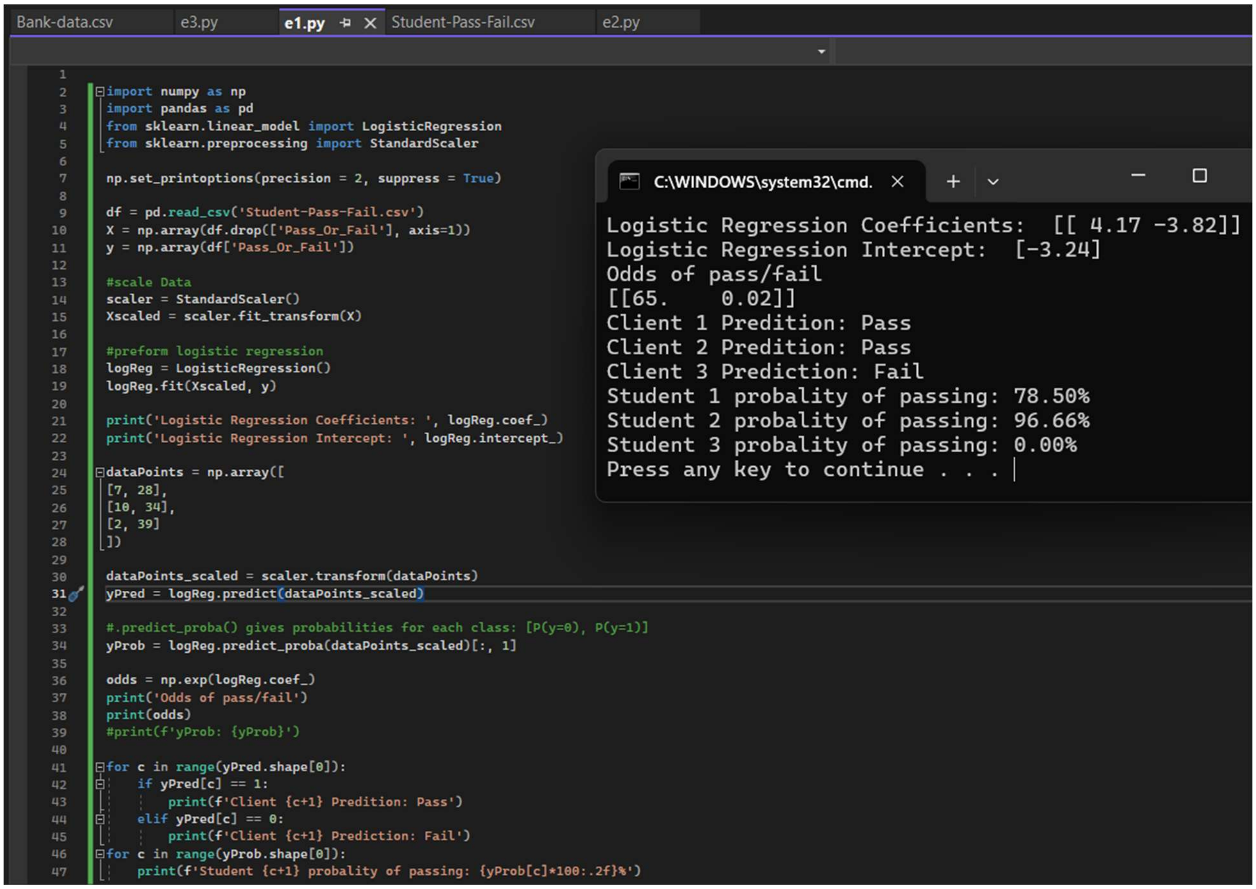
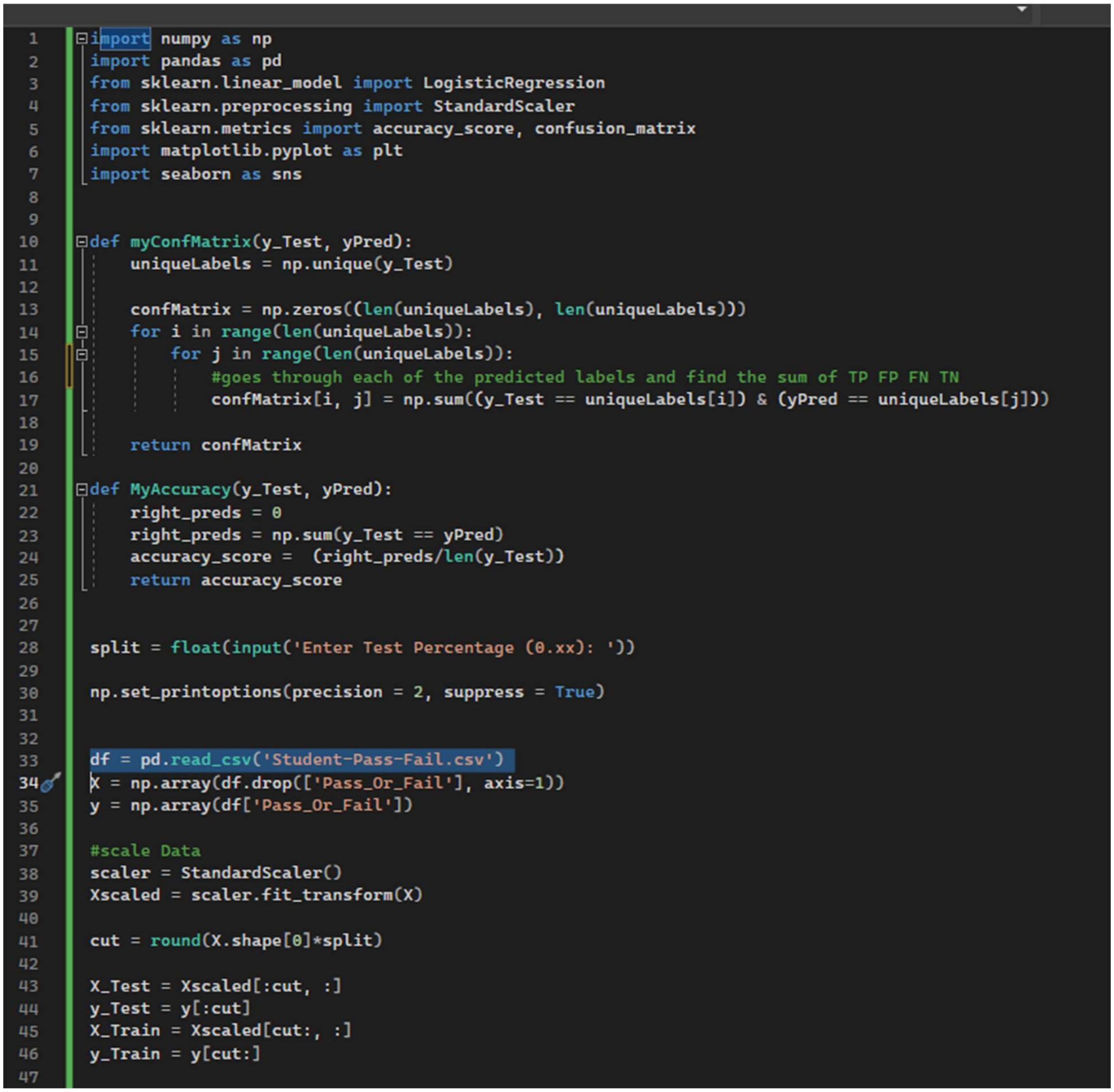
1.



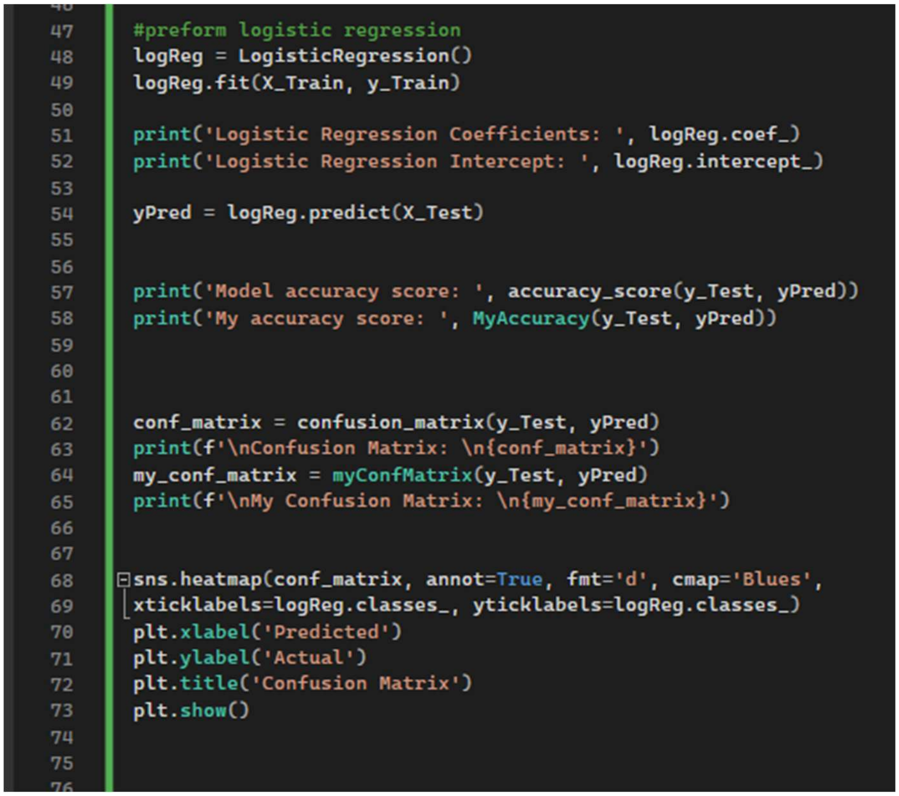
```python

import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler

np.set_printoptions(precision = 2, suppress = True)

df = pd.read_csv('Student-Pass-Fail.csv')
X = np.array(df.drop(['Pass_Or_Fail'], axis=1))
y = np.array(df['Pass_Or_Fail'])

#scale Data
scaler = StandardScaler()
Xscaled = scaler.fit_transform(X)

#preform logistic regression
logReg = LogisticRegression()
logReg.fit(Xscaled, y)

print('Logistic Regression Coefficients: ', logReg.coef_)
print('Logistic Regression Intercept: ', logReg.intercept_)

dataPoints = np.array([
[7, 28],
[10, 34],
[2, 39]
])

dataPoints_scaled = scaler.transform(dataPoints)
yPred = logReg.predict(dataPoints_scaled)

#.predict_proba() gives probabilities for each class: [P(y=0), P(y=1)]
yProb = logReg.predict_proba(dataPoints_scaled)[:, 1]

odds = np.exp(logReg.coef_)
print('Odds of pass/fail')
print(odds)
#print(f'yProb: {yProb}')

for c in range(yPred.shape[0]):
    if yPred[c] == 1:
        print(f'Client {c+1} Predition: Pass')
    elif yPred[c] == 0:
        print(f'Client {c+1} Prediction: Fail')
for c in range(yProb.shape[0]):
    print(f'Student {c+1} probality of passing: {yProb[c]*100:.2f}%')
```

```
Logistic Regression Coefficients:  [[ 4.17 -3.82]]
Logistic Regression Intercept:  [-3.24]
Odds of pass/fail
[[65.    0.02]]
Client 1 Predition: Pass
Client 2 Predition: Pass
Client 3 Prediction: Fail
Student 1 probality of passing: 78.50%
Student 2 probality of passing: 96.66%
Student 3 probality of passing: 0.00%
Press any key to continue . . .
```

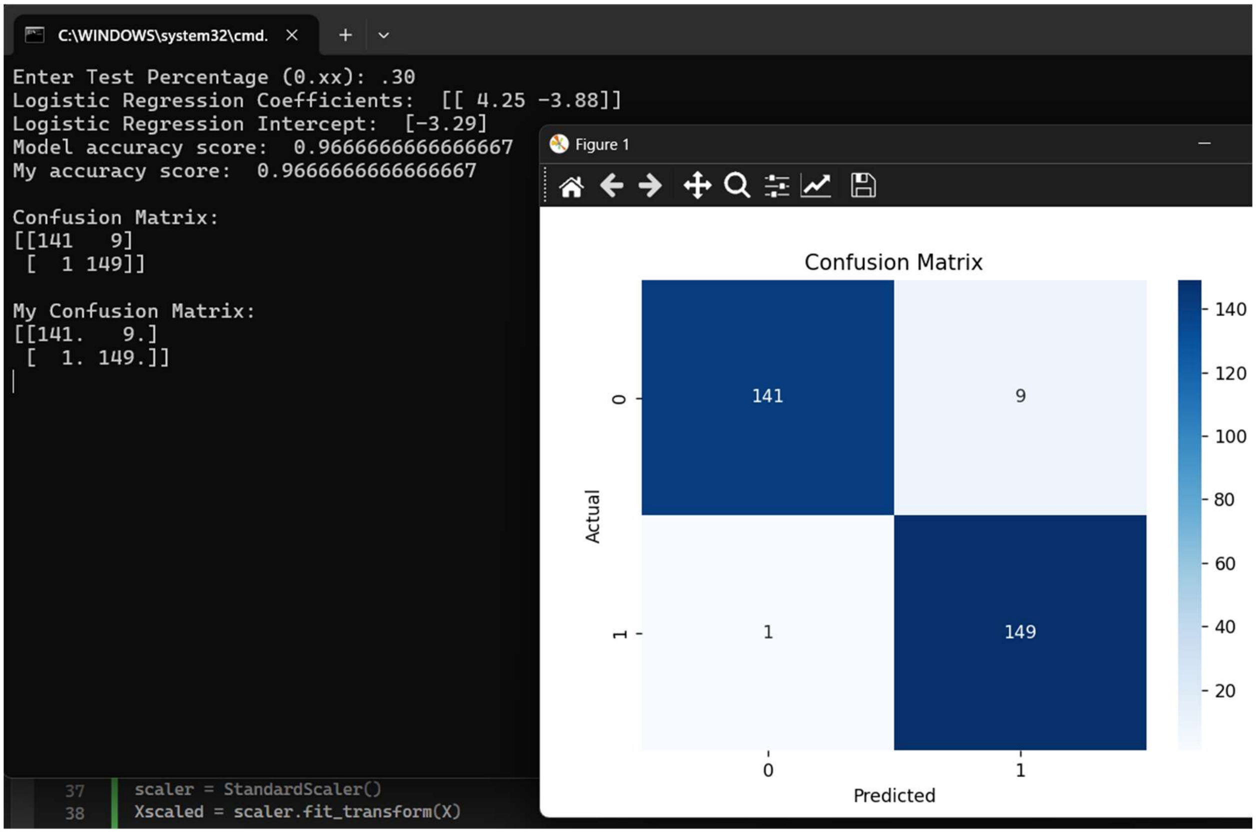
2.



```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns


def myConfMatrix(y_Test, yPred):
    uniqueLabels = np.unique(y_Test)

    confMatrix = np.zeros((len(uniqueLabels), len(uniqueLabels)))
    for i in range(len(uniqueLabels)):
        for j in range(len(uniqueLabels)):
            #goes through each of the predicted labels and find the sum of TP FP FN TN
            confMatrix[i, j] = np.sum((y_Test == uniqueLabels[i]) & (yPred == uniqueLabels[j]))

    return confMatrix

def MyAccuracy(y_Test, yPred):
    right_preds = 0
    right_preds = np.sum(y_Test == yPred)
    accuracy_score =  (right_preds/len(y_Test))
    return accuracy_score


split = float(input('Enter Test Percentage (0.xx): '))

np.set_printoptions(precision = 2, suppress = True)


df = pd.read_csv('Student-Pass-Fail.csv')
X = np.array(df.drop(['Pass_Or_Fail'], axis=1))
y = np.array(df['Pass_Or_Fail'])

#scale Data
scaler = StandardScaler()
Xscaled = scaler.fit_transform(X)

cut = round(X.shape[0]*split)

X_Test = Xscaled[:cut, :]
y_Test = y[:cut]
X_Train = Xscaled[cut:, :]
y_Train = y[cut:]

```



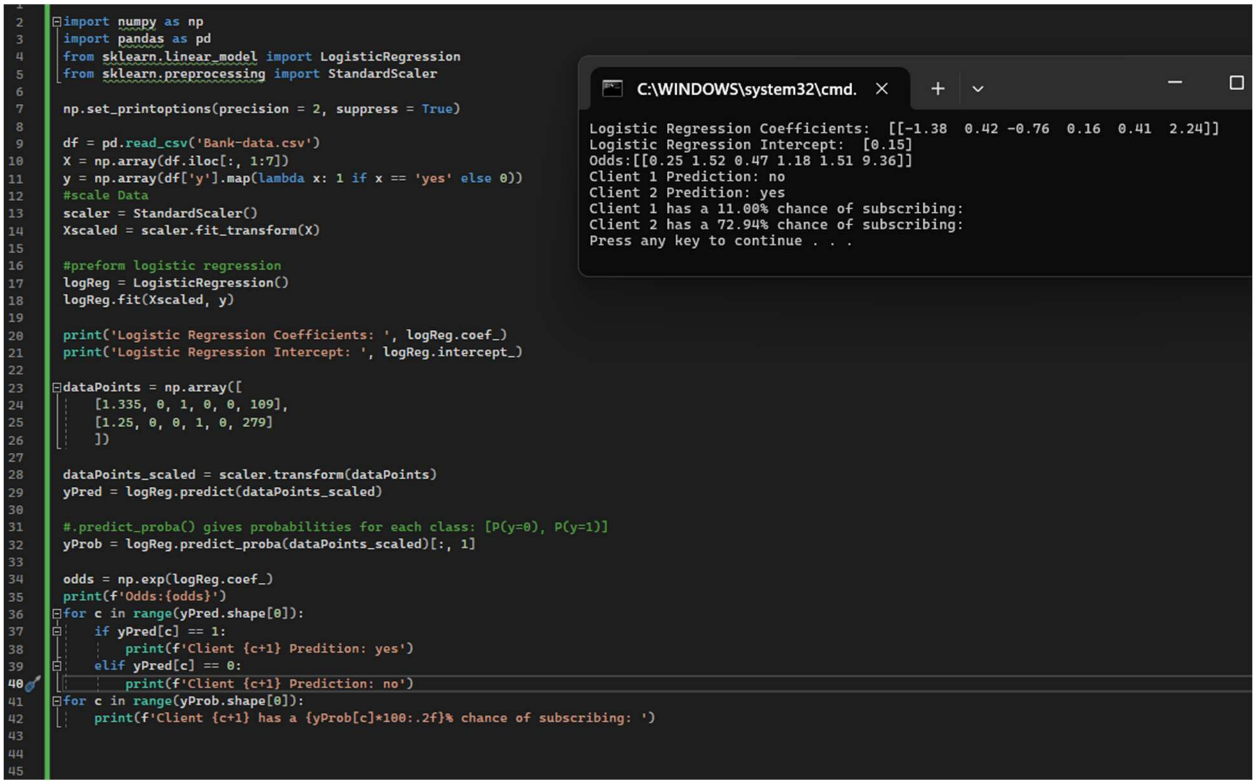
```
#preform logistic regression
logReg = LogisticRegression()
logReg.fit(X_Train, y_Train)

print('Logistic Regression Coefficients: ', logReg.coef_)
print('Logistic Regression Intercept: ', logReg.intercept_)

yPred = logReg.predict(X_Test)


print('Model accuracy score: ', accuracy_score(y_Test, yPred))
print('My accuracy score: ', MyAccuracy(y_Test, yPred))



conf_matrix = confusion_matrix(y_Test, yPred)
print(f'\nConfusion Matrix: \n{conf_matrix}')
my_conf_matrix = myConfMatrix(y_Test, yPred)
print(f'\nMy Confusion Matrix: \n{my_conf_matrix}')


sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
xticklabels=logReg.classes_, yticklabels=logReg.classes_)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```



```
Enter Test Percentage (0.xx): .30
Logistic Regression Coefficients:  [[ 4.25 -3.88]]
Logistic Regression Intercept:  [-3.29]
Model accuracy score:  0.9666666666666667
My accuracy score:  0.9666666666666667

Confusion Matrix:
[[141   9]
 [  1 149]]

My Confusion Matrix:
[[141.   9.]
 [  1. 149.]]
```

```
scaler = StandardScaler()
Xscaled = scaler.fit_transform(X)
```

3.



```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler

np.set_printoptions(precision = 2, suppress = True)

df = pd.read_csv('Bank-data.csv')
X = np.array(df.iloc[:, 1:7])
y = np.array(df['y'].map(lambda x: 1 if x == 'yes' else 0))
#scale Data
scaler = StandardScaler()
Xscaled = scaler.fit_transform(X)

#preform logistic regression
logReg = LogisticRegression()
logReg.fit(Xscaled, y)

print('Logistic Regression Coefficients: ', logReg.coef_)
print('Logistic Regression Intercept: ', logReg.intercept_)

dataPoints = np.array([
    [1.335, 0, 1, 0, 0, 109],
    [1.25, 0, 0, 1, 0, 279]
    ])

dataPoints_scaled = scaler.transform(dataPoints)
yPred = logReg.predict(dataPoints_scaled)

#.predict_proba() gives probabilities for each class: [P(y=0), P(y=1)]
yProb = logReg.predict_proba(dataPoints_scaled)[:, 1]

odds = np.exp(logReg.coef_)
print(f'Odds:{odds}')
for c in range(yPred.shape[0]):
    if yPred[c] == 1:
        print(f'Client {c+1} Predition: yes')
    elif yPred[c] == 0:
        print(f'Client {c+1} Prediction: no')
for c in range(yProb.shape[0]):
    print(f'Client {c+1} has a {yProb[c]*100:.2f}% chance of subscribing: ')
```

```
Logistic Regression Coefficients:  [[-1.38  0.42 -0.76  0.16  0.41  2.24]]
Logistic Regression Intercept:  [0.15]
Odds:[[0.25 1.52 0.47 1.18 1.51 9.36]]
Client 1 Prediction: no
Client 2 Predition: yes
Client 1 has a 11.00% chance of subscribing:
Client 2 has a 72.94% chance of subscribing:
Press any key to continue . . .
```