INDEX

Name Y. Shamil Ahamed. Sem/Std 6 Sec D

Roll No. 1BM21CS248 Subject ML LAB School/College BMSCE

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

Sl. No.	Date	Title	Page No.	Teacher Sign / Remarks
1	21/03/24	Week-1 (imp/Export data)		A
2	28/03/24	week-2 (Ch-2 hands on)		A
3	4/4/24	linear & Multiple Regression		
4	18/4/24	Decision Tree		
5	25/4/24	Logistic Regression (week-5)		A
6	09/5/24	Knn classifier, Svm (week-6)		
7	23/5/24	Prog 8, Prog 9a, 9b (week-7)		
8	30/5/24	prog 10 (K-means) prog 11 (PCA)		

21/03/24 Ass-3-24

Week-1

Q) Write a python prog to import and export data using pandas library functions

i/p:

```
import pandas as pd

iris_data = pd.read_csv(r'C:\Users\Admin\
                        Downloads\iris.csv')
iris_data.head()
```

o/p:

	5.1	3.5	1.4	0.2	Iris-setosa
0	4.9	3.0	1.4	0.2	Iris_setosa
1	4.7	3.2	1.3	0.2	Iris_setosa
2	4.6	3.1	1.5	0.2	Iris_setosa
3	5.0	3.6	1.4	0.2	Iris_setosa
4	5.4	3.9	1.7	0.4	Iris_setosa

using URL

i/p:

```
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()
```

o/p: Same as the previous.

Kaggle:

Creating Data:-

1) Dataframe:

eg: `pd.DataFrame({'yes': [50, 21], 'No': [131, 2]})`

	yes	No
0	50	131
1	21	2

2) Series:

eg: `pd.Series([1, 2, 3, 4, 5])`

0	1
1	2
2	3
3	4
4	5

Reading Data:

Eg: `data = pd.read_csv("college_data.csv")`

`data.head()` -> returns first five data values of dataset.

28/03/24

Week-2

Chapter-2

2) Get the data:

↳ Download the data
↳ take a quick look at the DST
↳ create a test set

- import os
import tarfile
import urllib

fetch_housing_data()

def load_housing_data(housing_path=HOUSING_PATH):

1) Look at the big picture.

↳ frame the problem
↳ select a performance measure
↳ check the assumptions

3) Discover & visualize the data to Gain Insights:

*) Put the test set aside, explore only the training set.

Strat_train_set.shape, Strat_test_set.shape

housing = Strat_train_set.copy(); housing.shape

→ Visualizing Geographical Data:

since latitude/~~log~~ longitude info is there use matplotlib plot method of pandas.

.plot()

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

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

→ looking for correlations:

```
corr_matrix = housing.corr()
```

→ Experimenting with attribut combinations:

4) Prepare the data for M.L Algorithm

```
housing = strat_train_set.drop("median_house_value",
                               axis=1)
housing_labels = strat_train_set["median_house_value"].
                                 copy()
housing.shape, housing_labels.shape
```

Data cleaning:

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy='median')
housing_num = housing.drop("ocean_proximity", axis=1)
```

5) Select & Train Model

```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(x=housing_prepared, y=housing_labels)

Cross_validation
Scross = cross_val_score(estimation=tree_reg,
        x=housing_prepared, y=housing_label,
        scoring='neg_mean_squared_error', cv=10)
```

6) time frame your model:

```
param_grid =[
{'n_estimates': [3,10,30], 'max_features':
    [2,4,6,8]}, {'bootstrap=[false],
    'n_estimates': [3,10], 'max_features':
        [2,3,4]}]
forest_reg = RandomforestRegression()
grid_search = GridsearchCV(estimation=
    forest_reg, param_grid, scoring='neg-
    mean_Square_error', cv=5, return_train_
        score_trues n_jobs=-1)

grid_search.fit(x=housing_prepared, y=housing_
        labels)
```

11/4/24

Week-3

Linear Regression

① df_sal = pd.read_csv(r'C:\Users\STUDENT... .csv')
df_sal.head()

② plt.title('Salary Distribution plot')
sns.distplot(df_sal['Salary'])
plt.show()

③ plt.scatter(df_sal['yearsExperience'], df_sal['Salary'],
color = 'lightcoral')
plt.xlabel('years of exp')
plt.ylabel('Salary')
plt.box(false)
plt.show()

④ Splitting variables

x = df_sal.iloc[:, :1]
y = df_sal.iloc[:, 1:]

⑤ x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size = 0.2, random_state = 0)

⑥ regressor.fit(x_train, y_train)

⑦ Prediction

y_pred_test = regressor.predict(x_test)
y_pred_test = regressor.predict(x_train)

① Import libraries
② import data
③ Analyze data
④ Split data
⑤ Predict results
⑥ visualize prediction

Multiple Linear Regression

① df_start = pd.read_csv('/content/50-startup_csv')
df_start.head()

② plt.title('profit Distribution plot')
sns.distplot(df_start['profit'])
plt.show()

③ Splitting
x = df_start.iloc[:, :-1].values
y = df_start.iloc[:, -1].values

④ x_train, x_test, y_train, y_test = train_test_split
(x, y, testSize = 0.2, random_state = 0)

⑤ regressor = LinearRegression()
regressior.fit(x_train, y_train)

⑥ Predict.
y_pred = regressor.predict(x_test)

Steps: 1) Import libraries
2) Import data
3) Analyze data
4) Split into Independant / dependant variables.
5) Predict results.
6) Compare predictions.

18/4/24

Week-4

Decision Tree

① cols = df.columns[0:-1]
for i in cols:
    sns.boxplot(y=df[i])
    plt.show()

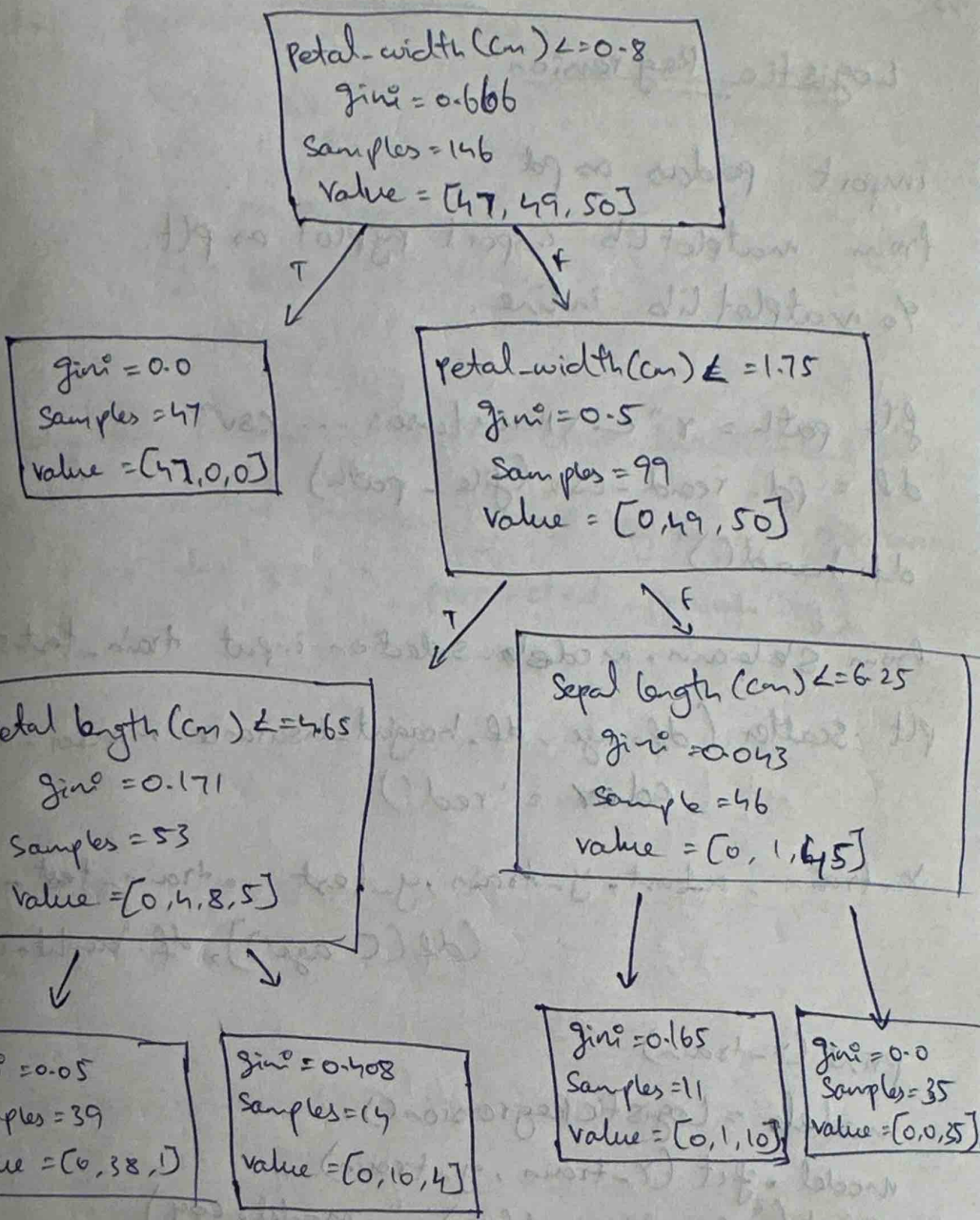
② dt = DecisionTreeClassifier(max_depth=3, min_samples_leaf=10, random_state=1)
dt.fit(x, y)
↳ x = df.drop("species", axis=1)
y = df["species"]

③ features = x_columns.
dot_data = export_graphviz(dt, out_files=None, feature_names=features)
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())

④ dt = DecisionTreeClassify(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)

⑤ print("Accuracy of DecisionTree-Train:", accuracy_score(y_pred_train, y_train))

Print("Accuracy of DecisionTree-Test:", accuracy_score(y_pred, y_test))


petal_width(cm) <= 0.8
gini = 0.666
samples = 146
value = [47, 49, 50]
T
F
gini = 0.0
samples = 47
value = [47, 0, 0]
petal_width(cm) <= 1.75
gini = 0.5
samples = 99
value = [0, 49, 50]
T
F
petal length(cm) <= 4.65
gini = 0.171
samples = 53
value = [0, 4, 8, 5]
Sepal length(cm) <= 6.25
gini = 0.043
sample = 46
value = [0, 1, 45]
gini = 0.05
samples = 39
value = [0, 38, 1]
gini = 0.408
samples = 14
value = [0, 10, 4]
gini = 0.165
Samples = 11
value = [0, 1, 10]
gini = 0.0
Samples = 35
value = [0, 0, 35]

25/01/24

Week-5

Logistic Regression

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

file_path = r"C:\Users\Home\Users ... csv"
df = pd.read_csv(file_path)
df.head()

from sklearn.model_selection import train_test_split
plt.scatter(df.age, df.bought_insurance, marker='+',
            color='red')

x_train, x_test, y_train, y_test = train_test_split
                    (df[['age']], df.bought_insurance

print(x_train)
model = LogisticRegression()
model.fit(x_train, y_train)
print("Co-efficient(m):", model.coef_)
print("Intercept(b):", model.intercept_)

def sigmoid(x):
    return 1/(1+math.exp(-x))
x=0
sigmoid_value = sigmoid(x)
print("Sigmoid value at x,i.e:", sigmoid value)

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

```
def prediction_function(age):
    m = 0.042
    b = -1.53
    z = m * age + b
    y = sigmoid(z)
    return y.

predicted probability = prediction_function(35)
print("predicted probability of buying insurance
    for age 35:", predicted_probability).
```

09/05/24 week-6

KNN classifier:

①
```
import numpy as np
import pandas as pd
import matplotlib as plt
plt.style.use('ssplot')
```

②
```
data = pd.read_csv('/diabetes.csv')
df.head
df.shape
```

③
```
x = df.drop('outcome', axis=1).values
y = df['outcome'].values
```

④
```
from sklearn.model_selection import
        train_test_split
x_train, x_test, y_train, y_test =
    train_test_split(x, y, test_size=0.4,
        random_state=42, stratify=y)
```

⑤
```
from sklearn.neighbors import KNeighbours
                              classifier.
neighbours = np.arrange(1,9)
train_accuracy = np.empty(len(neighbours))
test_accuracy = np.empty(len(neighbours))
Knn.fit(x_train, y_train)
train_accuracy[i,j] = knn.score(x_train, y_train)
test_accuracy[i,j] = knn.score(x_test, y_test)
```

⑥ plt.plot(neighbors, train_accuracy, label='Train')

plt.plot(neighbours, test_accuracy, label='Test')
plt.show()

⑦
plt.scatter(neighbours, train_accuracy, color='R')
plt.show()

⑧ knn.fit(x_train, y_train)

knn.score(x_test, y_test)

o/p Accuracy = 0.7348

9/10

SVM

```
from sklearn import datasets
    cancer = datasets.load_breast_cancer()
    # enter features to print.
    print(cancer.target)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =
        train_test_split(cancer.data, cancer.target,
        test_size = 0.3, random_state = 109)

from sklearn import svm
    clf = svm.SVC(kernel='linear')
    y_pred = clf.predict(X_test)

    from sklearn import metrics
print(metrics.accuracy_score(y_test, y_pred))
print(metrics.precision_score(y_test, y_pred))
print(metrics.          _score(y_test, y_pred))
import seaborn as sns
    sns.scatterplot(x=cancer.data[:,0],
        y=cancer.data[:,1], hue=cancer.target)
    plt.show()
```

o/p: Accuracy: 0.9766

23/5/24 Week-7 A 23-5-24

(ANN) Artificial neural network: (prog 8)

```
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(input_layer_neurons,
                            hiddenlayer_neurons))
bh = np.random.uniform(size = (1, hiddenlayer_neurons))
def sigmoid(x):
    return 1/(1+np.exp(-x))
def derivatives_sigmoid(x):
    return x*(1-x)
for i in range(epoch):
    hinpl = np.dot(x, wh)
    hinp = hinpl + bh
    hlayer_act = sigmoid(hinp)
    outinp = sigmoid(outinp)
EO = y - output
outgrad = derivatives_sigmoid(output)
EH = d_output . dot(wout . T)
```

```
hidden_grad = derivatives_sigmoid(hlayer_act)
d_hiddenlayer = EIt * hidden_grad.
wh += X.T.dot(d_hiddenlayer) * lr
//print I/P & O/P
```

O/P:

Input:

```
[[0.66667  1.    ]
 [0.3333   0.5556]
 [1.       0.66667]]
```

Actual output:

```
[[0.92]
 [0.86]
 [0.89]]
```

Predicted o/p:

```
[[0.8196977]
 [0.7962357]
 [0.8161606]]
```

Prog 9

9a) Random Forest:

```
// import libraries
data = pd.read_csv('Iris.csv')
X = data.drop(columns = ['Species'])
y = data['Species']

X_train, X_test, y_train, y_test =
    train_test_split(X, y, test_size = 0.2, random_state
                                    = 42)
rf_Classifier = RandomForestClassifier(n_estimators = 100,
                                    random_state = 42)
rf_classifier.fit(X_train, y_train)
y_pred = rf.classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
    //print accuracy, classification report

feature_importances = rf.classifier.feature_
                    importances_features = X.columns

importance_df = pd.DataFrame({'Feature': features,
                'Importance': feature_importance}).
                    sort_values(by = 'Importance',
                                ascending = False)
    print(importance_df)
```

o/p: Accuracy: 100.00%

9b) Ada boost

```
// import necessary libraries

data = pd.read_csv('/content/iris.csv')
x = data.drop(columns = ["species"])
y = data["Species"]

X_train, X_test, y_train, y_test = train_test_split
        (x, y, test_size = 0.2, random_state=42)

adaboost_df = AdaBoost_Classifier
        (n_estimators=150, learning_rate=1)

adaboost_df.fit(x_train, y_train)
y_pred = adaboost_df.predict(x_test)

accuracy = accuracy_score(y_test, y_pred)
    //print Accuracy.
```

o/p: Accuracy: 1.0

30/05/24

week-8

30-5-24.

prog 10: K-means algorithm to cluster a set of data stored in a .csv file.

①

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

②

```
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']
```

③

```
model = Kmeans(n_clusters=3)
model.fit(x)
```

④ visual cluster

```
plt.figure(figsize=(14,14))
colormap = np.array(['red', 'lime', 'black'])
```

⑤ orig clus

```
plt.subplot(2,2,1)
plt.scatter(x.petal_length, x.petal_width, c=colormap
                                    [y.targets], s=40)
plt.title('Real clusters')
plt.xlabel('petal length')
plt.ylabel('petal width')
```

model class

⑥ 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')

PCA

Dimensionality reduction using principle Component Analysis method.

① import matplotlib.pyplot as plt
import pandas as pd.
import numpy as np
import seaborn as sns
% matplotlib inline.

② from Sklearn.datasets import load_breast_cancer
cancer.keys()

df = pd.DataFrame(cancer['data'], columns = cancer['feature_name'])

PCA with

③ from sklearn.preprocessing import Standard Scaler as SS

Scaler = SS()
Scaler.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)

Scaled_data.Shape

x_pca.Shape

plot

(5)

```
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')
```