No.	Sign.	Remarks

LAB-1

```
# importing the csv file
import pandas as pd
airbnb_data = pd.read_csv("/contents/cleaned_data.csv")
airbnb_data.head()
```

```
# import from url
import pandas as pd
url = "https://archive.ics.uci.edu.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()
```

Sepal Length	Sepal width	Petal length	Petal Width
5.1	3.5	1.4	0.2
4.9	3	1.4	0.2
4.7	3.2	1.3	0.2
4.6	3.1	1.5	0.2
	3.6	1.4	0.2

Sri
21/2/24

LAB - II

4/4/24

1. Frame the problem

   Select performance measure

   Eg: $RMSE(X,h) = \frac{1}{m} \sum_{i=1}^{m} (h(x^i) - (y^i))^2$

2. Check data

```
import os
import tarfile
import urllib

download_root = " "
housing_path = os.path.join( )
housing_url = download_root: + "

def fetch_housing data (url, path)
    :

fetch_housing_data( )

housing_hist (bins = 50, figsize = ( ) )
plt.show( )
```

3. Discover & visualize data into graph insights

   • plot ( ) → scatter

4. Prepare data for ML algorithms data learning

   drop( )

   ~~fit( )~~

   median( )

5. Select & train model:

training set

linear regression()

decision tree regression()

6. Fine tune model:

grid search

randomised search

ensemble methods

7. Launch, monitor, maintain system

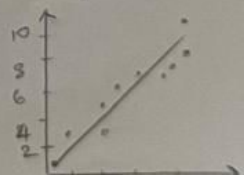
Week - 4

Python Implementation of Linear Regression:

```
import numpy as np
import matplotlib.pyplot as plt

def estimate_coeff(x,y):
    n = np.size(x)
    m_x = np.mean(x)
    m_y = np.mean(y)
    ss_xy = np.sum(y*x) - n*m_y*m_x
    ss_xx = np.sum(x*x)
    b_1 = ss_xy / ss_xx
    b_0 = m_y - b_1*m_x
    return(b_0,b_1)

def plot_regression_line(x,y,b)
    plt.scatter(x,y,color="m",marker="o",s=30)
    y_pred = b[0]+b[1]*x
    plt.plot(x,y_pred,color="g")
    plt.xlabel('x')
    plt.ylabel('y')

def main()
    x=np.array([0,1,2,3,4,5,6,7,8,9])
    y=np.array([1,3,2,5,7,8,9,10,12])
    b=estimate_coeff(x,y)
```

OUTPUT:

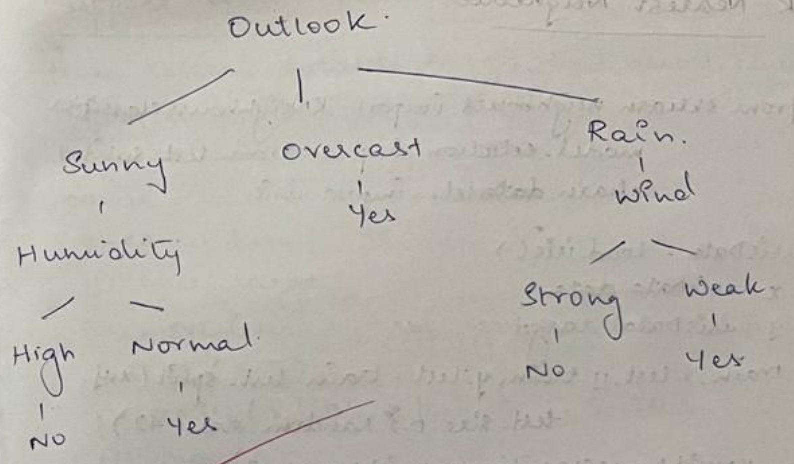
(b_0,b_1) = [1,2,3 . . . 1.16969. . .]



WEEK-5

Decision Tree - ID3

```
import numpy as np
import pandas as pd
df = pd.read_csv('url')
df.head()
df.info()
df.describe()

def find_entropy(df):
    target = df.key()[-1]
    entropy = 0
values = df[target].unique()
for value in values
        p = df[target].value_counts([value] /
                                    len(df[target]))
        entropy += p * np.log2(p)
        return entropy

def buildtree(df, tree = none):
    target = df.keys()[-1]
    node = find_winner(df)
    att = np.unique(df[node])
    if tree is none:
        tree = {}
        tree [Node] = {}
        for value in att:
        sub = get_subtable(df, node, value)
        counts = np.unique(subtable[target],
                           return_counts=True)
```

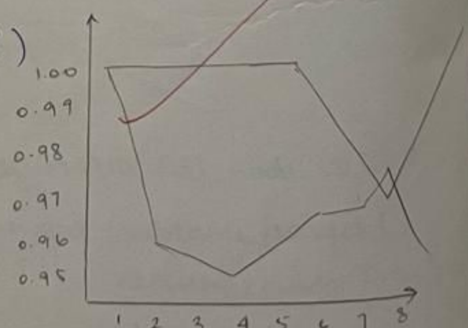


09-05-24

complete KNN LP

K-Nearest Neighbours.

```
from sklearn.neighbours import KNeighboursClassifier
  "        model_selection import train_test_split
  "        sklearn.datasets import iris

irisData = load_iris()
 x = irisData.data
 y = irisData.target
x_train, x_test, y_train, y_test = train_test_split(x,y,
                    test_size=0.2, random_state=42)
knn = KNeighboursClassifier(n_neighbours=7)
knn.fit  = (x_train, y_train)
print (knn.predict (x_test))

neighbours = np.arange(1,9)
for i,k in enumerate(neighbours)
    knn = KNeighboursClassifier(n_neighbours=k)
    knn.fit(x_train, y_train)
train_accuracy[i] = knn.score (x_train, y_train)
test_accuracy[i] = knn.score (x_test, y_test)
plt.plot (neighbours, test_accuracy, label = 'Testing dataset
                                               accuracy')
  "   (          "   , train_accuracy,  . . )

plt.legend ()
plt.xlabel ('n_neighbours')
plt.ylabel ('Accuracy')
plt.show()
```



Support Vector Machine:

```
from sklearn.datasets import load_breast_cancer
import matplotlib.pyplot as plt
from sklearn.svm import SVC
cancer = load_breast_cancer()
 x = cancer.data [: :2]
 y = cancer.target
 SVM = SVC(kernel = "rbf", gamma = 0.5, C=10)
 Svm.fit(x, y)
 plt.scatter(x [:0], x [1,1], c = y, s = 20)
 plt.show()
```

PCA using sklearn:

```
from sklearn.decomposition import PCA
pca = PCA (n_components = 2)
pca.fit(2)
 x_pca = pca.transform(2)
 df_pca = pd.Dataframe (x_pca, columns = ['PCA',
                 format(i+1) for i, in range
                               (n_components)])

print(df_pca)
```

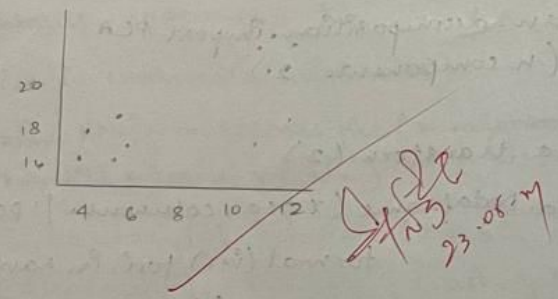
Output:

	PC 1	PC 2
0	9.1841	1.9468
⋮		

K-Means Clustering

```
from sklearn.cluster import KMeans
data = list(zip(x,y))
inertias = []
for i in range(1,11):
    KMeans = KMeans(n_clusters = 2)
    KMeans.fit(data)
plt.scatter(x,y, c=kmeans.labels)
plt.show()
```

Result:



23.08.24

Random Forest

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,
    classification_report
import warnings
warnings.filterwarnings('ignore')
url = "https://.../dataset"
titanic_data = pd.read_csv(url)

titanic_data = titanic_data.dropna(subset=['Survived'])
X = titanic_data[['P_class', 'Sex', 'Age', 'Sibsp',
                  'Parch', 'Fare']]

y = titantic_data['Survived']
X.loc[:,'Sex'] = X['Sex'].map({'female':0', 'male':1})

X.loc[:,'Age'].fillna(X['Age'].median(), inplace=True)
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)
accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test,
                                           y_pred)

print("Classification Report", classification_rep)
```

30.05.24

Output:

Accuracy: 0.8