LAB-1

① Complete the upload of reading of csv file.

```
import pandas as pd
airbnb_data = pd.read_csv("./iris/iris.data")
Print (airbnb_data.head())

url = " https://archive.ics.uci.edu/ml/iris/iris.data"
col_names = ["Sepal_l", "Sepal_w", Petal_l", "Petal_w", "class"]
iris_data = pd.read_csv (url, names = col_names)
Print ( Iris_data.head (5))
Iris_data.to_csv ("cleaned_iris_data.csv")
```

	Sepal_l	Sepal_w	Petal_l	Petal_w	class
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa

NLP 21/8/24

LAB-2

=> End to End Machine - Learning - Project

Imports required:

os for joining paths
tarfile for opening tarfiles
urllib for downloading data
matplot.pyplot for plotting data

fetching data:

os.makedirs for creating a directory

download the data using urllib.request.urlretrieve
(url, filename)

extract the files using tarfile.open()
.extract all()

load dataframe:

Pd.read_csv(data_path) to create a dataframe

Create test data:

np.random.permutations() → generates random numbers
test_size ⇒ store the % of data in test_size

indices[:test_set_size] → first set
indices[test_set_size:] → second set

Use a seed to get the same data always (or) store in a file.

Linear Regression

```
def estimate_coef(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) - n*m_x*m_x

    b_1 = SS_xy / SS_xx
    b_0 = m_y - b_1*m_x

def plot_regression_line(x, y, b):

    plt.scatter(x, y)

    y_pred = b[0] + b[1]*x

    plt.plot(x, y_pred)


def main():

    x = np.array([0, 1, 2, 3])
    y = np.array([1, 3, 2, 5])
    plot_regression_line(x, y, b)
    plt.show()

main()
```

Decision Tree

```
def importData():

    balance_data = pd.read_csv('./iris/iris.data')

def splitDataset(balance_data):

    x = balance_data.values[:, 1:5]
    y = balance_data.values[:, 0]

    train_test_split(x, y, test_size=0.3)

def train_using_gini(x_train, x_test, y_train):

    clf_gini = DecisionTreeClassifier(
                    criterion = "gini",
                    random_state, max_depth = 3)

    clf_gini.fit(x_train, y_train)

def plot_decision_tree():

    plt.figure(size = (15,10))
    plot_tree(clf_object, filled = True)
    plt.show()
```

Logistic Regression

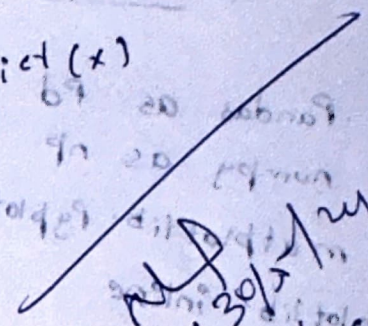
```
df = pd.read_csv("insurance_data.csv")

plt.scatter(df.age, df.bought_insurance, marker='+', color='red')

x_train, x_test, y_train, y_test = train_test_split(age,
                                   insurance, train_size=0.8)

model = LogisticRegression()
model.fit(x_train, y_train)

y_predicted = model.predict(x_test)

model.predict_proba(x_test)
```

Knn

```
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 = KNeighborsClassifier(n_neighbours = 7)

knn.fit(x_train, y_train)

Print(knn.Predict(x_test))
```

K-means Implementation

```
Import Pandas as pd
data = pd.readcsv ("iris.csv")
data.head()

import numpy as np
Import seaborn as sns
Pmport matplotlib.Pyplot as plt
from sklearn.datasets import loadiris
from sklearn.cluster import Kmeans

x, y = load_iris (return_x-y = True)
K means = Kmeans (n_cluster = 3, randomstate = 2)
Kmeans.fit(x)
Pred = Kmeans.fit.Predict(x)
```



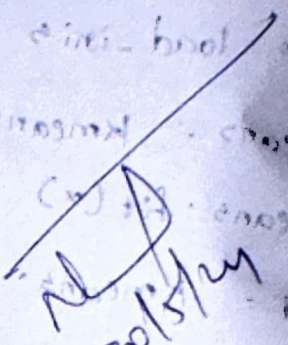
Svm

```
from sklearn.datasets import load_breast_cancer
import matplotlib.Pyplot as plt
from sklearn.inspection import DecisionBoundryDisplay
from sklearn.svm import svc

Cancer = load_breast_cancer()
x = cancer.data [:, :2]
y = Cancer.target

svm = svc (Kernel: "rbf", gamma = 0.5,
svm.fit = (x, y)
```

Decision Boundary Display from estimator(

svm,
x,
response_method = "predict",
map = plt.cm.spectral
alpha = 0.8
xlabel = cancer.feature_names[0]
ylabel = cancer.feature_names[1],
)

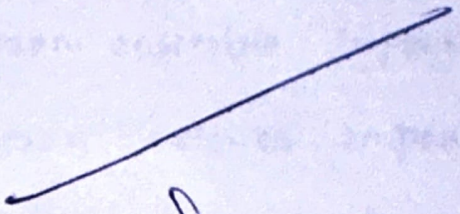
plt.scatter(x[:,0], x[:,1], c=y, s=20,
edgecolors = "k")



PCA

import Pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.decomposition import PCA
from sklearn.preprocessing import Standard scale
from sklearn.datasets import load_breast_cancer
data = load_breast_cancer()
data.keys()
Print (data['feature_names'])
df1 = pd.DataFrame(data['data'], columns = data['feature_names'])
Scaling = StandardScaler()
Scaling.fit(df1)
Scaled_data = scaling.transform(df1)

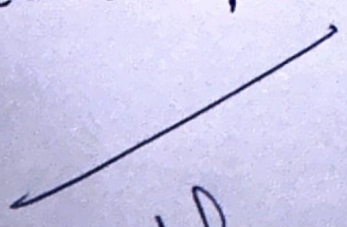
```
Principal = PCA ( n_components = 3)
Principal.fit ( scaled_data)
x = Principal.transform ( scaled_data)

plt.figure ( figSize = (10,10) )
plt.scatter [x[:0], x[:1], c = data ['target'] )
plt.xlabel ('Pc1')
plt.ylabel ('PC2')
```

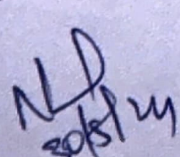


30/8/24

Random- forest

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

iris = datasets.load_iris()
iris_data = pd.DataFrame(iris);
iris_data.head()
model = RandomForestClassifier(n_estimators=100)
model1.fit(x_train, y_train)
feature_imp = pd.Series(data=model1.feature_importances_,
                        index=iris.feature_names)

feature_imp.
```





Boosting

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score

iris = load_iris()
X = iris.data
Y = iris.target

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                   test_size=0.4, random=32)

adaboost_clf = AdaBoostClassifier(n_estimators=30,
  learning_rate=1, random_state=42)

adaboost_clf.fit(x_train, y_train)
y_Pred = adaboost_clf.Predict(x_test)
```

30/5/24