INDEX

NAME: Shreesha H Shetty STD.: 6 SEC.: D ROLL NO.: 209 SUB.: Machine Learning Lab

S. No.	Date	Title	Page No.	Mark	Teacher's Sign / Remarks
1	21-03-24	Import & Export data using Pandas library function	1	10	21/3/24
2	28-03-24	Demonstrate various data preprocessing techniques of dataset	2	10	28/3/24
3.	04-04-24	Implement linear and Multi linear regression algorithm using appropriate dataset.	7	10	11-4-24
4	18-04-24	Use an appropriate data set for building the decision tree (ID3) and apply this knowledge to classify a new sample.	11	10	25-4-24
5	25-04-24	Build logistic Regression model for a given dataset.	14	10	9/5
6	09-05-2024	Build KNN Classification and SVM model for given dataset.	18	10	23-5-24
7	23-05-2024	Build Artificial Neural Network model with back propagation a given dataset.	20	10	1-7
8	23-05-2024	a) Implement Random forest ensemble method on a given dataset. b) Implement Boosting ensemble method on a given dataset.	22	10	30/7

DYC0623301 C

S. No.	Date	Title	Page No.	Teacher's Sign / Remarks
9	30-05-2024	Build K-Means algorithm to cluster a set of data stored in a .csv file	25 — 40	
10	30-05-2024	Implement Dimensionality reduction using PCA method	26 — 40	

21/03/24

Lab-1

1. Write a python program to import and export data using Pandas library functions.

→

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

url = "https://archive.ics.uci.edu/ml/machine-learning_databas
      /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()
iris_data.to_csv("cleaned_iris_data.csv")
```

Output:

①

	id	name	host_id	host_name	neighbourhood	price
0	2265	Zen-East in the..	2466	Paddy	78702	179
1	5245	Eco friendly,...	2466	Paddy	78702	114

②

	sepal_length_in_cm	sepal_width_in_cm	petal length_in cm	petal width_in_cm
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2

class

iris-setosa

iris-setosa

21/3/24

28/03/2024

Lab-2

2 Demonstrate various data pre-processing techniques for a given dataset.

→

2 Get the Data:

Download the Data:

```
import os
import tarfile
import urllib

DOWNLOAD_ROOT = "https://raw.githubusercontent.com/
                 ageron/handson-ml2/master/"
HOUSING_PATH = os.path.join("data", "os")
HOUSING_URL = DOWNLOAD_ROOT + "datasets/housing/housing.
```

```
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_p
    housing_tgz = tarfile.open(name = tgz_path)
    housing_tgz.extractall(path = housing_path)
    housing_tgz.close()

fetch_housing_data()

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)
```

classmate
Date
Page

Quick look at the Data :

```
housing = load_housing_data()

housing.head()

housing.info()

housing.describe()

import matplotlib.pyplot as plt
import seaborn as sns

housing.hist(bins=50, figsize=(20,15))
plt.show()
```

Create a Test Set :

```
import numpy as np

def split_train_test(data, test_ratio=0.2):
    shuffled_indices = np.random.permutation(len(data))
    test_set_size = int(len(data) * test_ratio)
    test_indices = shuffled_indices[:test_set_size]
    train_indices = shuffled_indices[test_set_size:]
    return data.iloc[train_indices], data.iloc[test_indices]

train_set, test_set = split_train_test(data=housing)
len(train_set), len(test_set)
```

3 Discover & Visualize the Data to Gain Insights

```
strat_train_set.shape , strat_test_set.shape
```

Visualizing Geographical Data

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

Looking for Correlations

```
corr_matrix = housing.corr()

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

from pandas.plotting import scatter_matrix

attributes = ['median_house_value', 'median_income', 'total_room
                                    'housing_median_age']

scatter_matrix (frame = housing [attributes], figsize = (12, 8))
plt.show()

housing ['rooms_per_household'] = housing ['total_rooms']/housing
                                  ['households']
```

4 Prepare the Data for Machine Learning Algorithms

Data Cleaning

```
from sklearn.impute import SimpleImputer

imputer = SimpleImputer (strategy = 'median')

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

imputer.fit (housing_num)
```

```
imputer.statistics_

housing_num.median().values

Handling Text & Categorical Attributes
from sklearn.preprocessing import OrdinalEncoder

ordinal_encoder = OrdinalEncoder()

ordinal_encoder.categories_
```

5 Select & Train a Model

Training & Evaluating on the Training Set

```
from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()

lin_reg.fit(X = housing_prepared , y = housing_labels)

from sklearn.metrics import mean_squared_error

housing_predictions = lin_reg.predict(housing_prepared)

lin_mse = mean_squared_error(housing_labels, housing_predictions)

lin_rmse = np.sqrt(lin_mse)
lin_rmse
```

```
from sklearn.tree import DecisionTreeRegressor

tree_reg = DecisionTreeRegressor()

tree_reg.fit(X = housing_prepared, y = housing_labels)
```

Better Evaluation using Cross-validation

```
from sklearn.model_selection import cross_val_score

scores = cross_val_score(estimator = tree_reg, X = housing_
         prepared, y = housing_labels, scoring = 'neg_mean_
         squared_error', cv = 10)

tree_rmse_scores = np.sqrt(-scores)

from sklearn.ensemble import RandomForestRegressor

forest_reg = RandomForestRegressor()

forest_reg.fit(X = housing_prepared, y = housing_labels)
```

6 Fine-Tune Your Model

```
grid_search.best_params_

grid_search.best_estimator_

cat_encoder = full_pipeline.named_transformers_['cat']

final_model = grid_search.best_estimator_

X_test_prepared = full_pipeline.transform(X = X_test)
final_rmse = np.sqrt(final_mse)
final_rmse
```

7 Launch, Monitor & Maintain your System

04-04-24

Lab-4

3. Implement Linear and Multi Linear Regression algorithm 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['YearsExperience'], df_sal['Salary'],
            color='lightcoral')

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 = regressor.predict(X_train)

plt.scatter(X_train, y_train, color='lightcoral')
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.legend(['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='lightcoral')
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_}')
```

Multiple 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,
                                    test_size = 0.2, random_state = 0)

regressor = LinearRegression()
regressor.fit (X_train, y_train)

y_pred = regressor.predict (X_test)

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

18-04-2024

Lab-3

4. Use an appropriate data set for building the decision tree (ID3) and apply this knowledge to classify a new sample.

→

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

from sklearn.de
iris_data = load_iris()

print(iris_data.descr)

features
X = iris_data.data
y = iris_data.target

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_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()
```

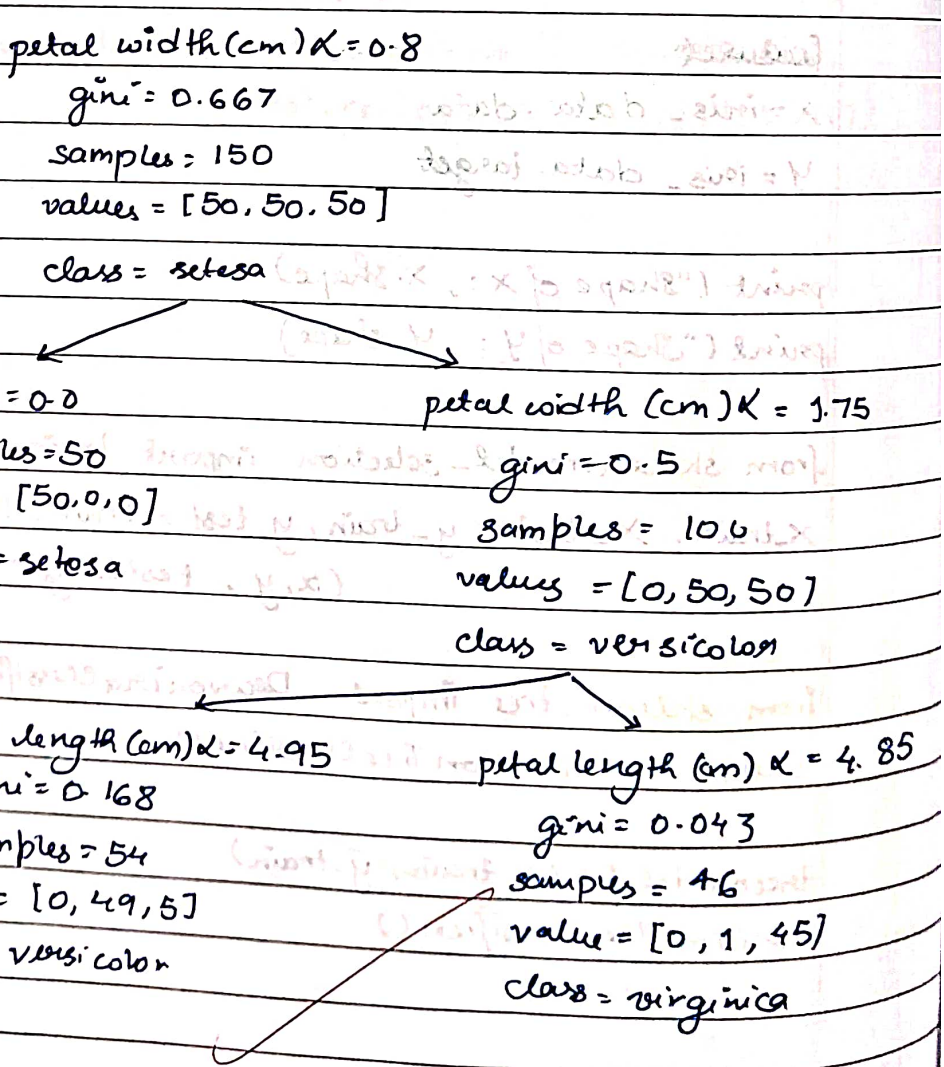
```
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt

clf = DecisionTreeClassifier()

clf.fit(X, y)

plt.figure(figsize = (12,8))
plot_tree(clf, filled = True), feature_name = iris_data.
  feature_name, class_names = iris_data.target_names)
plt.show()
```

Output:



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

Output :

Mean Accuracy : 0.9666

25-04-2024

Lab-5

5 Build Logistic Regression model for a given dataset.

→

```
import pandas pas pd
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

df = pd.read_csv("insurance_data.csv")
print (df.head(1)

X_train, X_test, y_train, y_test = Train_test_split (df [['age']]
        df.bought_insurance, test_size = 0.2
print ("X_test:")
print (X_Test)

model = Logistic Regression()
model.fit (X_train, y_train)
y_pred = model.predict (X_test)
print (y_pred)
print (model.predict_proba(X_test))
print(model.score (X_test, y_test))
```

Output:

Accuracy = 0.5

```
import math
def sigmoid (z)
    return 1 [(1+Math.exp(-z))

def predi (age):
        z = 0.042 * age - 1.53
        y = sigmoid (z)
        returny
```

print(predi(35))
print(predi(43))

Output:

Prediction: array([1,0,1,0,0,0,0,1,0,0])

Score: 0.8888

Linear Reg score: 0.584321

Predictions: 0.485

0.5685

04/05/2024

Lab-6 & 7

Build KNN classification and SVM model for a given data set

→

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

df = pd.read_csv("Iris.csv")
df.head()

classes = df["Species"].unique()
colors = ['r', 'g', 'b']

for i, cls in enumerate(classes):
    class_data = df[df["Species"] == cls]
    plt.scatter(class_data["SepalLengthCm"], class_data["PetalLen
                c=colors[i], label=cls)

plt.xlabel('Sepal length [cm]')
plt.ylabel('Petal length [cm]')
plt.legend()
plt.show()

for i, cls in enumerate(classes):
    class_data = df[df["Species"] == cls]
    plt.scatter(class_data["SepalWidthCm"], class_data["PetalWi
                c=colors[i], label=cls)

plt.xlabel('Sepal width [cm]')
plt.ylabel('Petal width [cm]')
plt.legend()
plt.show()
```

```
y = df["Species"]
x = df.drop(["Species"], axis=1)

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, y, train_size=0.7, test_size=0.3, random_state=0)

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
y2=knn.predict(X_train)

from sklearn.metrics import accuracy_score
score = accuracy_score(y_pred, y_test)
print(f"Training Accuracy : {score}")
score2 = accuracy_score(y2, y_train)
print(f"Training Accuracy : {score2}")
```

Output:

Training Accuracy : 1.0
Training Accuracy : 1.0

```
from sklearn.svm import SVC
model = SVC(kernel = 'linear', random_state = 0, C=1.0)
model.fit(X_train, y_train)
y2 = model.predict(X_train)

from sklearn.metrics import accuracy_score
score = accuracy_score(y_pred, y_test)
print(f"Testing Accuracy : {score}")
score2 = accuracy_score(y2, y_train)
print(f"Training Accuracy : {score2}")
```

Output: Training Accuracy : 1.0
Training Accuracy : 1.0

Lab - 8

Build Artificial Neural Network model with back propagation on a given dataset.

→

```
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,
                               hiddenlayer_neurons))
bh = np.random.uniform(size = (1, hiddenlayer_neurons))
wout = np.random.uniform(size = (hiddenlayer_neurons,
                                 output_neurons))
bout = bh = 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):
    hinpl = np.dot(X, wh)
    hinp = hinpl + bh
    hlayer_act = sigmoid(hinp)
    outinp1 = np.dot(hlayer_act, wout)
```

```
outinp = outinp1 + bout
output = sigmoid(outinp)
EO = y - output
outgrad = derivatives_sigmoid(output)
d_output = EO * outgrad
EH = d_output.dot(wout.T)

hiddengrad = derivatives_sigmoid(hlayer_act)
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)
```

Output:

```
Input:  [[0.666667  1.      ]
         [0.333333  0.55556 ]
         [1.        0.666667]]

Actual output:  [[0.92]
                 [0.86]
                 [0.89]]

Predicted output:  [[0.89130561]
                    [0.88519821]
                    [0.88962179]]
```

Lab-9

9 a) Implement Random forest ensemble method on a given dataset.
b) Implement Boosting ensemble method on a given dataset.

a) →

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

df = pd.read_csv("Iris.csv")
df.head()

y = df["Species"]
X = df.drop(["Species"], axis=1)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                   test_size = 0.3, random_state=
```

```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(n_estimators = 100)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
y2 = clf.predict(X_train)
```

```
from sklearn.metrics import accuracy_score
score = accuracy_score(y_pred, y_test)
print(f"Testing Accuracy : {score}")
score2 = accuracy_score(y2, y_train)
print(f"Training Accuracy : {score2}")
```

Output:

Testing Accuracy : 1.0
Training Accuracy : 1.0

classmate
Date
Page

~~from sklearn.linear_model import LogisticRegression~~
~~lr_model = LogisticRegression()~~
~~adbhp AdaBoo~~

b) →
```
from sklearn.ensemble import AdaBoostClassifier
adb = AdaBoostClassifier()
adb_model = adb.fit(X_train, y_train)
y_pred = adb.predict(X_test)
y2 = adb.predict(X_train)

from sklearn.metrics import accuracy_score
score = accuracy_score(y_pred, y_test)
print(f"Testing Accuracy : {score}")
score2 = accuracy_score(y2, y_train)
print(f"Training Accuracy: {score2}")
```

Output: Testing Accuracy : 0.977777
Training Accuracy : 1.0

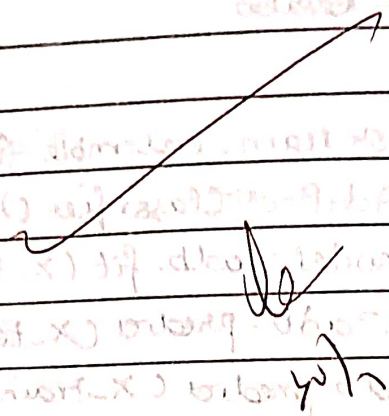
```
from sklearn.linear_model import LogisticRegression
lrmodel = LogisticRegression()
adbhp = AdaBoostClassifier(n_estimators = 150, estimator = lrmodel,
                           learning_rate = 1)
model = adbhp.fit(X_train, y_train)
y_pred = model.predict(X_test)
y2 = model.predict(X_train)

from sklearn.metrics import accuracy_score
score = accuracy_score(y_pred, y_test)
print(f"Testing Accuracy: {score}")
score2 = accuracy_score(y2, y_train)
print(f"Training Accuracy: {score2}")
```

Output :

Testing Accuracy: 1.0

Training Accuracy: 1.0



Lab-10

10 Build k-Means algorithm to cluster a set of data stored in .csv file.

→

```
import matplotlib.pyplot as plt
from sklearn import dataset
from sklearn.cluster import KMeans
import pandas as pd
import numpy as np

iris = dataset.load_iris()
x = pd.DataFrame(iris.data, columns=['Sepal_length',
    'Sepal_width', 'petal_length', 'petal_width'])
y = pd.DataFrame(iris.target, columns=['Targets'])

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['PetalWidth'],
            c=colormap[y['Targets']], s=40)
plt.title('Real 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 Width')
plt.show()
```

Lab-11a

11 Implement Dimensionality reduction using PCA Method

11b Build k-means algorithm to cluster a set of data stored in .csv file.

→
```
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 = StandardScaler()
scaler.fit(df)
scaled_data = scaler.transform(df)

from sk.learn.decomposition import PCA
pca = PCA(n_components = 2)
pca.fit(scaled_data)
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')
```