INDEX

Name Yapara Karthikeya Std VI Sec D

Roll No. 1249 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	import & export data using pandas library function		
2.	28-03-24	week - 2		
3.	4-04-24	week - 3		
4.	18-04-24	week - 4		
5.	25-04-24	week-5		
6.	9-05-24	week - 6		
7.	23-05-24	week - 7		
8.	30-05-24	week - 8		

week - 1 A 21-3-24 Date 21/03/24

Page 1

write a python program to import and export data using pandas library function

in:-

```
import pandas as pd
airbnb_data = pd.read_csv("dataset.csv")
airbnb_data.head()
```

output:-

	year	industry_aggregation_NZSIOC	industry_code NZSIOC
0	2021	Level-1	99999
1	2021	level 1	99999

in:-

Reading data from URL

```
url = "https://archive.ics.uci.edu/ml/machine
        learning-database/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_name
iris_data.head()
```

output:-

	sepal_length_in_cm	sepeal_width_in_cm	petal_length_in_cm
0	5.1	3.5	1.4
1	4.9	3.0	1.4
2	4.7	3.2	1.4
3	4.6	3.1	1.4
4	5.0	3.6	1.4

Date __/__/__
Page _____

Kaggle

```
import pandas as pd
```

Data frame

```
Pd.DataFrame({'Yes': [50, 21], 'No': [131, 2]})
```

	Yes	NO
0	50	131
1	21	2

```
Pd.DataFrame({'Bob': ['I liked it.', 'It was awful.'],
              'Sue': ['Pretty good.', 'Bland.']})
```

	Bob	Sue
0	I liked it	Pretty good
1	it was awful	Bland

Series

```
Pd.series([1, 2, 3, 4, 5])
```

```
0    1
1    2
2    3
3    4
4    5
dtype: int64
```

```
Pd.series([30, 35, 40], index = ['2015 Sales', '2016 Sales',
                                 '2017 Sales'],
                                 name = 'product A')
```

```
2015 Sales    30
2016 Sales    35
2017 Sales    40
name: product A, dtype: int64
```

Date 28/03/24

week - 2

Page

End - End ML project

A 28/3/24

1. get the data:-

```
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.makedir(name=housing_path, exist_ok=True)
tgz_Path = os.path.join(housing_path, housing.tgz")
urlib.request.urlretrieve(url=housing_url, filename=tgz_path)
housing_tgz = tarfile.open(name=tgz_path)
housing_tgz.close()

# to down data, call the function
fetch_housing_data()
```

Date __/__/__
Page ____

3. Discover & visualize the data to gain insights

latitude/longitude info can be represented using .plot()

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

plt.show()
```

To estimate densities use 'alph' parameter

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

To check the rotation between two or more data info, 'corr' method is use

```
housing[['population', 'median_house_value']]
```

for a better correlation we need to do

```
corr_matrix = housing.corr()

corr_matrix['median_house_value']
                    sort_values(ascendin
```

Date __/__/__

Page _____

4. prepare Data for ML Algorithms

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

Now we fit impute over our data

fitting is basically training

```
imputer.fit(housing_num)
```

Now we can use this to transform the numbers by replacing their missing values values with corresponding medians

```
X = imputer.transform(housing_num)
```

X is a numpy array with the transformed features. we add it back to the dataframe

```
housing_tr = pd.DataFrame(data=X, index=
                          housing_num.index,
                          columns=housing_num
                          .columns)
```

Date __/__/__
Page _____

5- Select & Train a model

```
lin_reg = linear regression()
lin_reg.fit(X = housing_period, y = housing_lab
some_data = housing.iloc[:5]
some_label = housing_labels.iloc[:5]
some_data_prep = full_pipeline.transform
                 (some_data)
```

some_labels.tolist() is not accurate. So, we use another method

```
from sklearn.metrics import mean
                            squad error
housing_pred = lin_reg.predict(housing_prep)

lin_mse = mean_squared_error(housing_labels,
                             housing_predictions
                             )

lin_rmse = np.sqrt(lin_mse)

lin_rmse
```

this is the root mean squared error which is more accurate

linear regression has better performance but

Date 04/04/24

Page

week - 3

```
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model,
matrics
dataurl = 'https://lib.st
from sklearn.model_selection import
train_test_split
from sklearn.compose import ColumnTransformer
from sklearn.linear_model import
linear regression

import data

df_start = pd.read_csv('50_Startups.csv')
df_start.head()
```

Analyze data

```
df_start.describe()
plt.title('profit distribution plot')
sns.distplot(df_start['profit'])
plt.show()

plt.scatter(df_start['R&D Spend'],
            df_start['Profit'])
plt.show()

plt.scatter(df_start['R&D Spend']
            df_start['Profit'], color=
            'Light coral')
plt.title('Profit vs R&D spend')
```

Date __/__/__
Page

```
plt.xlabel('R&D Spend')
plt.ylabel('Profit')
plt.box(False)
plt.show()
```

Split into independent / Dependent Variable

```
x = df.Start.iloc[:, :-1].values
y = df.Start.iloc[:, -1].values
```

Encoding

```
ct = ColumnTransformer(transformers=
    [('encoder', OneHotEncoder(), [3])]
    remainder='passthrough')

x = np.array(ct.fit_transformer(x))
```

Split into Train/Test

```
X_train, x_test, y_train, y_test=
train_test_split(x, y, test_size=0.2
                 random_state=0)
```

Train model

```
regressor = LinearRegression()
regressor.fit(x_train, y_train)
```

predict_model

```
y_pred = regressor.predict(x_te=
```

Date __/__/__

Page _____

Compare predictions

```
np.set_printoptions(precision = 2)
result = np.concatenate((y_pred.
                        reshape(len(y_pred),1))
                        y_test.reshape(len(y_test),1))

result
```

Simple linear regression

```
import numpy as np
import matplotlib.pyplot as plt.
import pandas as pd
import seaborn as sns
from sklearn.model_selection import
train_test_split
from pandas.core.common import
_state
from sklearn.linear_model import
linearRegression
df_sal = pd.read_csv("/content/sal
df_sal.head()
df_sal.describe()
plt.title("salary distribution plot")
sns.displot(df_sal['salary'])
plt.show()
plt.scatter(df_sal['Years Exp'], df_sal['sal'],
            color = 'lightcoral')
plt.title('sal vs Exp')
plt.xlabel('Exp')
plt.ylabel('Salary')
plt.box(False)
plt.show()

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

split into test set
x_train, x_test, y_train, y_test
= train_test_split(x, y, test_size=0.2,
                   random_state=0
```

Date __/__/__

Page _____

Train model

```
regressor = LinearRegression()
regressor.fit(x_train, y_train)
```

Predict_Results:

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

Visualize predictions;

```
plt.scatter(x_train, y_train, 'lightcoral')
plt.plot(x_train, y_pred_train, color='firebrick')
plt.title('Salary vs Exp(Training set)')
plt.xlabel('Years of Exp')
plt.ylabel('Salary')
plt.legend(['x_train/pred(y_test)',
            'x train/y_train'], title='salary
            loc='best', face color='white')

plt.box(False)
plt.show()

plt.scatter(x_test, y_test, Color='lightcoral
plt.plot(x_train, y_pred_train, color='firebrick
plt.title('Sal vs Exp(Test set)')
plt.title('Sal vs Exp(Test set)')
plt.xlabel('Years of Exp')
plt.ylabel('Sal')
plt.show()
```

Coefficient and Intercept

```
print(f'Coeff: {regressor.coef}')
print(f'Interept: {regressor.intercept}
```

output:

```
coef: [9312.57]
Intercept: [26780.099]
```

Date 18/04/2

Page

week - 4

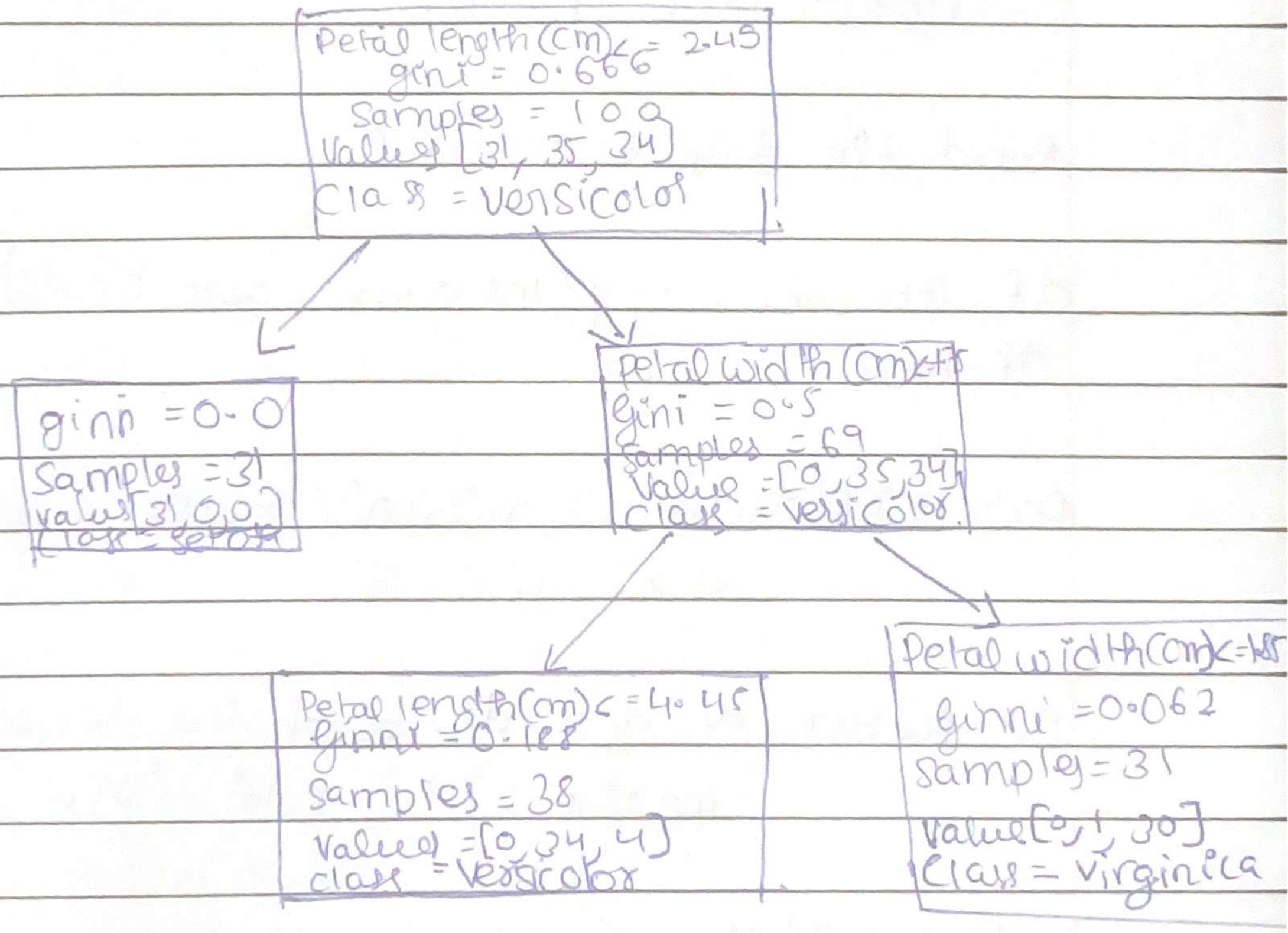
```
import pandas as pd
import matplotlib.pylot as Pd
from sklearn.datasets import load_iris
impord seaborn as sns
from sklearn.model_selection import
train_test_split
from sklearn.tree import DescisionTree
classifier
from sklearn import tree

iris_data = load_iris()
iris_df = Pd.dataFrame(data=iris_data.
                columns=iris_data.feature_names)
iris_df['target'] = iris_data.target

print(iris_df.head())

x = iris_df.drop('target', axis=1)
y = iris_df['target']

x_train, x_test, y_train, y_test =
train_test_split(x, y, testsize=0.33,
    random_state=42)

clf = DescisionTreeClassifier(criterion=
'gini', random_state=100, max_depth=5
min_samples_leaf=8)

clf.fit(x_train, y_train)

DescisionTreeClassifier(max_depth=5,
min_samples_leaf=8, randomstate=100)
```

```
accuracy = clf.score(x_test, y_test)
print("Accuracy:", accuracy)
```

Accuracy = 0.98

Plot the decision tree

```
plt.figure(figsize=(12,8))
tree.plot_tree(clf, feature_names=iris.data.
               features_names, class_names=
iris.data.target_names, filled=True)
plt.show()
```



Week -5 A 23-5-24

Date __/__/__
Page

Logistic Regression and Sigmoid Function

Import

```
import pandas as pd
from matplotlib import pylot as plt
from sklearn model_selection import
                                  test train split
from sklearn linear model import logistic
                                  Regression LR
import math
% matplotlib inline
```

Read the dataset (csv file)

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

Data exploration & visualization

```
# x , y
plt.scatter (df.age, df.bough_insurance
             marker = "+", color = 'red')
```

perform split

```
x_train, x_test, y_train y_test = tts(df
                                [['age']]
                                df.boughtInsurance, train_size
                                random_state = 42)
```

Date __/__/__

Page _____

Create & fit the model:

```
model = LR()
model.fit(x_train, y_train)
x_test # display.
```

Predict

```
y_predict = model.predict(x_test)
y_predict #display.
```

Accuracy:

```
model.predict_proba(x_test)
model.score(x_test, y_test)
```

→ 0.9090

Sigmoid function

```
model.coef
→ array([[0.12335856]])
```

```
model.intercept_
→ array([-4.87081951])
```

Analyze Sigmoid:

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

Predic function:

```
def predic function(age):
    z=0.042 * age - 1.53
    y = Sigmoid(z)
    return y
```

check values with probability

age = 35

prediction function(age)

→ 0.485 age = 88 → 0.89715

Date __/__/__
Page

week - 6

KNN & SVM

```
import pandas as pd
from matplot import pyplot as plt
import seaborn as sns
% matplotlib inline

Read the dataset
df = pd.read_csv("/content/Iris.csv")

Show the plot:
plt.show()
plt.figure(figsize = (10, 6)
sns.scatterplot(data = df, x = df["petal length"]
                y = df["sepal width"]
Add labels & legend = False, hue = df["species"]

Add labels & Title to plot:
plt.xlabel("petal length")
plt.ylabel("sepal width")
plt.title("petal length vs sepal width")
plt.show()

KNN:
from sklearn.model_selection import
                test_train_split as tts
from sklear.neighbour import KNeighbour classifier
x_train, x_test, y_train, y_test = tts(x,y,0.5,0)
neigh = KNeighbour classifier(n.neighbour =2)
neigh.fit(x,y)
```

Date __/__/__

Page _____

```
y_pred = neigh.predict(x_test)
acc = accuracy_score(y_pred, y_test)
print(acc)
```

→ 1.0

SVM:

```
from sklearn.svm import SVC
model = SVC()
model.fit(x, y)
y_pred = model.predict(x_test)
acc = accuracy_score(y_pred, y_test)
print(acc)
```

→ 1.0

Date __/__/__
Page ____

week - 7

ANN

```
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=(inputlayer_neurons,
                                hidden_layer_neurons)
bh = np.random.uniform(size=(1,hidden_layer_neurons)
wout = np.random.uniform(size=(hidden_layer_neurons,
                                output_neurons)
bout = 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)
    outinp = outinp1 + bout
    output = sigmoid(outinp)
```

Date __/__/__

Page _____

```
EO = y - output
outgrad = derivatives_sigmoid (output)
d_output = EO * outgrad
EH = d_output.dot(wout.T)

hiddengrad = derivatives_sigmoid(hlayer
d_hiddenlayer = EH * hiddengrad

wout += hlayer_act.T.dot(d_output) * lr
wh += X.T.dot(d_hiddenlayer) * lr
print("Input : \n" + str(x))
print("actual output : \n" + str(y))
print("predicted output : \n", output)
```

~~Input :~~

~~[[0.6666667 1.]~~

~~[0.333333]~~

Date __/__/__
Page

9A Random Forest

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.dataset import load_iris
from sklearn.model selection import
                                  train_test_split
from sklearn.ensemble import RandomForest

iris_data = load_iris()
iris_df = pd.dataframe(data=iris_data,
                       columns=iris_data.feature_names)
iris_df['target'] = iris_data.target

print(iris_df.head())

x = iris_df.drop('target', axis=1)
y = iris_df['target']

x_train, x_test y_train y_test
                = train_test_split
                  (x, y, test_size=0.3,
                   random_state=42)

clf = RandomForestClassifier(n_estimators=50)
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)

import sklearn.metrics as metrics
accuracy = metrics.accuracy_score(y_test
                                  , y_pred)
print("Accuracy", accuracy)
Accuracy: 1.0
```

Date __/__/__

Page _____

AdaBoost;

#load Data same as Random Forest

x_train x_test, y_train, y_test = tts(x, y, train size=0.7 random=0)

from sklearn.ensemble import AdaBoostClassifier as ada

model = ada(n_estimators=50)

model.fit(x, y)

y_pred = model.predict(x_test)

a = acc(y_pred, y_test)

print(a)

→ 1.0

Asso 5-24

30-05-24

week-8

KMEANS:-

```
import matplotlib.pylot as plt
from sklearn import datasets
from sklearn.cluster import KMEAN
import pandas as pd
import numpy as np

iris = dataset.load_iris()
x = pd.DataFrame(iris.data)
x = column['sepal_length', 'sepal width',
           'petal_length', 'petal width']
y = pd.Dataframe(iris.target)
y.columns = ['Target']

# Building model
model = KMeans(n_cluster=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 Cluster')
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-Mean Clustering')
plt.xlabel('petal length')
plt.ylabel('Petal width')
```

A30-5-24

30-05-24

week-8

KMEANS:-

```
import matplotlib.pylot as plt
from sklearn import datasets
from sklearn.cluster import KMEANS
import pandas as pd
import numpy as np

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

# Building model
model = KMeans(n_cluster=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 Cluster')
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-Mean Clustering')
plt.xlabel('petal length')
plt.ylabel('petal width')
```

Date __/__/__

Page _____

PCA

```
import matplotlib.pylot as plt
import pandas as pd
import numpy as np
import seaborn as sns
% matplotlib inline
from sklearn.dataset import load_breast_cancer
cancer = load_breast_cancer()
cancer.keys()
print(cancer['DESCR'])

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

from sklearn.preprocessing import standardscaler
scaler = StandardScaler()
scaler.fit(df)
StandardScaler(copy=True, withmean=True,
                width_std=True)
scaled_data = scaler.transform(df)
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pca.fit(scaled_data)
PCA(copy=True, n_components=2, whiten=False)

X_pca = pca.transform(scaled_data)
scaled_data.shape
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')
```