1BM21CS253

INDEX

NAME: Yashaswini G. A STD.: VI SEC.: 'D' ROLL NO.: ______ SUB.: ML LAB

S. No.	Date	Title	Page No.	Teacher's Sign / Remarks
01	21/03/2024	Importing and exporting.	10	21/3/24
02	28/03/2024	End-to-End project	10	28/3/24
03	4/04/2024	Linear Regression (simple & Multiple)	9	18/4/24
04	18/04/2024	Decision tree (ID3)	10	
05	25/04/2024	Logistic Regression	10	4/5
06	9/05/2024	KNN ~~and~~	10	9/5
07	9/05/2024	SVM	10	9/5
08	23/05/2024	ANN	10	
09	23/05/2024	a) Random Forest b) AdaBoost		23/5/24
10	30/05/2024	K-Means clustering	10	30/5/24
11	30/05/2024	Principal Component Analysis.	10	30/5/24

21/03/2024

WEEK - 1

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

```
import pandas as pd
url = "https://archive.ics.uci.edu/ml/machine-learning-
        databases/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()
```

OUTPUT

	sepal_length_in cm	sepal_width_in_cm	~~sepal~~ pet al_length_in_cm	petal_width -in_cm	class
0	5.1	3.5	1.4	0.2	Iris-setosa

```
iris_data.to_csv("cleaned_iris_data.csv")
df2 = pd.read_csv("cleaned_iris_data.csv")
```

	sepal_length_in cm	sepal_width_in cm	petal_length_ in_cm	petal_width -in_cm	class
0	5.1	3.5	1.4	0.2	Iris setosa

21/3/24

28/03/2024

End-to-End Machine Learning Project

1. Select a Performance Measure

2. Get the Data

```
HOUSING_PATH = "https://raw.githubusercontent.com/housing.csv"
housing = pd.read_csv(HOUSING_PATH)
housing.head()
housing.info()
housing['ocean_proximity'].value_counts()
housing.describe()
housing.groupby(by=['longitude', 'latitude']).count()
            ['total_rooms'].sort_values()

from sklearn.model_selection import train_test_split
train_set, test_set = train_test_split(housing, test_size=0.2,
                    random_state=42)

housing['income_cat'] = pd.cut(x=housing['median_income'],
 bins=[0, 1.5, 3, 4.5, 6, np.inf], labels=[1, 2, 3, 4, 5]

housing['income_cat'].hist()
```

3. Discover and Visualize the Data to Gain Insights

Plot

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

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

Correlation

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

scatter_matrix() for correlation

```
from pandas.plotting import scatter_matrix
attributes = ['median_house_value', 'median_income',
    'total_rooms', 'housing_median_age']

scatter_matrix(frame=housing[attributes], figsize=(12, 8))
```

Experimenting with attribute combinations

```
housing ['rooms_per_household'] = housing ['total_rooms'] /
                                  housing ['households']
housing['bedrooms_per_room'] = housing ['total_bedrooms'] /
                               housing ['total_rooms']
housing ['population_per_household'], ~~sort_values (ascending~~
    = housing['population'] /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
```

One Hot Encoding

```
from sklearn.preprocessing import OneHotEncoder
one_hot_encoder = OneHotEncoder()
housing_cat_1hot = one_hot_encoder.fit_tranform(housing_cat.
                                                 values)
housing_cat_1hot
housing_cat_1hot.toarray()
one_hot_encoder.categories_
```

Transformation pipelines

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import standard scaler
num_pipeline = Pipeline([('imputer', SimpleImputer
        (strategy='median')), ('attribs_adder',
    combinedAttributesAdder()), ('std_scaler', StandardScaler())
])
housing_num_tr = num_pipeline.fit_tranform(housing_num)
housing_num_tr.shape
```

20/3/24

04/04/2024

5. Select and Train a model

```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X=housing_prepared, y=housing_labels)
```

Try to modelling on a few instances from the training set:

```
some_data = housing.iloc[:5]
some_labels = housing_labels.iloc[:5]
some_data_prepared = full_pipeline.transform(some_data)
print("Predictions:", lin_reg.predict(some_data_prepared))
print("Labels:", some_labels.tolist())
```

Measuring performance using RMSE metric.

```
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
```

6. Fine-Tune Your Model

Grid search

Mentioning hyper-parameters and values to test and it will test out all combinations of hyper-parameters and are cross-validation for evaluation.

```
from sklearn.model_selection import GridSearchCV
param_grid = [ { 'n_estimators': [3,10,30], 'max_features': [2,4,6,8]},
{'bootstrap': [False], 'n_estimators': [3,10], 'max_features':
    [2,3,4]} ]

forest_reg = RandomForestRegressor()
grid_search = GridSearchCV(estimator = forest_reg,
    param_grid = param_grid, scoring = 'neg-mean_squared_error',
 cv = 5, return_train_score = True, n_jobs = -1)

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

grid_search.best_params_
grid_search.best_estimator_
cvres = grid_search.cv_results_

for mean_score, params in zip(cvres['mean_test_score'],
  cvres['params']):
print(np.sqrt(mean_score), params)
```

7. Launch, Monitor, & Maintain your system

4/4/2024

Simple Linear Regression

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborne as sns
from sklearn.model_selection import train_test_split
from pandas.core.common import random_state
from sklearn.linear_model import linear Regression

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()
```

Split data

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

Split into Train test split sets

```
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 results

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

Visualize predictions

```
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.scatter title ('Salary vs Experience (Test set)')
plt.xlabel ('Years of Experience')
plt.ylabel ('Salary')
plt.show()
```

Coefficient and Intercept

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

Coefficient : [[9312.57]]
Intercept : [26780.099]

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.linear_model import LinearRegression


df_start = pd.read_csv('/contontent/startup.csv')
df_start.head()

df_start.describe()
```

Distribution

```
plt.title('Profit Distribution Plot')
sns.distplot(df_start['Profit'])
plt.show()
```

Relation between Profit and R&D Spend

```
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()
```

Split into Independent/ Dependent variables

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

One-hot encoding

```
ct = ColumnTransformer(transformers= [('encoder', OneHotEncoder(),
     [3])], remainder = 'passthrough')
x = np.array(ct.fit_transform(x))
```

Split into Train/Test sets.

```
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 results

```
y_pred = regressor.predict(x_test)
```

Compare predictions

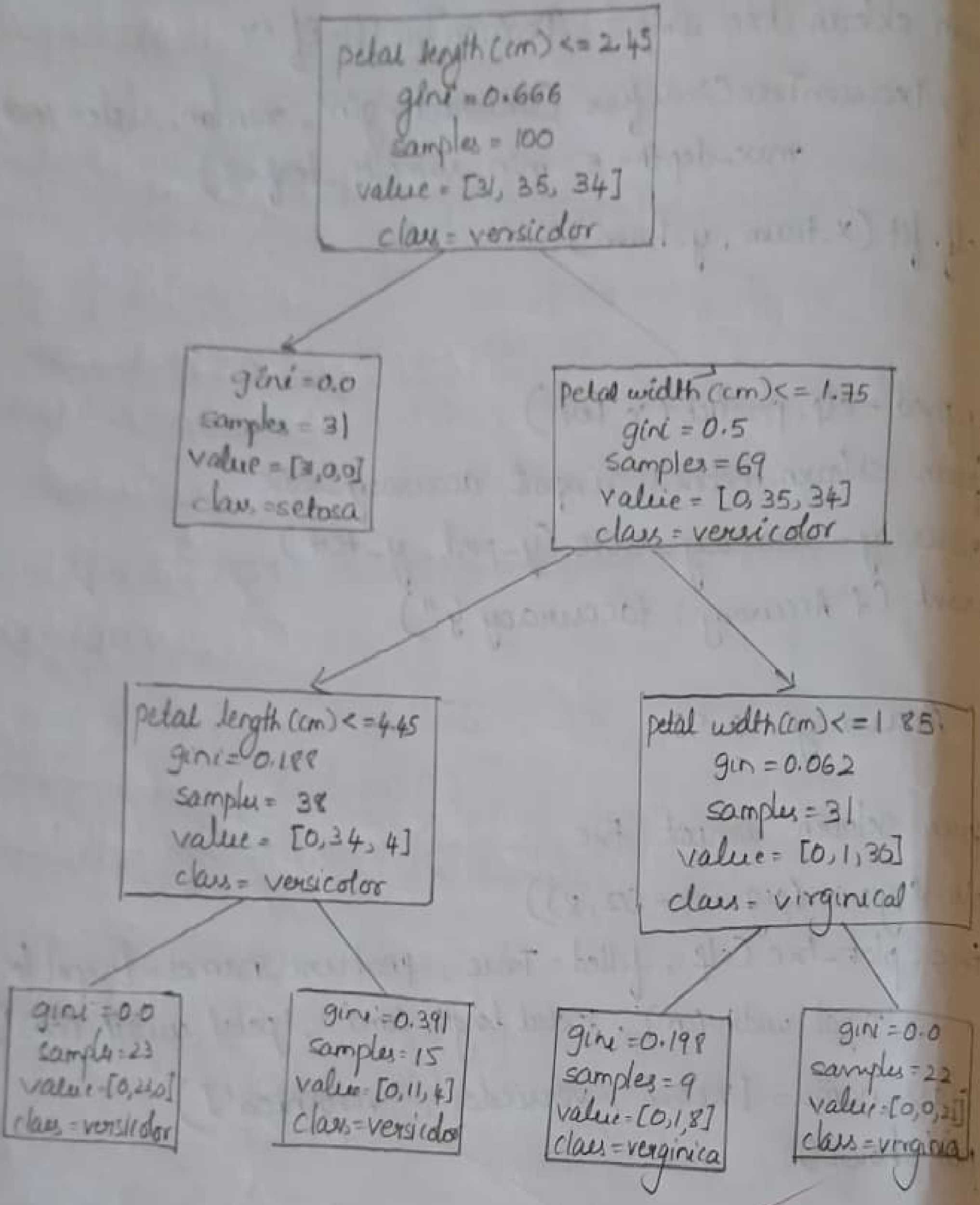
```
np.set_printoptions(precision = 2)
result = np.concatenate((y_pred.reshape(len(y_pred), 1),
         y_test.reshape(len(y_test, 1)), 1)
result
```

18/04/2024

Decision tree (ID3)

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

~~df = pd.read_csv("/content/drive/MyDrive/Iris.csv")~~
~~df.head()~~

from sklearn import datasets
iris = datasets.load_iris()
iris_df = pd.DataFrame(data=iris.data, columns=iris.
                                feature_names)

iris_df['species'] = iris.target
iris_df['species'] = iris_df['species'].map({0: 'setosa', 1:
                 'versicolor', 2: 'virginica'})

iris_df.head()

y = iris_df["species"]
x = iris_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=0.33, random_state=42)
```

```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier (criterion='gini', random_state=100,
          max_depth=5, min_samples_leaf=8)
clf.fit (x_train, y_train)


y_pred = clf.predict (x_test)
from sklearn.metrics import accuracy_score
accuracy = accuracy_score (y_pred, y_test)
print (f"Accuracy: {accuracy}")

  Accuracy : 0.98

from sklearn import tree
plt.figure (figsize = (12, 8))
tree.plot_tree (clf, filled = True, feature_names= ['sepal length (cm)'
     , 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)'],
 class_names= ['setosa', 'versicolor', 'virginica'])
plt.show()
```


petal length (cm) <= 2.45
gini = 0.666
samples = 100
value = [31, 35, 34]
class = versicolor
gini = 0.0
samples = 31
value = [31, 0, 0]
class = setosa
petal width (cm) <= 1.75
gini = 0.5
samples = 69
value = [0, 35, 34]
class = versicolor
petal length (cm) <= 4.45
gini = 0.188
samples = 38
value = [0, 34, 4]
class = versicolor
petal width (cm) <= 1.85
gini = 0.062
samples = 31
value = [0, 1, 30]
class = virginica
gini = 0.0
samples = 23
value = [0, 23, 0]
class = versicolor
gini = 0.391
samples = 15
value = [0, 11, 4]
class = versicolor
gini = 0.198
samples = 9
value = [0, 1, 8]
class = virginica
gini = 0.0
samples = 22
value = [0, 0, 22]
class = virginica

18/4/24

~~[illegible]~~
25/04/2024

Insurance System

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

df= pd.read_csv('id.csv')
x_train, x test, y_train, ytest = train_test.split(df['age']],
      df.bought_insurance, test_size=0.2)

model = logisticRegression()
model.fit (x_train, y_train)
y-predicted = model.predict (x_test)
print (y-predicted)

print (model.predict_prob (x_test))
print (model.score (x_test, y_test))
```

```
import Math
def sigmoid(z):
    return 1/(1+ Math.exp(-z))

def predi(age):
    z = 0.042 * age - 1.53
    y = sigmoid(z)
    retur y.
print (predi (35))
print (predi(43))
```

Output

Prediction: array ([1, 0, 1, 0, 0, 0, 0, 1, 0])
Score : 0.8888
Linear Reg score : 0.584321
Predictions : 0.485
0.5685.

Lva
9-5-2024

09/08/2024

Program 6
KNN - K Nearest Neighbour

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
df = pd.read_csv("/content/drive/My Drive/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"], classdata["PetalLengthCm"], c=colors[i], label=cls)

    plt.xlabel('Sepal length [cm]')
    plt.ylabel('Petal length [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, 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"Testing Accuracy: {score}")
score2 = accuracy_score(y2, y_train)
print(f"Training Accuracy: {score2}")
```

Testing Accuracy: 1.0
Training Accuracy: 1.0

Program 7

SVM - Support Vector Machine

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

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

Testing Accuracy: 1.0
Training Accuracy: 1.0

9-5-24

23/05/2024

Program 8

Implementation of ANN using Back propogation for given values

```
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.array(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-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)
    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:
[[0.6667  1.    ]
 [0.3334  0.556 ]
 [1.      0.6667]]
Actual output:
[[0.92]
 [0.86]
 [0.89]]
Predicted Output:
 [[0.935]
  [0.923]
  [0.9339]]
```

Program 9a

Random Forest Algorithm

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

df = pd.read_csv("/content/drive/MyDrive/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=0)
```

Random

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

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

Output

Accuracy: 1.0.

AdaBoost (with default parameters)

```
from sklearn.ensemble import AdaBoostClassifier.
adb= AdaBoostClassifier()
adb_model = adb.fit (x_train, y_train)
y-pred = adb_model. predict (x_test)

score = accuracy_score(y-pred, y-test)
print (f"Accuracy : {score}")
```

Accuracy : 0.977

AdaBoost (With Hyper Parameters)

```
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)

score = accuracy_score(y-pred, y-test)
print (f"Accuracy : {score}")
```

OUTPUT

Accuracy : 1.0.

23/05/24

30/05/2024

Lab program - 10

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

```
import matplotlib.pyplot 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.columns = ['Sepal_Length', 'Sepal_width', 'Petal_Length',
             'Petal_Width']
y = pd.DataFrame (iris.target)
y.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.Petal_Width, 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')
```

5/5

Lab program - 11

Implement Dimensionality reduction using Principle Component Analysis (PCA) method

```
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()

print(cancer['DESCR'])

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

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(df)

scaled_data= scaler.transform(df)

from sklearn.decomposition import PCA
pca= PCA(n_components =2)
pca.fit(scaled_data)

~~x_pca~~ x_pca= pca.transform(scaled_data)
scaled_data.shape
(569, 30)
```

```
x_pca.shape
(569, 2)

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')
```

3/3/2