	sepal_width_in_cm	petal_length_in_cm	petal_width_in_cm	class
0	5.1	3.5	1.4	0.2	Iris-setosa

Export data

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

21/3/24

Week-2 End to End Project 28/03/24

Step-1 Get the data

download

```
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()

fetch_housing_data()

import pandas as pd
def load_housing_data(housing_path = Housing_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 = 0.2):
    shuffled_indices = np.random.permutation(len(data))
    test_set_size = int(len(data) * test_ratio)
```

Step-2 Discover & Visualize

```
strat_train_set.shape, strat_test_set.shape
strat_test_set.reset_index().to_feather(fname = "data/01/strat_test_set.f")
housing = strat_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()
```

Looking for correlations

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

Step-3 Prepare the data for ML Algorithms

Data Cleaning

```
from sklear.impute import SimpleImputer
imputer = SimpleImputer(strategy='median')
housing_num = housing.drop("ocean_proximity", axis=1)
imputer.fit(housing_num)
imputer.statistics
housing_num.median().values
x = imputer.transform(housing_num)
x.shape
```

Handling text

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

Step-4 Select & Train model

```
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))
print("Labels:", some_labels.tolist())
```

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()

Step-5 Fine Tune

param_grid = [{'n_estimators': [3, 10, 30], 'max_features': [2, 4, 6, 8]},]

forest_reg = RandomForestRegressor()

grid_search = GridSearchCV(estimator = forest_reg, param_grid = param_grid,
scoring = 'neg_mean_squared_error', cv = 5)

grid_search.fit(x = housing_prepared, y = housing_labels)

grid_search.best_params_

grid_search.best_estimator_

cvres = grid_search.cv_results

28/3/24

WEEK-3 Regression

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('sample_data/salary_data.csv')
df_sal.head()
df_sal.describe()
plt.title('Salary Distribution Plot')
sns.displot(df_sal['Salary'])
plt.show()
plt.scatter(df_sal['YearsExperience'], df_sal['Salary'])
plt.box(False)
plt.show()
X = df_sal.iloc[:, :1]
Y = df_sal.iloc[:, 1:]
regressor = LinearRegression()
regressor.fit(X_train, Y_train)
y_pred_test = regressor.predict(X_test)
Y_pred_test = regressor.predict(X_train)
plt.scatter(X_test, Y_test, color='lightcoral')
plt.plot(X_train, Y_train, color='firebrick')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.box(False)
plt.show()
print(f'Coefficient : {regressor.coef_}')
print(f'Intercept : {regressor.intercept_}')
```

Output → Coefficient : [[9312.57512673]]
Intercept : [26780.09915063]

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.preprocessing import OneHotEncoder

df_start = pd.read_csv('sample_data/50_Startups.csv')
df_start.head()
df_start.describe()
plt.title('Profit Distribution')
sns.distplot(df_start['Profit'])
plt.show()
plt.scatter(df_start['R&D spend'],df_start['Profit'])
plt.xlable('R&D spend')
plt.ylabel('Profit')
plt.show()
X=df_start.iloc[:,1:].values
Y=df_start.iloc[:,:-1].values
X=np.array(ct.fit_transform(X))
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)),1)
```

Output → array([[103015.2, 103282.38],
[132582, 144259],
...
...]])

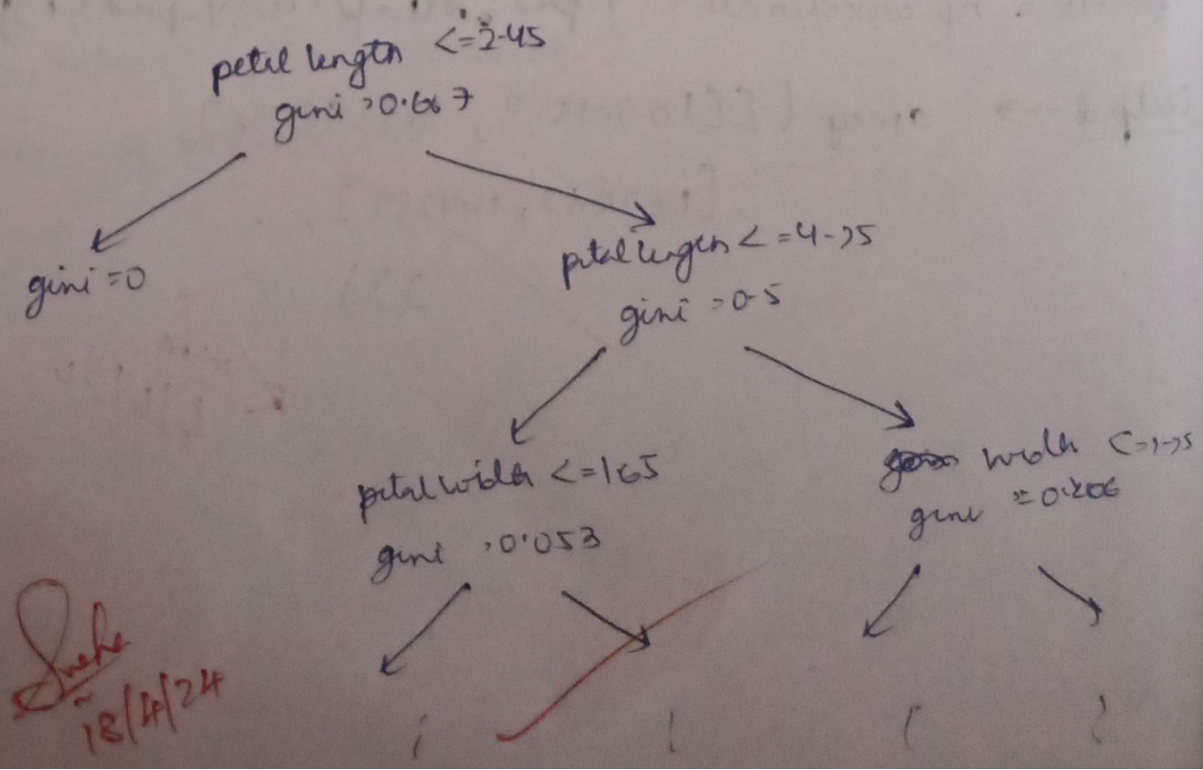
4/4/24

Week-4 Decision Tree

```
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

iris_data = load_iris()
iris_df = pd.DataFrame(data = iris_data.data , column =iris_data.features_na
iris_df ['target '] = iris_data.target
print(iris_df.head())

x =iris_df.drop('target ',axis=1)
y=iris_df ['target']
X_train ,y_train ,X_test ,y_test =train_test_split(X,y, test_size =0.2, random =42)
clf = DecisionTreeClassifier(criterion='gini',random_state=100, depth=5, leaf=8)
clf.fit(X_train,y_train)
plt.figure(figsize = (12, 8))
tree.plot_tree(clf, feature_name= iris_data.feature_name, class_name =
        iris_data.target_names, filled = True)

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

Output –

Accuracy : 1.0



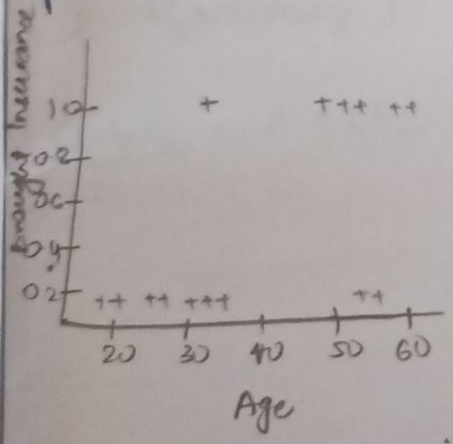
18/4/24

Week-5 Predicting if a person would buy life insurance based on his age using logistic regression

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

df = pd.read_csv("sample_data/insurance_data.csv")
print(df.head())
```

	age	brought_insurance
0	22	0
1	25	0
2	47	1
3	52	0
4	46	1

```
plt.scatter(df.age, df.brought_insurance, marker = '+', color = 'red')
plt.xlable('Age')
plt.ylable('Brought insurance')
plt.show()
```




```
X_train, X_test, Y_train, Y_test = train_test_split(df[['age']], df.brought_insurance, train_size = 0.8)
model = LogisticRegression()
model.fit(X_train, Y_train)
Y_predicted = model.predict(X_test)
probabilities = model.predict_proba(X_test)
accuracy = model.score(X_test, Y_test)
print(accuracy)
```

Model Accuracy : 0.8333

```
print("coefficient (m): ", model.coef_)
print("Intercept (b): ", model.intercept_)
```

Coefficient (m): [[0.15604994]]

Intercept (b): [-6.666603]

```
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 = 35
point ("Probability for age 35:", prediction-function (age-35))
point ("Probability for age 43:", prediction-function (43))
```

Probability for age 35 : 0.48500449

Probability for age 43 : 0.568565229

Sneha
25/4/24

Week-6 KNN classification model for given dataset

```
iris = load_iris()
x = iris.data()
y = iris.target()
df = pd.DataFrame(x, columns=iris.feature_name)
df['target'] = y
plt.figure(figsize(8,6))
plt.scatter(df.iloc[:,0], df.iloc[:,1], c=y, cmap='viridis')
plt.show
X_train, X_test, Y_train, Y_test = train_test(x, y, test_size=0.2,
        random_state=42)
knn = KNeighborsClassifier()
knn.fit(X_train, Y_train)
Y_pred = knn.predic(X_test)
accuracy = knn.score(Y_test, Y_pred)
print(accuracy)
```

Output - 0.9342

Week-7 SVM model

```
iris = load_iris()
x = iris.data
y = iris.target
df = pd.DataFrame(x, columns = iris.feature_names)
df['target'] = y
print(df.head)
plt.scatter(df.iloc[:,0], df.iloc[:,1], c = y, s = 100)
plt.title('Scatter plot for iris data')
plt.show()

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test = 0.3)
svm = SVC(kernel = 'linear')
svm.fit(X_train, Y_train)
Y_pred = svm.predict(X_test)
accuracy = svm.score(Y_test, Y_pred)
```

Output - 0.9834

Week-8 Artificial Neural Network model

```
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
input_layer_neurons = 2
output_neurons = 1
wh = np.random.uniform(size=(input layer neurons, hiddenlayer))
bh np.random.uniform(size=(1, hiddenlayer_neurons))

def sigmoid(x):
    return 1/(1+np.exp(-x))

def derivations_sigmoid(x):
    return x*(1-x)

for i in range(epoch):
    hinp1 = np.dot(X, wh)
    hinp = hinp1 + bh
    hlayr_act - sigmoid(hinp)
    outinp1 = np.dot(hlayr_act, wout)
    outinp = sigmoid(outinp)
    EO = y - output
    outgrad = derivatives_sigmoid(output)
    d_out - EO * outgrad
wout += hlayr_act.T.dot(d_output)*lr
wh += X.T.dot(d_hiddenlayr)*lr
print(str(x))
print(str(y))
print(output)
```

Output

Input :

[[0.66667][0.2333 0.556]]

Output

[[0.76767301] [0.12480 28] [0.7450]]

(9) A. Random forest algorithm

```
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
rf_classifier = RandomForestClassifier()
rf_classifier.fit()
Y_pred = rf_classifier.predict(X_test)
accuracy = accuracy_score(Y_test, Y_pred)
print(accuracy)
```

Output - 0.93333

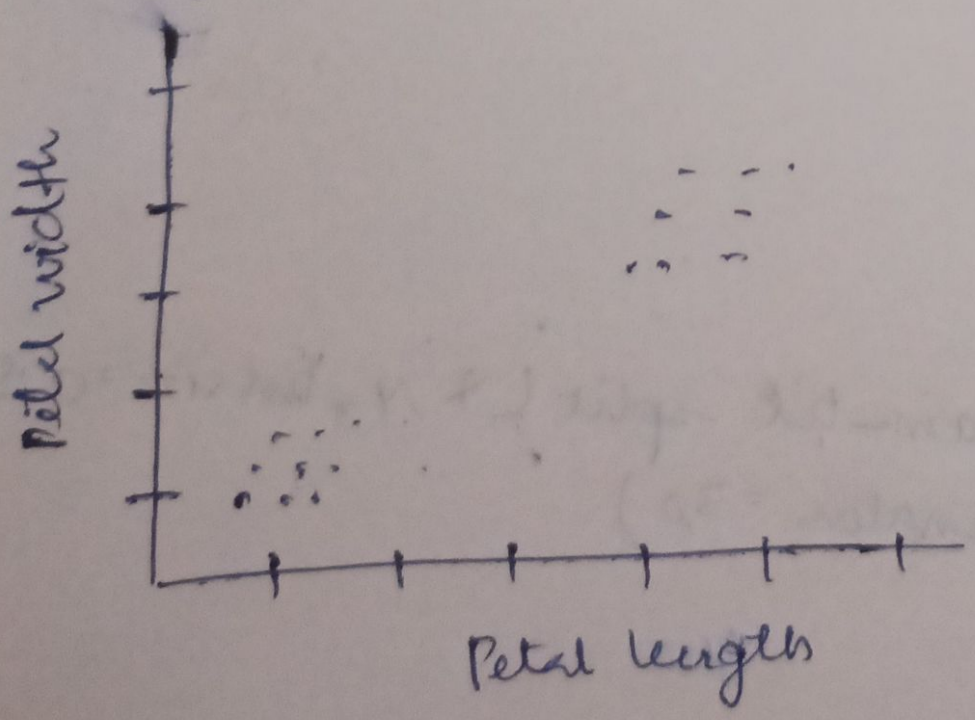
(B) Adaboost

```
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
ada = AdaBoostClassifier(n_estimators = 30)
ada.fit(X_train, Y_train)
Y_pred = ada.predict(X_test)
accuracy = accuracy_score(Y_test, Y_pred)
print(accuracy)
```

Output : 0.9666

Week-10 K means algorithm

```
iris = dataset.load_iris()
X = pd.DataFrame(iris.data)
Y = pd.DataFrame(iris.target)
Y.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)
plt.title('K-means Clustering')
```



Week-11 Principal Component Analysis

```
cancer = load_breast_cancer()
cancer.keys()
print (cancer['DESCR'])
df = pd.DataFrame (cancer['data'], columns = cancer['feature'])
df.head()
scaler = StandardScaler()
scaler.fit(df)
scaled_data = scaler.transform(df)
pca = PCA(n_components = 2)
pca.fit(scaled_data)
```

```
(569, 30)
```

```
x_pca.shape
```

```
(569, 2)
```

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