INDEX

Name Saurav Chhetri

Sub. Machine-learning lab

Std.: 6th

Div. 6th Sem

Roll No. 1BM21CS194

Telephone No. 8580581844

E-mail ID. sauravchhetri.cs21@bmsce.ac.in

Blood Group.

Birth Day.

Sr.No.	Title	Page No.	Sign./Remarks
1	import & Export database using pandas	21/3/24	10
2	End-End ML Project	28/3/24	10
3.	Linear & multilinear regression	04/04/24	10
4.	Decision tree	18/04/24	10
5.	Logistic Regression	25/04/24	10
6.	K-Nearest Neighbour	09/05/24	10
7.	SVM	09/05/24	10
8.	Random Forest	23/05/24	10
9.	Adaboost	23/05/24	10
10.	ANN	23/05/24	10
11.	K-means algorithm	30/05/24	10
12.	Principal component analysis	30/05/24	10

Teacher's Sign

21/3/24

LAB 1

Write a program to import and export data using pandas library functions.

1) 1) Download listing-austin.csv file

code:-

```
import pandas as pd
airbnb_data = pd.read_csv("data/listings_austin.csv")
airbnb_data.head()
```

Output:-

	id	name	host_id	hostname
0	2265	zen-fast in the heart of Aust	2466	Paddy
1	5245	Eve friendly, colorful, clean	2466	Paddy
2	5456	Walk to 6th, Rainey St and Co	8028	Sylvia
3	5769	Mini Austin Room	8186	Elizabeth
4	6413	Gem of a studio newtown	13879	Todd

2) Reading data from URL

```
# webpage URL
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"

# define the column names
col_names = ["sepal_length_in_cm", "sepal_width_in_cm", "petal_length_in_cm", "petal_width_in_cm", "class"]

# Read data from URL
iris_data = pd.read_csv(url, names=col_names)
iris_data.read()
```

Output :-

	sepal length(cm)	sepal width(cm)	petal length(cm)	petal width(cm)	class
0	5.1	3.5	1.4	0.2	iris-setosa
1	4.9	3.0	1.4	0.2	iris-setosa
2	4.7	3.2	1.3	0.2	iris-setosa
3	4.6	3.1	1.5	0.2	iris-setosa
4	5.0	3.6	1.4	0.2	iris-setosa

1)

25-4-24

Week-2 28/3/24

End-End ML Project

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.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()

# download the data and call the function
    fetch_housing_data()

# small function to load the data using pandas

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)

housing = load_housing_data()

housing.head()
```

Week-2 28/3/24

End-End ML Project

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.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()

# download the data and call the function
    fetch_housing_data()

# small function to load the data using pandas

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)

housing = load_housing_data()

housing.head()
```

Output:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms
0	-122.23	37.88	41.0	880.0	129.0
1	-122.22	37.86	21.0	7099.0	1106.0
2	-122.24	37.85	52.0	1467.0	190.0
3	-122.25	37.85	52.0	1274.0	235.0
4	-122.25	37.85	52.0	1627.0	280.0

2) Discover & visualize the data to gain Insights

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

```
housing.plot(kind='scatter', x='longitute', y='latitude')
pt.show()
```

To estimate densities, use 'alph' parameter

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

To check the relation between two or more data info, 'corr' method is used

```
housing[['population', 'median_house_value']].corr()
```

Output:

	population	median_house_value
population	1.00000	-0.02692
median_house_value	-0.02692	1.00000

This generates a weak correlation between the specified parameters

for a better correlation, we need to do

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

Output

median_housing_value	1.000000
median_income	0.687160
total_rooms	0.135097
housing_median_age	0.114110
households	0.064506

3) Prepare data for ML Algorithms

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy='median')
```

This can handle only numbers.

```
housing_num = housing.drop("ocean_proximity",
                           axis=1)
```

Now we fit imputer 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 with corresponding medians

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

Output: (16512, 8)

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)
housing_tr.head
```

Output

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms
17606	-121.89	37.29	38.0	1568.0	1568.0
18632	-121.93	37.05	14.0	679.0	679.0
14650	-117.20	32.77	31.0	1952.0	1952.0
3230	-119.61	36.31	25.0	1847.0	1847.0
3555	-118.59	34.23	17.0	6592.0	6592.0

25-4-24

using appropriate dataset.

→

Simple 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('/content/salary_Data.csv')
df_sal.head()

df_sal.describe()

plt.title('Salary Distribution Plot')
sns.distplot(df_sal['salary'])
plt.show()

plt.scatter(df_sal['Years Experience'],df_sal['Salary'],
            color='light coral')

plt.title('salary vs Experience')
plt.xlabel('Years Of Experience')
plt.ylabel('Salary')
plt.box(False)
plt.show()

x=df_sal.iloc[:,:1]
y=df_sal.iloc[:,1:]
x_train,x_test,y_train,y_test=train_test_split
    (x,y,test_size=0.2,random state=0)

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

```
y_pred_test = regressor.predict(X_test)
y_pred_train = regression.predict(X_train)

plt.scatter(X_train, y_train, color='light coral')
plt.plot(X_train, y_pred_train, color='firebrick')
plt.title('Salary vs Experience (Training Set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.legent(['X_train/Pred(y_test)', 'X_train/y_train'],
    title='Sal/Exp', loc='best', facecolor='white')
plt.box(False)
plt.show()

plt.scatter(X_test, y_test, color='light coral')
plt.plot(X_train, y_pred_train, color='firebrick')
plt.title('Salary vs Experience (Test Set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.legend(['X_train/Pred(y_test)', 'X_train/y_train'], title='Sal/Exp', loc='best', facecolor='white')
plt.box(False)
plt.show()

print(f'Coefficient: {regressor.coef_}')
print(f'Intercept: {regressor.intercept_}')
```

Multi 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
from sklearn.linear_model import LinearRegression

df_start = pd.read_csv('/content/50_startups.csv')
df_start.head()

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'],
            color='lightcoral')
plt.title('Profit vs R&D spend')
plt.xlabel('R&D spend')
plt.ylabel('Profit')
plt.box(False)
plt.show()

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

ct = ColumnTransformer(transformers = [('encoder',
OneHotEncoder(), [3])], remainder = 'passthrough')
x = np.array(ct.fit_transform(x))
x_train, x_test, y_train, y_test = train_test_split(x, y,
```

```
regressor=linearRegression( )
regressor.fit(x_train,y_train)
y_pred=regressor.predict(x_test )
np.set_printoptions(precision=2)
result=np.concentrate((y_pred.reshape(len(y_pred),
        1),y_test.reshape(len(y_test,1)),1)
result.
```

25-4-24

Week - 4 18/04/2024

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
import seaborn as sns

from sklearn.datasets import load_iris

# load the Iris dataset
iris_data = load_iris()

# display the dataset description
print(iris_data.DESCR)

# Extract independent features (X) and dependent
feature (Y)

X = iris_data.data
Y = iris_data.target

# display the shapes of X and Y
print("shape of X:", X.shape)
print("shape of Y:", Y.shape)

from sklearn.model_selection import train_test_
split

X_train, X_test, Y_train, Y_train, Y_test = train_test_
split(X, Y, test_size = 0.33, random_
state = 42)

from sklearn.tree import DecisionTreeClassifier.

treemodel = DecisionTreeClassifier()

treemodel.fit(X_train, Y_train)
DecisionTreeClassifier()
```

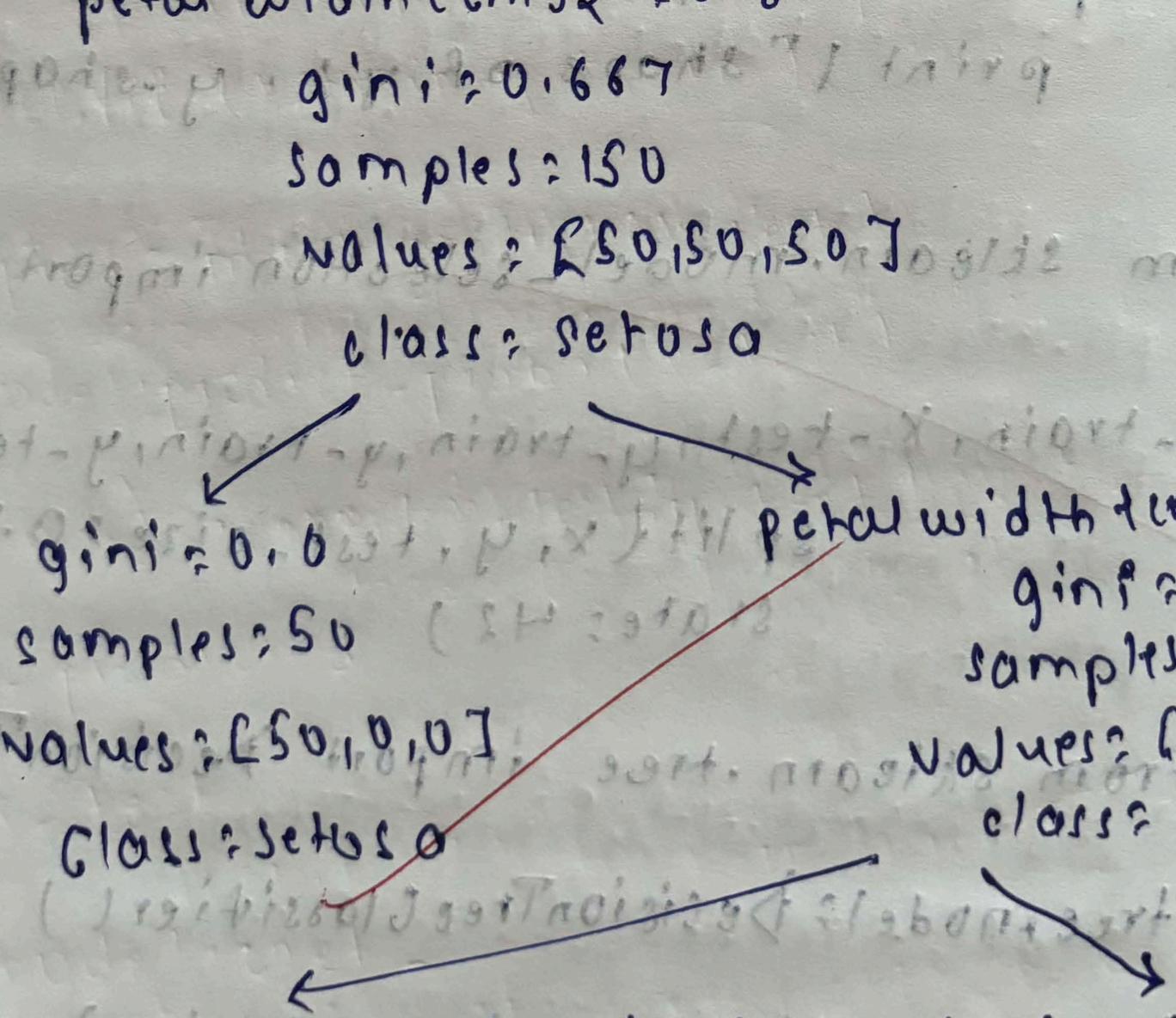
```
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt

# create a decision tree classifier
clf = DecisionTreeClassifier()

# Train the classifier on Iris Dataset
clf.fit(X,Y)

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

Output




```
from sklearn.
score = cross
accuracy
print ("
```

O/P mean

Que 25-4-22

```
from sklearn.model_selection import cross_val_score
score = cross_val_score(df, x, y, cv=5)
accuracy = score.mean()
print("Mean Accuracy:", accuracy)
```

O/P mean Accuracy: 0.9666

Que
25-4-22

Week-5 25/04/2024

Logistic Regression

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

df = pd.read_csv("insurance data.csv")
df.head()

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

from sklearn.model_selection import train_test_
split

x_train, x_test, y_train, y_test = train_test_split
(df[['age']], df.bough_insurance, train_size
= 0.8)

x_test

from sklearn.linear_model import LogisticRegression
model.fit(x_train, y_train)

y_pred = mod.predict(x_test)

model.predict_proba(x_test)

model.score(x_test, y_test)

y_predicted

x_test

model.coef_
```

→ 0.1475

```
model.intercept_
```

→ -5.5078847

```
import math
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 = 43
prediction_function(age)
→ 0.56856  # will buy insurance.
```

9-5-24

Regression

Week-06

k-Nearest Neighbour

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import numpy as np

data = pd.read_csv('spam_ham_dataset.csv')

data.head()

data.drop(['label', 'Unnamed: 0'], axis=1, inplace=True)

data.head()

x = data['text']
y = data['label_num']

vectorizer = TfidfVectorizer()
x = vectorizer.fit_transform(x)

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

k = 6
knn_classifier.fit(x_train, y_train)

y_pred = knn_classifier.predict(x_test)

accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

→ Accuracy: 0.9355

Week-07

SVM

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris

iris = load_iris()
data = pd.DataFrame(data=iris.data, columns=iris.
                    feature_names)
data['species'] = iris.target

x = data.drop('species', axis=1)
y = data['species']
print(data.head())

plt.scatter(x['sepal length (cm)'], x['sepal width
    (cm)'], c=y, cmap='viridis')
plt.xlabel('Sepal length (cm)')
plt.ylabel('Sepal width (cm)')
plt.title('Scatter Plot of Iris Dataset')
plt.show()

x_train, x_test, y_train, y_test = train_test_split
    (x, y, test_size=0.25, random_state=42)

svm_classifier = SVC(kernel='linear', random_state=
                     42)
svm_classifier.fit(x_train, y_train)
y_pred = svm_classifier.predict(x_test)
accuracy = accuracy_score(y_test, y_pred
print("Accuracy:", accuracy)
```

9.a.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.consemble import RandomForest
                                Classifier
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

iris_data= pd.read_csv('Iris.csv')
iris_data.head()

iris_data.drop('Id',inplace=True,axis=1)
x= iris_data.drop('species', axis=1)
y= iris_data['species']
iris_date.info()

x_train, X_test, Y_train, y_test= train_test_
        split(x, y, test_size=0.4, random_
                state=42)

rf_Classifier= RandomForestClassifier(n_
        estimator=100, random_state=42)

rf_classifier= RandomForestClassifier(
        n_estimators=100, random_
        state=42)

rf_classifier.fit(x_train, y_train)
```

Accuracy: 0.9731

9.b Ada

```
import
import
import
df = pd.
df.hea

y= df[

from
split,
y, te

from
tion(
dR
x_

from
```

pr

3.57

9.b **Adaboost**

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df = pd.read_csv("/content/iris.csv");
df.head()

y = df[['species']], axis = 1)

from sklearn.model selection import train_test_
split, X_train, X_test, Y_train, y_test = train(x,
y, test_size = 0.3, test_split = 0)

from sklearn.essemble import Adaboost classifica-
tion()
adb.product(X_test)
adb.product(X_train)

from sklearn.metrices import recovery
are_score = recovery.score(Y_pred, y_test)

print(recovery_score)
```

: 0.977

3.5

Artificial Neural Network

```
import numpy as np
X = np.array([[2,9],[1,5],[3,6]],type=float)
Y = np.array([[92],[86],[89]],type=float)
X = X/np.max(X,axis=0)
Y = Y/100

epoch=5000
lr=0.1
inputlayer_neurons=2
hiddenlayer=3
outputneurons=1

wh=np.random.uniform(size=(inputlayer_
    neurons,hidden_layer_neurons))

bh=np.random.uniform(size=(1,hiddenlayer))
wout=np.random.uniform(size=(1,output_
                neurons))

#sigmoid function
def sigmoid(x)
    return(1/1+np.expe x))
    Eo = y-output
outprod = derivations_sigmoid(output)
    Q_output = Eo * output.

EH = d_output_ c)t(wout.T)
```

```
hidden prod = derivation_sigmoid (hleryrec)
d_hiddenlayer = EH * hiddenprod
wout += hlayer_out, Tdot (d_output) *
print (Input = +str)
print ('Actual Output' + str(y))
print (' producted output', output)
```

actualoutput = [0.92] [0.86] [0.89]

predicted output = [0.86] [0.85] [0.867]

Lab 10

k-Means algorithm

Code:

```
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.cluster import KMeans
import pandas as p
import numpy as np

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']

model.fit(x)
plt.figure(fig size = (14,14)
colormap = np.array(['red','Iris','block'])

plt.subplot(2,2,1)
plt.scatter(x.petal_length, x.Petal_width c
  = colormap[y.Targets], s = 40)

plt.title('Reas Clusters')
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-Means Clustering')
plt.xlabel('Petal length')
plt.ylabel('Petal ~~length~~ width')
plt.show()
```

Principle Component Analysis

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

from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()
cancer.keys()

df = pd.Dataframe(cancer['data'], columns=cancer[
        'feature_names'])

from sklearn.preprocessing import standardscaler
scaler = StandardScalar()
scalar.fit(df)
scalar_data = scaler.transform(df)

from sklearn.decompositor import PCA
pca = PCA(n_components=2)
pca.jet(scalar_data)
x_pro = pro.transform(scalare_data)

Scarled_doter.shape
x_pur.shape
pet.figure(figsize=(8,6))

plt.scatter(x_pro[:,0], x_peer[:,1], c=cancer
    ['target'], comap='plasma')
```

```
plt.xlabel(
plt.ylabel(
```

Principle Component Analysis

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

from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()
cancer.keys()

df = pd.Dataframe(cancer['data'], columns=cancer[
        'feature_names'])

from sklearn.preprocessing import standardscaler
scaler = StandardScalar()
scalar.fit(df)
scalar_data = scaler.transform(df)

from sklearn.decompositor import PCA
pca = PCA(n_components=2)
pca.jet(scalar_data)
x_pro = pro.transform(scalare_data)

Scarled_doter.shape
x_pur.shape
pet.figure(figsize=(8,6))

plt.scatter(x_pro[:,0], x_peer[:,1], c=cancer
    ['target'], comap='plasma')
```

```
plt.xlabel(
plt.ylabel(
```