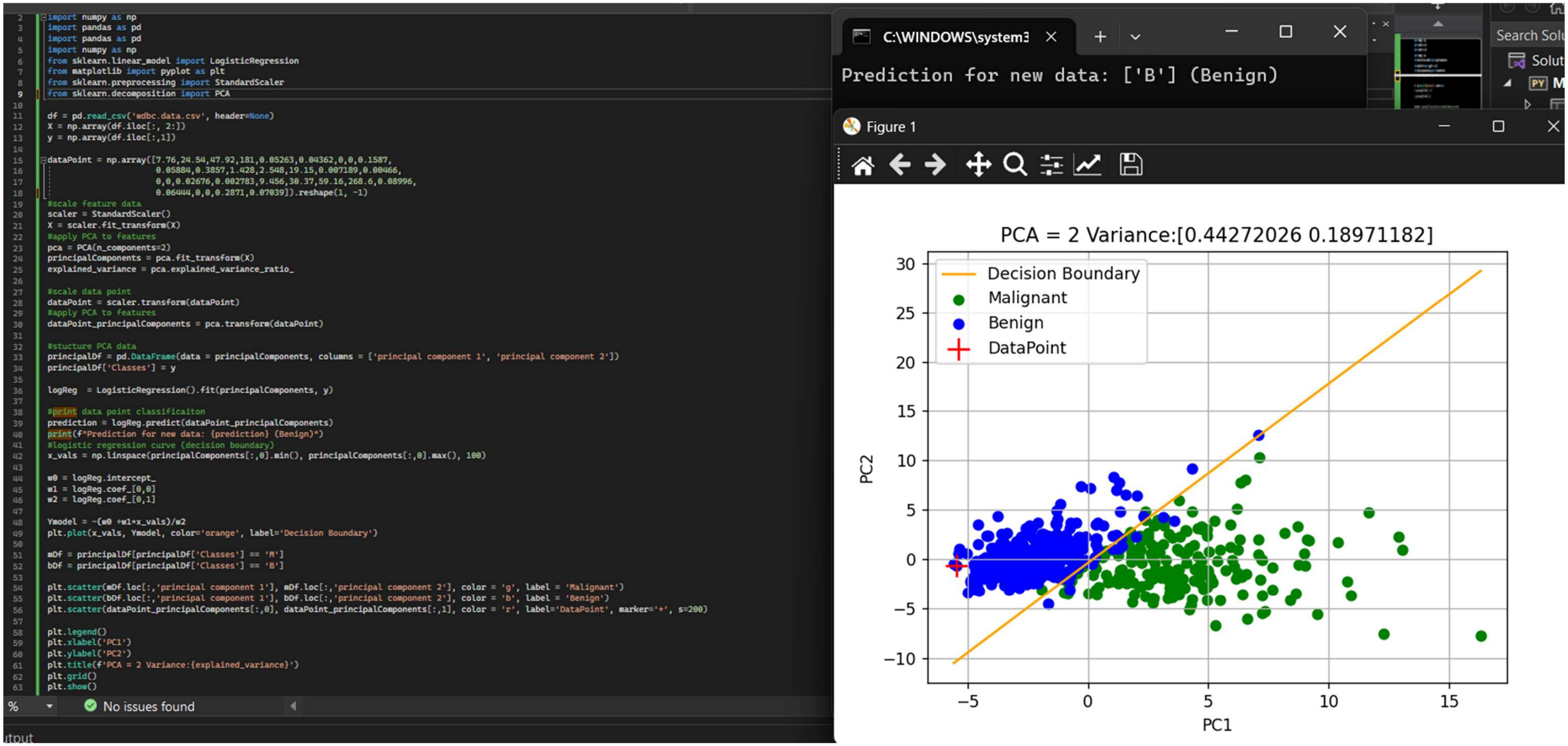
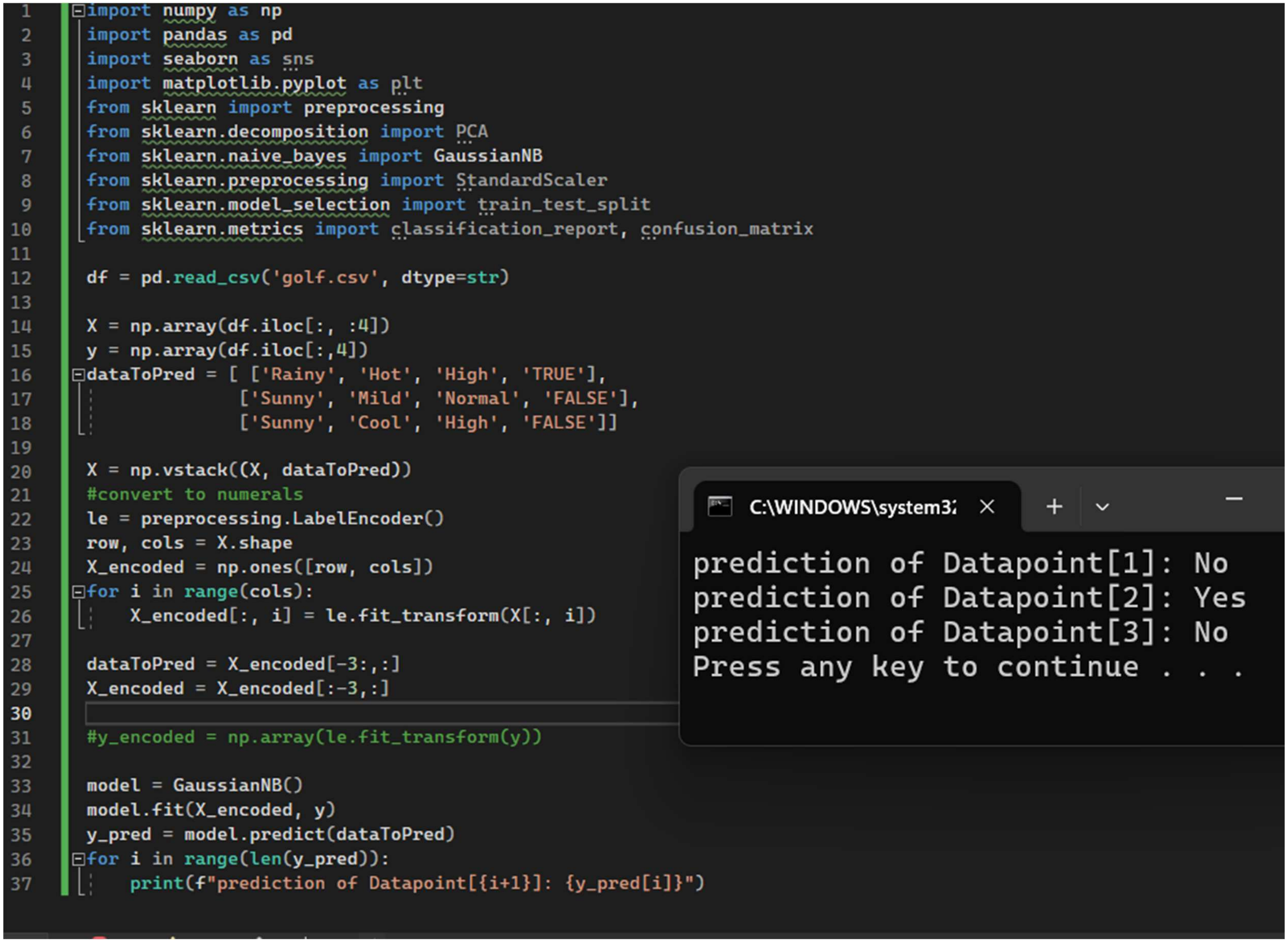
1.



```python
import numpy as np
import pandas as pd
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from matplotlib import pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

df = pd.read_csv('wdbc.data.csv', header=None)
X = np.array(df.iloc[:, 2:])
y = np.array(df.iloc[:,1])

dataPoint = np.array([7.76,24.54,47.92,181,0.05263,0.04362,0,0,0.1587,
                      0.05884,0.3857,1.428,2.548,19.15,0.007189,0.00466,
                      0,0,0.02676,0.002783,9.456,30.37,59.16,268.6,0.08996,
                      0.06444,0,0,0.2871,0.07039]).reshape(1, -1)
#scale feature data
scaler = StandardScaler()
X = scaler.fit_transform(X)
#apply PCA to features
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(X)
explained_variance = pca.explained_variance_ratio_

#scale data point
dataPoint = scaler.transform(dataPoint)
#apply PCA to features
dataPoint_principalComponents = pca.transform(dataPoint)

#stucture PCA data
principalDf = pd.DataFrame(data = principalComponents, columns = ['principal component 1', 'principal component 2'])
principalDf['Classes'] = y

logReg  = LogisticRegression().fit(principalComponents, y)

#print data point classificaiton
prediction = logReg.predict(dataPoint_principalComponents)
print(f"Prediction for new data: {prediction} (Benign)")
#logistic regression curve (decision boundary)
x_vals = np.linspace(principalComponents[:,0].min(), principalComponents[:,0].max(), 100)

w0 = logReg.intercept_
w1 = logReg.coef_[0,0]
w2 = logReg.coef_[0,1]

Ymodel = -(w0 +w1*x_vals)/w2
plt.plot(x_vals, Ymodel, color='orange', label='Decision Boundary')

mDf = principalDf[principalDf['Classes'] == 'M']
bDf = principalDf[principalDf['Classes'] == 'B']

plt.scatter(mDf.loc[:,'principal component 1'], mDf.loc[:,'principal component 2'], color = 'g', label = 'Malignant')
plt.scatter(bDf.loc[:,'principal component 1'], bDf.loc[:,'principal component 2'], color = 'b', label = 'Benign')
plt.scatter(dataPoint_principalComponents[:,0], dataPoint_principalComponents[:,1], color = 'r', label='DataPoint', marker='+', s=200)

plt.legend()
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.title(f'PCA = 2 Variance:{explained_variance}')
plt.grid()
plt.show()
```

```
Prediction for new data: ['B'] (Benign)
```

2.



```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.decomposition import PCA
from sklearn.naive_bayes import GaussianNB
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix

df = pd.read_csv('golf.csv', dtype=str)

X = np.array(df.iloc[:, :4])
y = np.array(df.iloc[:,4])
dataToPred = [ ['Rainy', 'Hot', 'High', 'TRUE'],
              ['Sunny', 'Mild', 'Normal', 'FALSE'],
              ['Sunny', 'Cool', 'High', 'FALSE']]

X = np.vstack((X, dataToPred))
#convert to numerals
le = preprocessing.LabelEncoder()
row, cols = X.shape
X_encoded = np.ones([row, cols])
for i in range(cols):
    X_encoded[:, i] = le.fit_transform(X[:, i])

dataToPred = X_encoded[-3:,:]
X_encoded = X_encoded[:-3,:]

#y_encoded = np.array(le.fit_transform(y))

model = GaussianNB()
model.fit(X_encoded, y)
y_pred = model.predict(dataToPred)
for i in range(len(y_pred)):
    print(f"prediction of Datapoint[{i+1}]: {y_pred[i]}")
```

```
prediction of Datapoint[1]: No
prediction of Datapoint[2]: Yes
prediction of Datapoint[3]: No
Press any key to continue . . .
```