Name: Shane Bowen
ID: R00149085
Class: SDH4-A

Task 1

```
def main():
    product_images = pd.read_csv("product_images.csv")
    num_samples = int(input("Enter the num of sample: "))
    num_splits = int(input("Enter the num of splits: "))
    splitting_and_counting(product_images)
    perceptron(product_images, num_samples, num_splits)
    support_vector_machine(product_images, num_samples, num_splits)

#task1
def splitting_and_counting(product_images):
    print("\n***************Task 1*******************")
    print(product_images)

    sneakers = product_images[product_images.label == 0]
    print(sneakers)

    plt.imshow(sneakers.values[0][1:].reshape(28, 28))
    plt.show()

    ankle_boots = product_images[product_images.label == 1]
    print(ankle_boots)

    plt.imshow(ankle_boots.values[0][1:].reshape(28, 28))
    plt.show()

    print("Num of Sneakers:", len(sneakers))
    print("Num of Ankle Boots:", len(ankle_boots))
```

In Task 1, I am loading the product images csv file using pandas read_csv, I split the data into sneakers if label = 0 and into ankle_boots if label = 1, then I plot the first of both the sneakers and the ankle_boots. To get the number of samples for both items, I do a length of the array.

Task 2

```
#task2
def perceptron(product_images, num_samples, num_splits):
    print("\n***************Task 2*******************")
    training_time = []
    prediction_time = []
    prediction_accuracy = []

    kf = model_selection.KFold(n_splits=num_splits, shuffle=True)

    product_images = product_images.head(num_samples)

    for train_index,test_index in kf.split(product_images.values):
        clf1 = linear_model.Perceptron()

        start = time.time()
        clf1.fit(product_images.values[train_index], product_images.label[train_index])
        prediction1 = clf1.predict(product_images.values[test_index])
        end = time.time()

        training_time.append(end - start)

        start = time.time()
        score1 = metrics.accuracy_score(product_images.label[test_index], prediction1)
        end = time.time()

        prediction_time.append(end - start)

        confusion = metrics.confusion_matrix(product_images.label[test_index], prediction1)

        prediction_accuracy.append(score1)
        print("Perceptron accuracy score: ", score1)
        print("Confusion Matrix: \n", confusion)
```

```
print("----------Training Time-----------")
print("Max time value: ", np.max(training_time))
print("Min time value: ", np.min(training_time))
print("Avg time value: ", np.mean(training_time))
print()

print("----------Prediction Time-----------")
print("Max time value: ", np.max(prediction_time))
print("Min time value: ", np.min(prediction_time))
print("Avg time value: ", np.mean(prediction_time))
print()

print("----------Prediction Accuracy-----------")
print("Max accuracy score: ", np.max(prediction_accuracy))
print("Min accuracy score: ", np.min(prediction_accuracy))
print("Avg accuracy score: ", np.mean(prediction_accuracy))
print()
```

In Task 2, the function I parameterize the data, number of samples and number of splits. I use all three of these items in my k-fold cross validation procedure, the data is split up into training and test subsets, I create my Perceptron classifier that I will use to train the data, measure the start and end time for the time it took to train the data, the time taken is the difference between these two values, I do the same for when I calculate my accuracy score. I also calculate the confusion matrix by passing in the labels and the prediction. The result I get in each split for the time and accuracy score, I append it to an array to store the results. I get the min, max, and avg for the training time, prediction time and prediction accuracy using numpy. I print these values to the console.

Task 3

```
#task3
def support_vector_machine(product_images, num_samples, num_splits):
    print("\n****************Task 3*******************")
    training_time = []
    prediction_time = []
    linear_prediction_accuracy = []
    rbf_prediction_accuracy = []

    kf = model_selection.KFold(n_splits=num_splits, shuffle=True)

    product_images = product_images.head(num_samples)

    for train_index,test_index in kf.split(product_images.values):
        clf1 = svm.SVC(kernel="linear", gamma=1e-2)
        clf2 = svm.SVC(kernel="rbf", gamma=1e-2)

        start = time.time()
        clf1.fit(product_images.values[train_index], product_images.label[train_index])
        prediction1 = clf1.predict(product_images.values[test_index])

        clf2.fit(product_images.values[train_index], product_images.label[train_index])
        prediction2 = clf2.predict(product_images.values[test_index])
        end = time.time()

        training_time.append(end - start)

        start = time.time()
        score1 = metrics.accuracy_score(product_images.label[test_index], prediction1)
        score2 = metrics.accuracy_score(product_images.label[test_index], prediction2)
        end = time.time()

        prediction_time.append(end - start)


    confusion1 = metrics.confusion_matrix(product_images.label[test_index], prediction1)
    confusion2 = metrics.confusion_matrix(product_images.label[test_index], prediction1)

    linear_prediction_accuracy.append(score1)
    rbf_prediction_accuracy.append(score2)
    print("SVM with Linear kernel accuracy score: ", score1)
    print("SVM with RBF kernel accuracy score: ", score2)
    print("Linear Kernel Confusion Matrix: \n", confusion1)
    print("RBF Kernel Confusion Matrix: \n", confusion2)

    print()

print("----------Training Time-----------")
print("Max time value: ", np.max(training_time))
print("Min time value: ", np.min(training_time))
print("Avg time value: ", np.mean(training_time))
print()

print("----------Prediction Time-----------")
print("Max time value: ", np.max(prediction_time))
print("Min time value: ", np.min(prediction_time))
print("Avg time value: ", np.mean(prediction_time))
print()

print("-------Linear Prediction Accuracy--------")
print("Max accuracy score: ", np.max(linear_prediction_accuracy))
print("Min accuracy score: ", np.min(linear_prediction_accuracy))
print("Avg accuracy score: ", np.mean(linear_prediction_accuracy))
print()

print("--------RBF Prediction Accuracy---------")
print("Max accuracy score: ", np.max(rbf_prediction_accuracy))
print("Min accuracy score: ", np.min(rbf_prediction_accuracy))
print("Avg accuracy score: ", np.mean(rbf_prediction_accuracy))
print()
```

In Task 3, I pass my three arguments to the function, I create a k-fold cross validation procedure by passing in my argument to it. I declare the linear and rbf as my two classifier to test my data, each are used to give a prediction score, time taken to train the data is monitored, an accuracy score is also calculated from the two prediction scores, the prediction time is monitored, the two accuracy scores and the training and testing time is appended to the array for each split. The confusion matrix is calculated using the labels and the prediction score for both linear and rbf, min, max and avg score are calculated using numpy for each array and printed to the console. For choosing the gamma I tested values in a range of 1e-1 < γ < 1e-5, and found out that 1e-2 returned the highest accuracy score in the tests I conducted and the Max and Min values weren't too far apart.

Task 4

```
----------Training Time-----------
Max time value:  0.08040404319763184
Min time value:  0.02731800079345703
Avg time value:  0.05671858787536621

----------Prediction Time-----------
Max time value:  0.004206418991088867
Min time value:  0.001882791519165039
Avg time value:  0.0023771286010742187

-------Perceptron Prediction Accuracy-------
Max accuracy score:  0.965
Min accuracy score:  0.925
Avg accuracy score:  0.9479999999999998
```

```
-------Linear Training Time--------
Max time value:  0.21038365364074707
Min time value:  0.17273426055908203
Avg time value:  0.18523092269897462

-------Linear Prediction Time--------
Max time value:  0.03943681716918945
Min time value:  0.030283689498901367
Avg time value:  0.03485560417175293

---------RBF Training Time---------
Max time value:  0.9424102306365967
Min time value:  0.9303417205810547
Avg time value:  0.9350171566009522

---------RBF Prediction Time---------
Max time value:  0.23081398010253906
Min time value:  0.2265760898590088
Avg time value:  0.22765560150146485

-------Linear Prediction Accuracy--------
Max accuracy score:  0.955
Min accuracy score:  0.925
Avg accuracy score:  0.9389999999999998

--------RBF Prediction Accuracy---------
Max accuracy score:  0.53
Min accuracy score:  0.425
Avg accuracy score:  0.481
```

The Perceptron classifier has the quickest runtime of the three and has the second highest accuracy score.
The Linear classifier has the second quickest runtime of the three and has the highest accuracy score.
The RBF classifier has the slowest runtime of the three and has the lowest accuracy score.

For me it would be a choice between the Perceptron or the Linear classifier to choose. If you are looking for a quick runtime go for the Perceptron classifier, But if you are looking for the highest accuracy score go for the Linear classifier. For me I would go for the Linear as a small drop in runtime will not be a huge factor for this dataset and also the Max and Min are closer together compared to the Linear meaning the avg is a better representation of the scores.