# ELEC 390 – Lab 06

Department of Electrical and Computer Engineering

Queen’s University

Composed By

Nicholas Seegobin (20246787)

Zeerak Asim (20237955)

Lauren Steel (20218337)

Saman Saeidi (20217992)

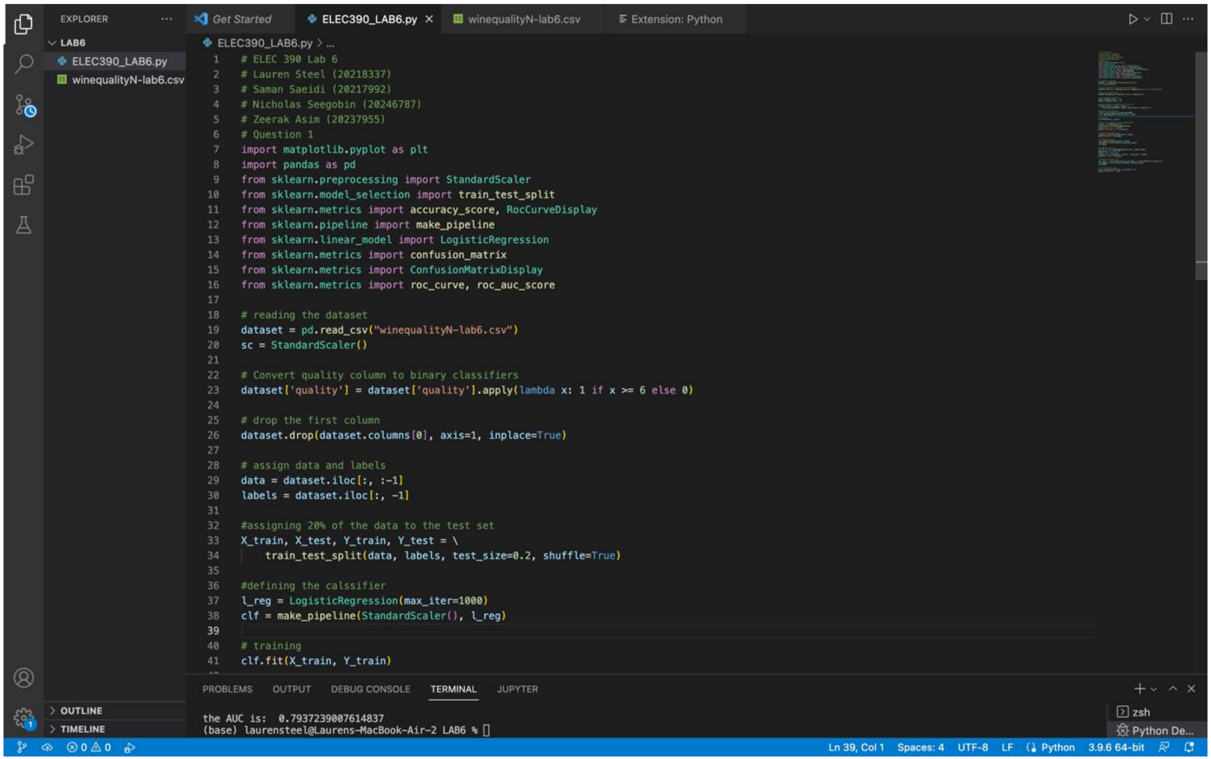
Section 03

Date of Submission

2023 March 30th

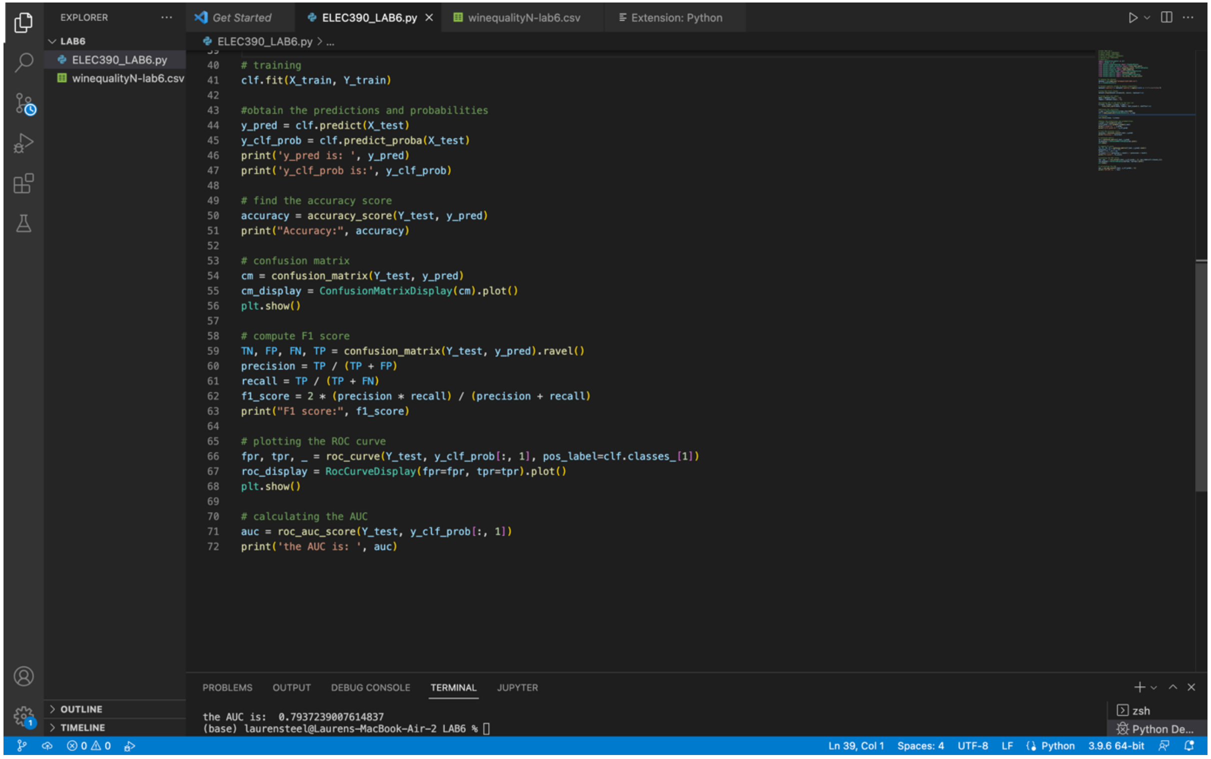
# Part 1

## Python Code



```python
# ELEC 390 Lab 6
# Lauren Steel (20218337)
# Saman Saeidi (20217992)
# Nicholas Seegobin (20246787)
# Zeerak Asim (20237955)
# Question 1
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, RocCurveDisplay
from sklearn.pipeline import make_pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.metrics import roc_curve, roc_auc_score

# reading the dataset
dataset = pd.read_csv("winequalityN-lab6.csv")
sc = StandardScaler()

# Convert quality column to binary classifiers
dataset['quality'] = dataset['quality'].apply(lambda x: 1 if x >= 6 else 0)

# drop the first column
dataset.drop(dataset.columns[0], axis=1, inplace=True)

# assign data and labels
data = dataset.iloc[:, :-1]
labels = dataset.iloc[:, -1]

#assigning 20% of the data to the test set
X_train, X_test, Y_train, Y_test = \
    train_test_split(data, labels, test_size=0.2, shuffle=True)

#defining the calssifier
l_reg = LogisticRegression(max_iter=1000)
clf = make_pipeline(StandardScaler(), l_reg)

# training
clf.fit(X_train, Y_train)
```

```
the AUC is:  0.7937239007614837
(base) laurensteel@Laurens-MacBook-Air-2 LAB6 % 
```

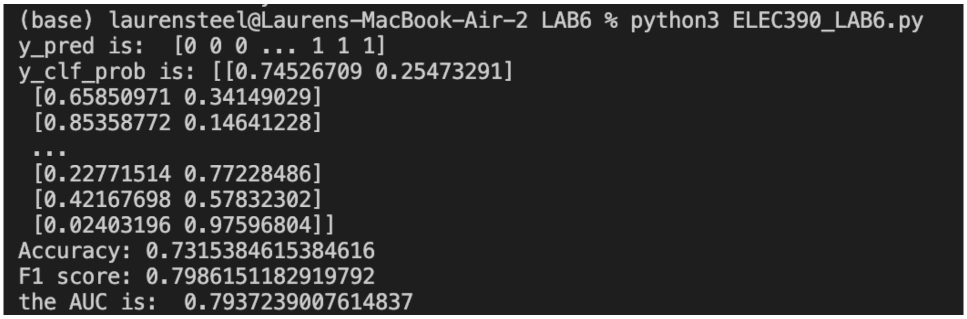


```python
# training
clf.fit(X_train, Y_train)

#obtain the predictions and probabilities
y_pred = clf.predict(X_test)
y_clf_prob = clf.predict_proba(X_test)
print('y_pred is: ', y_pred)
print('y_clf_prob is:', y_clf_prob)

# find the accuracy score
accuracy = accuracy_score(Y_test, y_pred)
print("Accuracy:", accuracy)

# confusion matrix
cm = confusion_matrix(Y_test, y_pred)
cm_display = ConfusionMatrixDisplay(cm).plot()
plt.show()

# compute F1 score
TN, FP, FN, TP = confusion_matrix(Y_test, y_pred).ravel()
precision = TP / (TP + FP)
recall = TP / (TP + FN)
f1_score = 2 * (precision * recall) / (precision + recall)
print("F1 score:", f1_score)

# plotting the ROC curve
fpr, tpr, _ = roc_curve(Y_test, y_clf_prob[:, 1], pos_label=clf.classes_[1])
roc_display = RocCurveDisplay(fpr=fpr, tpr=tpr).plot()
plt.show()

# calculating the AUC
auc = roc_auc_score(Y_test, y_clf_prob[:, 1])
print('the AUC is: ', auc)
```

```
the AUC is:  0.7937239007614837
(base) laurensteel@Laurens-MacBook-Air-2 LAB6 % 
```

*Figure 1: Screenshots of python code for lab 6 part 1.*

## Output



```
(base) laurensteel@Laurens-MacBook-Air-2 LAB6 % python3 ELEC390_LAB6.py
y_pred is:  [0 0 0 ... 1 1 1]
y_clf_prob is: [[0.74526709 0.25473291]
 [0.65850971 0.34149029]
 [0.85358772 0.14641228]
 ...
 [0.22771514 0.77228486]
 [0.42167698 0.57832302]
 [0.02403196 0.97596804]]
Accuracy: 0.7315384615384616
F1 score: 0.7986151182919792
the AUC is:  0.7937239007614837
```

*Figure 2: Output form the python code part 1.*

From Figure 2 shown above we can see that the accuracy of the model is approximately 0.714615.

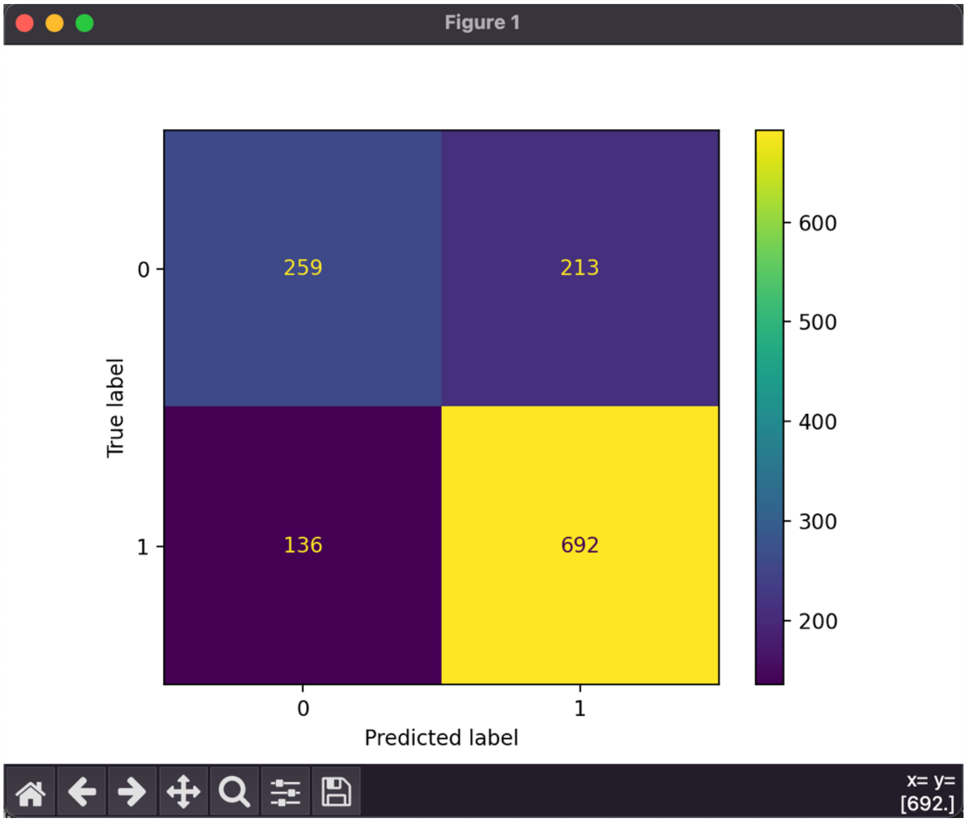


*Figure 3: Confusion Matrix*

Above in Figure 3 is the Confusion matrix produced by the code seen in Figure 1.

As seen in Figure 2 the F1 score is approximately 0.798615 and the AUC is 0.793723.

Finally, Figure 4 shown below is the ROC model.

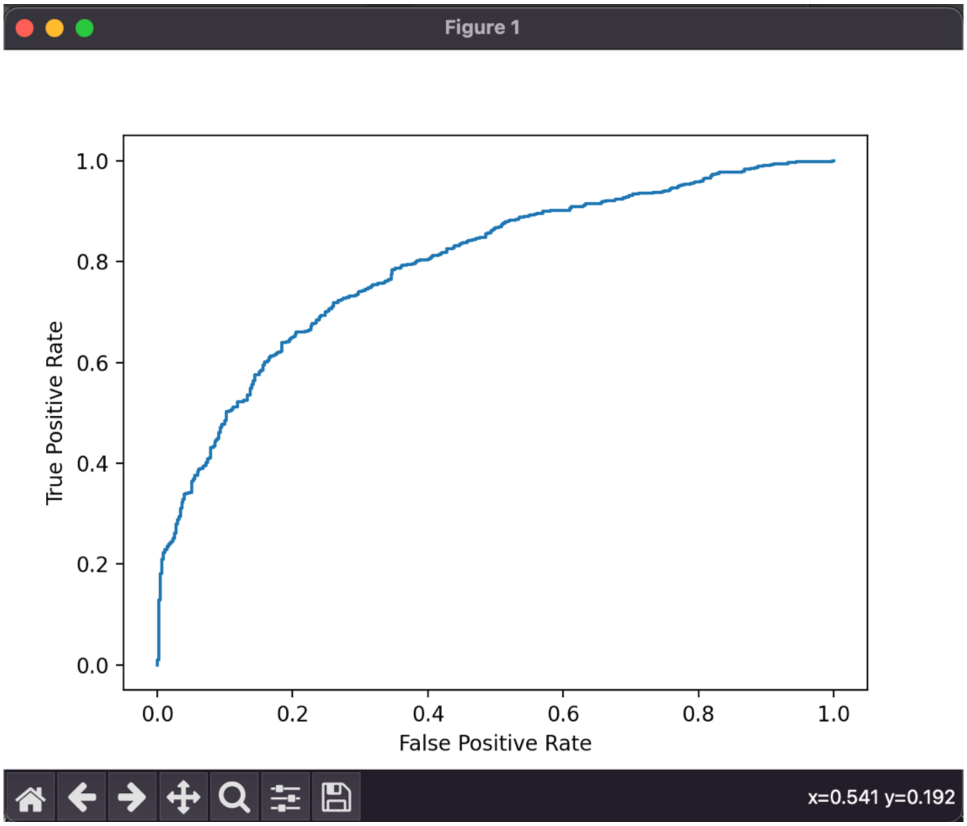
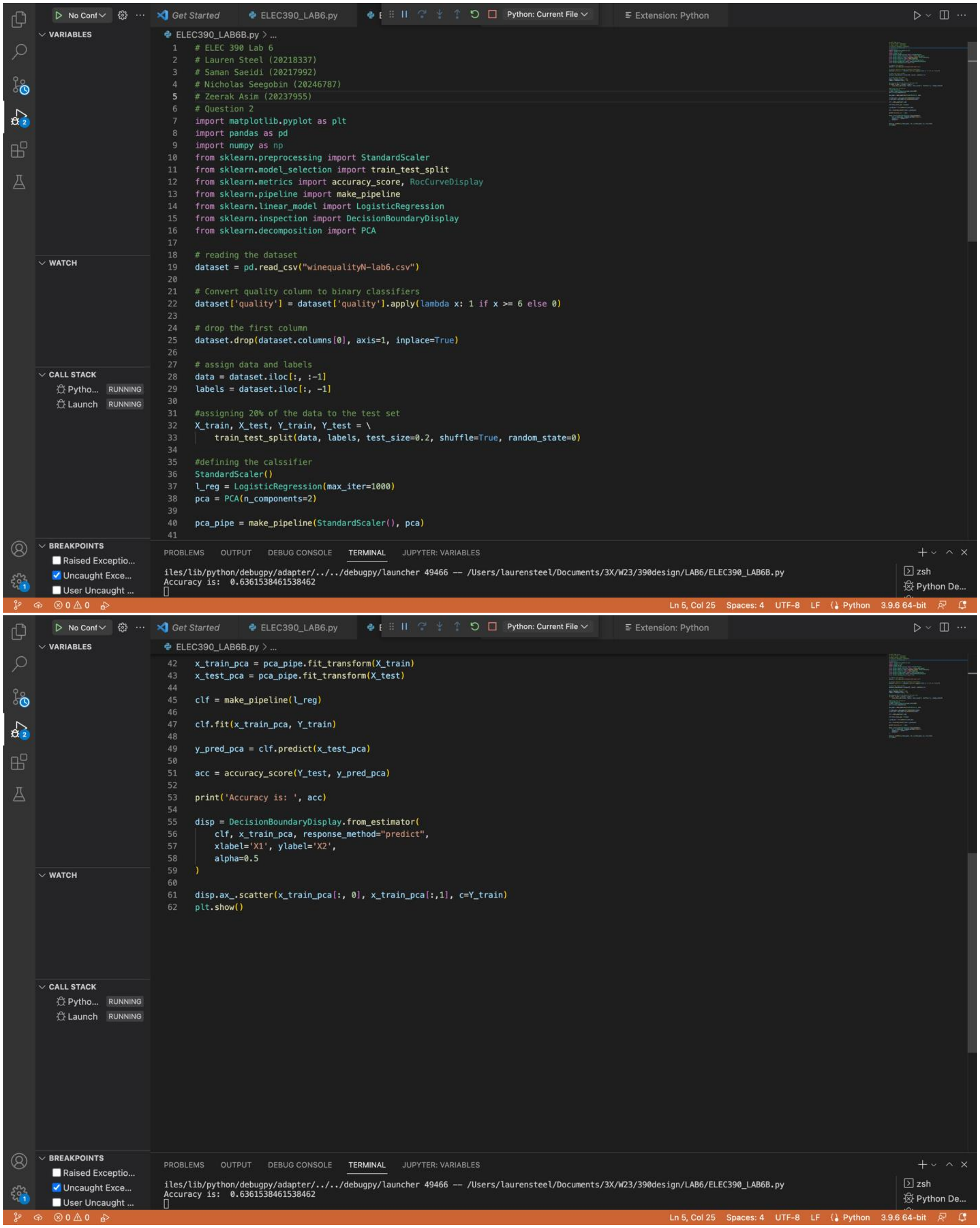


*Figure 4: ROC Model*

# Part 2

## Python Code



```python
# ELEC 390 Lab 6
# Lauren Steel (20218337)
# Saman Saeidi (20217992)
# Nicholas Seegobin (20246787)
# Zeerak Asim (20237955)
# Question 2
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, RocCurveDisplay
from sklearn.pipeline import make_pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.inspection import DecisionBoundaryDisplay
from sklearn.decomposition import PCA

# reading the dataset
dataset = pd.read_csv("winequalityN-lab6.csv")

# Convert quality column to binary classifiers
dataset['quality'] = dataset['quality'].apply(lambda x: 1 if x >= 6 else 0)

# drop the first column
dataset.drop(dataset.columns[0], axis=1, inplace=True)

# assign data and labels
data = dataset.iloc[:, :-1]
labels = dataset.iloc[:, -1]

#assigning 20% of the data to the test set
X_train, X_test, Y_train, Y_test = \
    train_test_split(data, labels, test_size=0.2, shuffle=True, random_state=0)

#defining the calssifier
StandardScaler()
l_reg = LogisticRegression(max_iter=1000)
pca = PCA(n_components=2)

pca_pipe = make_pipeline(StandardScaler(), pca)

```

```
iles/lib/python/debugpy/adapter/../../debugpy/launcher 49466 -- /Users/laurensteel/Documents/3X/W23/390design/LAB6/ELEC390_LAB6B.py
Accuracy is:  0.6361538461538462
```

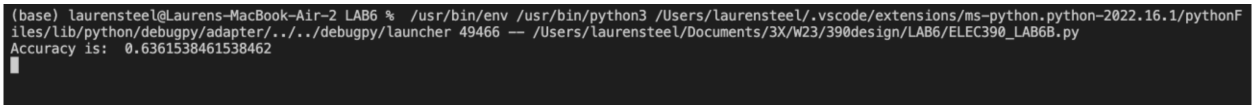
```python
x_train_pca = pca_pipe.fit_transform(X_train)
x_test_pca = pca_pipe.fit_transform(X_test)

clf = make_pipeline(l_reg)

clf.fit(x_train_pca, Y_train)

y_pred_pca = clf.predict(x_test_pca)

acc = accuracy_score(Y_test, y_pred_pca)

print('Accuracy is: ', acc)

disp = DecisionBoundaryDisplay.from_estimator(
    clf, x_train_pca, response_method="predict",
    xlabel='X1', ylabel='X2',
    alpha=0.5
)

disp.ax_.scatter(x_train_pca[:, 0], x_train_pca[:,1], c=Y_train)
plt.show()
```

```
iles/lib/python/debugpy/adapter/../../debugpy/launcher 49466 -- /Users/laurensteel/Documents/3X/W23/390design/LAB6/ELEC390_LAB6B.py
Accuracy is:  0.6361538461538462
```

*Figure 5: Screenshots of the python code for lab 6 part 2.*

## Output

Below in Figure 6 is a screenshot of the output developed by the code from Figure 5. As we can see in the image the accuracy of the model is approximately 0.636153.



```
(base) laurensteel@Laurens-MacBook-Air-2 LAB6 %  /usr/bin/env /usr/bin/python3 /Users/laurensteel/.vscode/extensions/ms-python.python-2022.16.1/pythonF
iles/lib/python/debugpy/adapter/../../debugpy/launcher 49466 -- /Users/laurensteel/Documents/3X/W23/390design/LAB6/ELEC390_LAB6B.py
Accuracy is:  0.6361538461538462
█
```

*Figure 6: Output from the python code part 2.*

## Training Data

Figure 7 is a representation of the training data and its decision boundary.

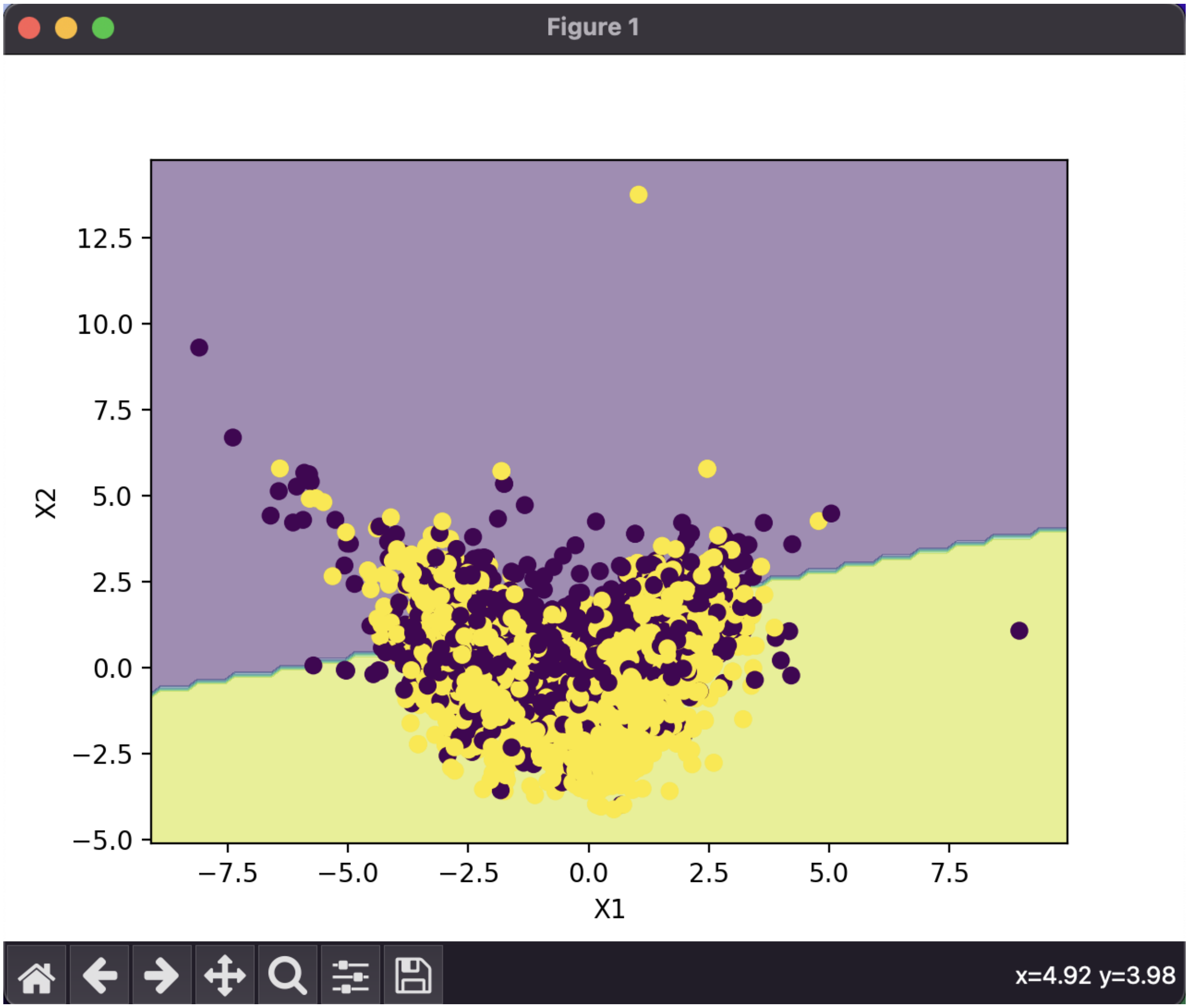


*Figure 7: Training data (reduced to 2 dimensions) and its decision boundary produced from the python code part 2.*

**Compare the accuracy of Part 2 with Part 1. What is your observation and why?**

As seen in the data presented above the accuracy of the model in part 1 is approximately 0.714615 while the accuracy of the model in part 2 is approximately 0.636153. Overall, a higher accuracy value is better because it shows that the model is making correct predictions more frequently. This makes sense as the model in part 2 only uses 2 components of the PCA instead of all the data.