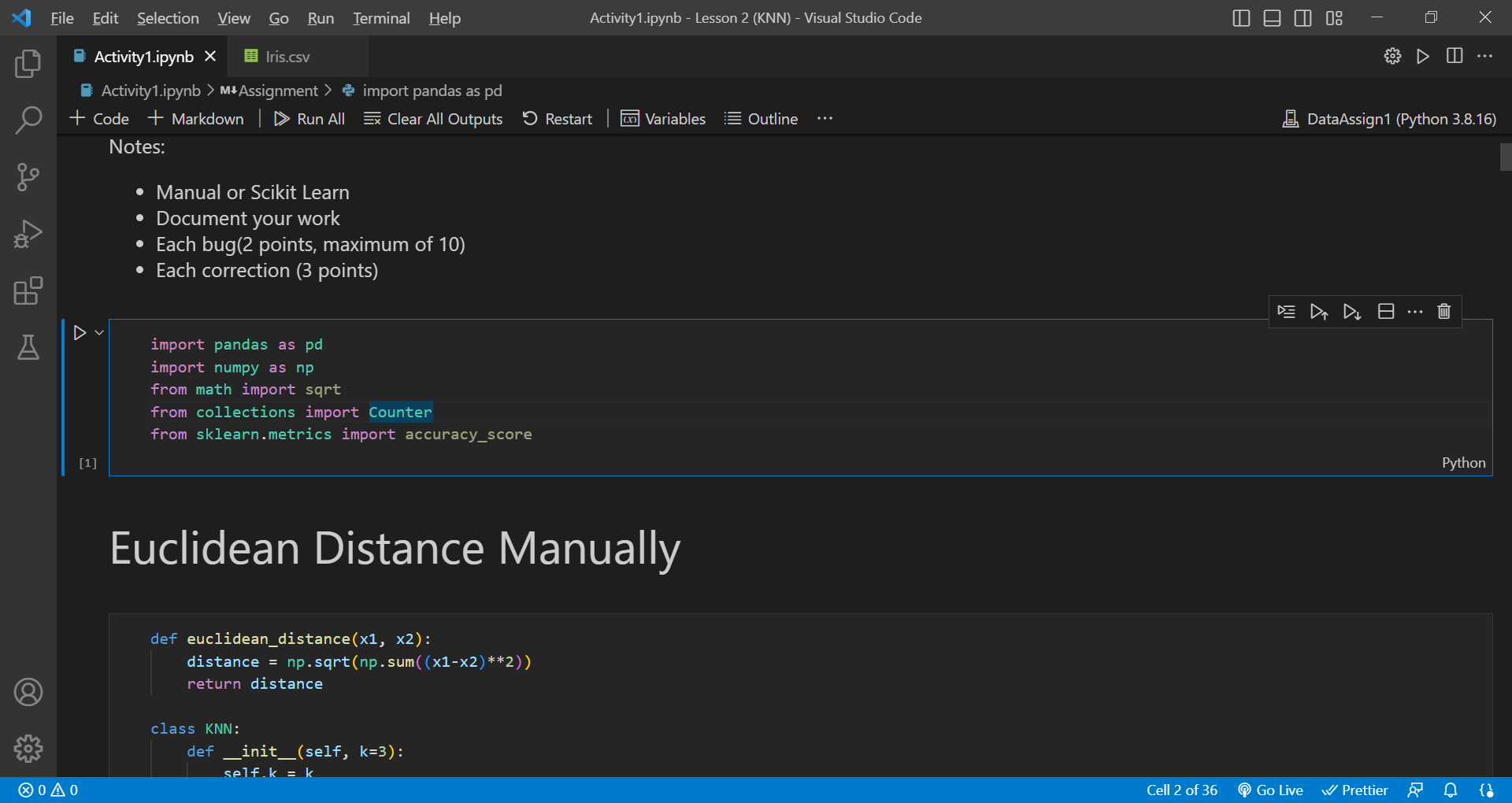
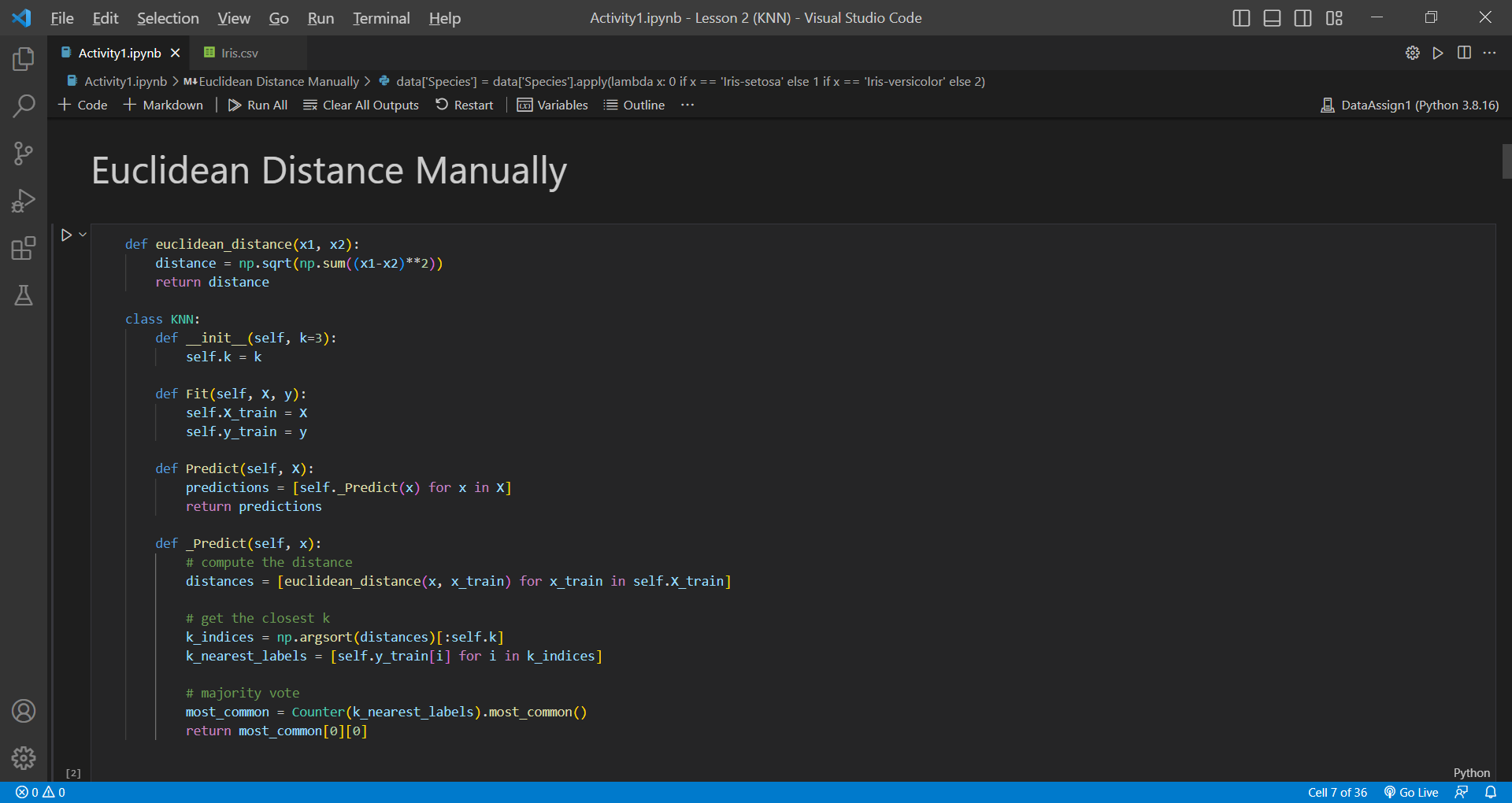
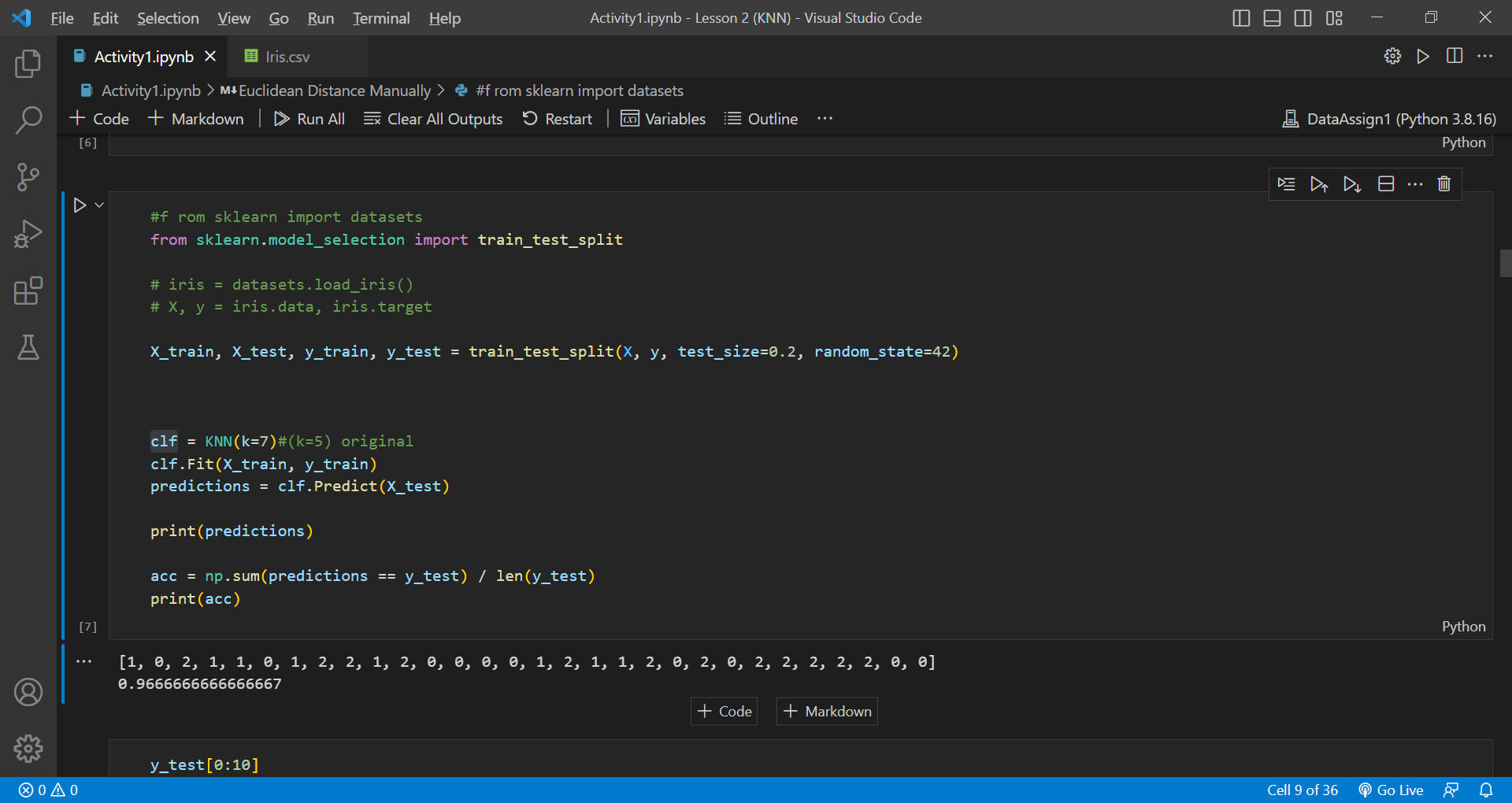
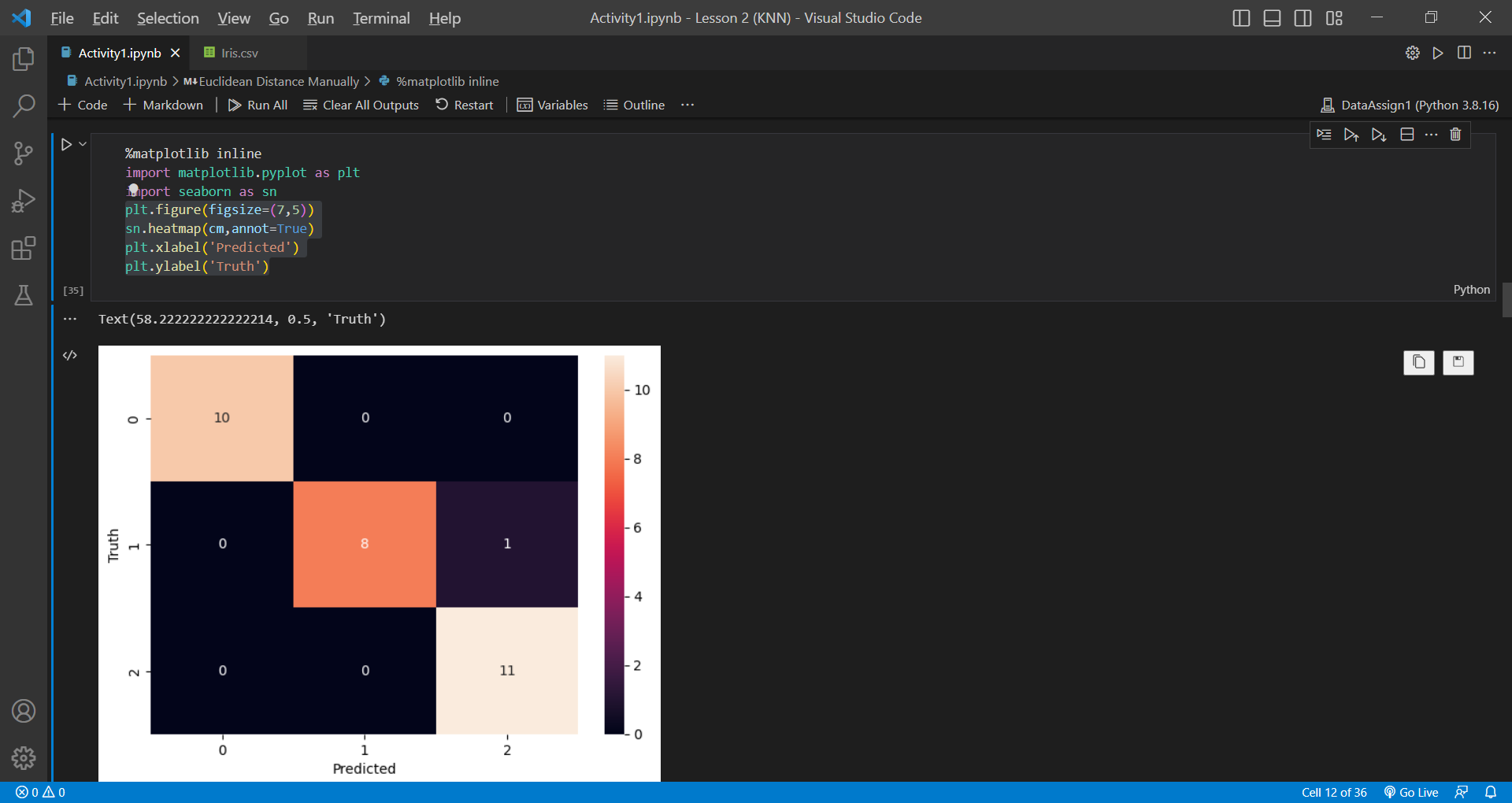
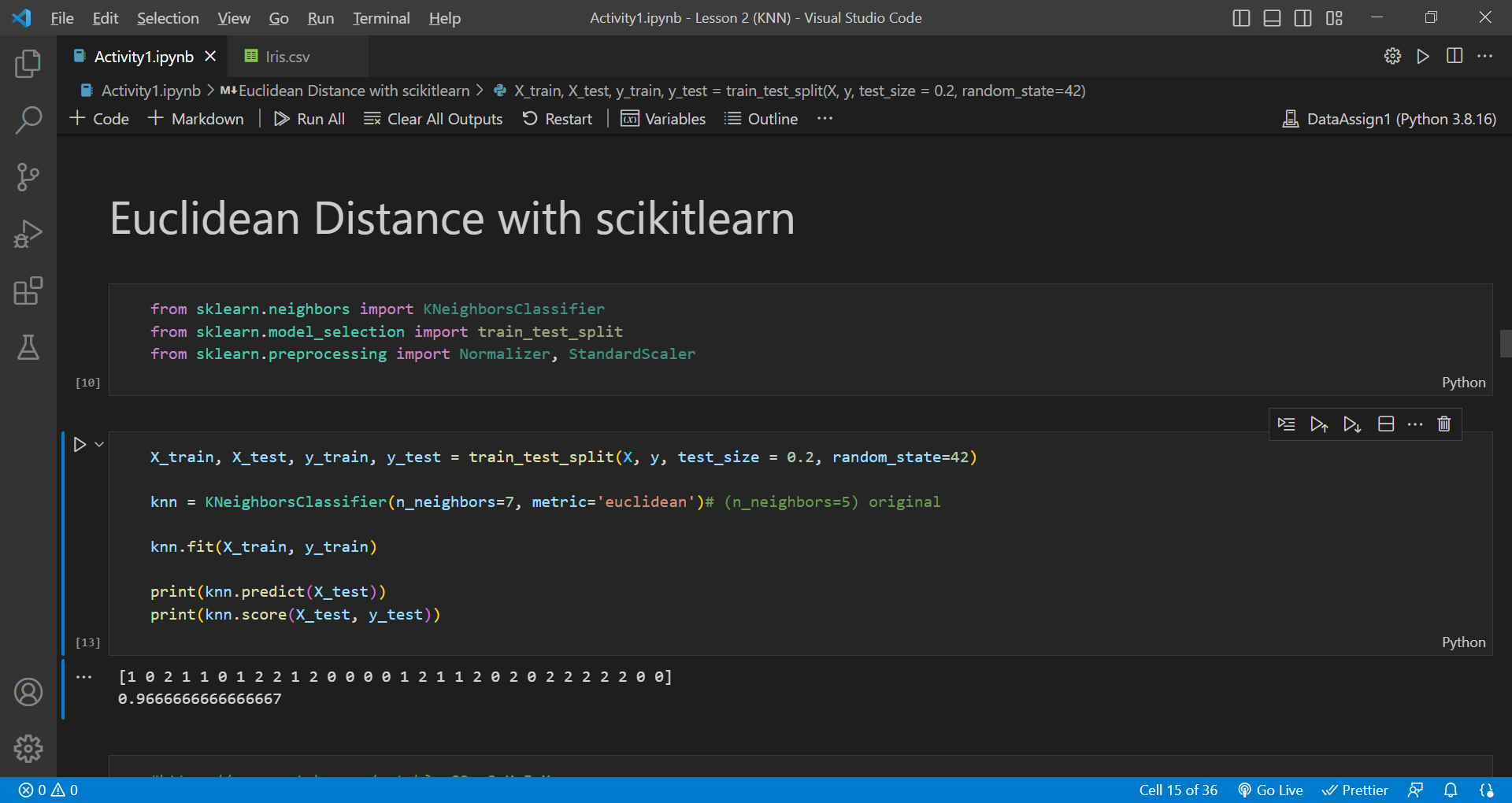
Libraries

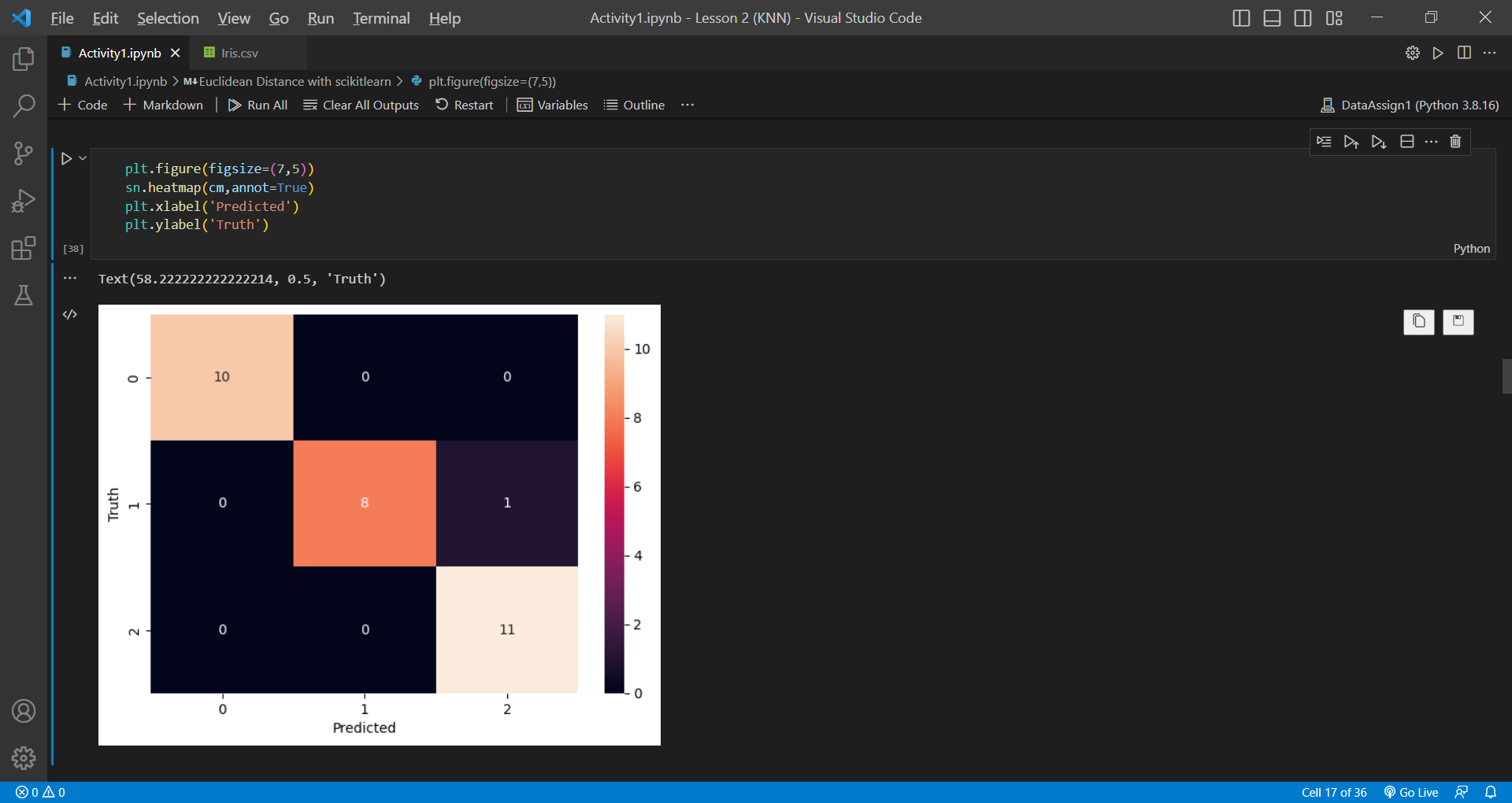


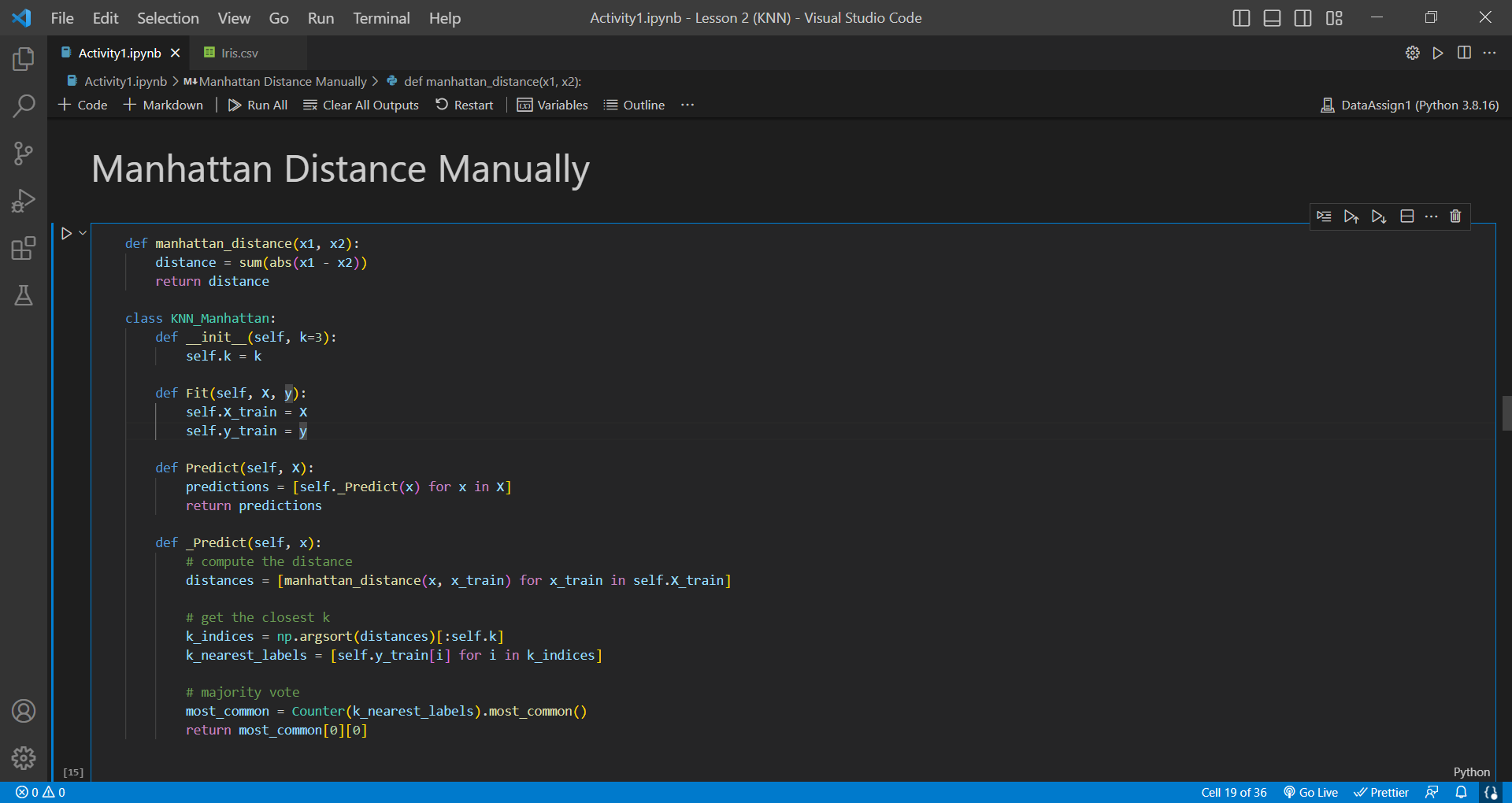


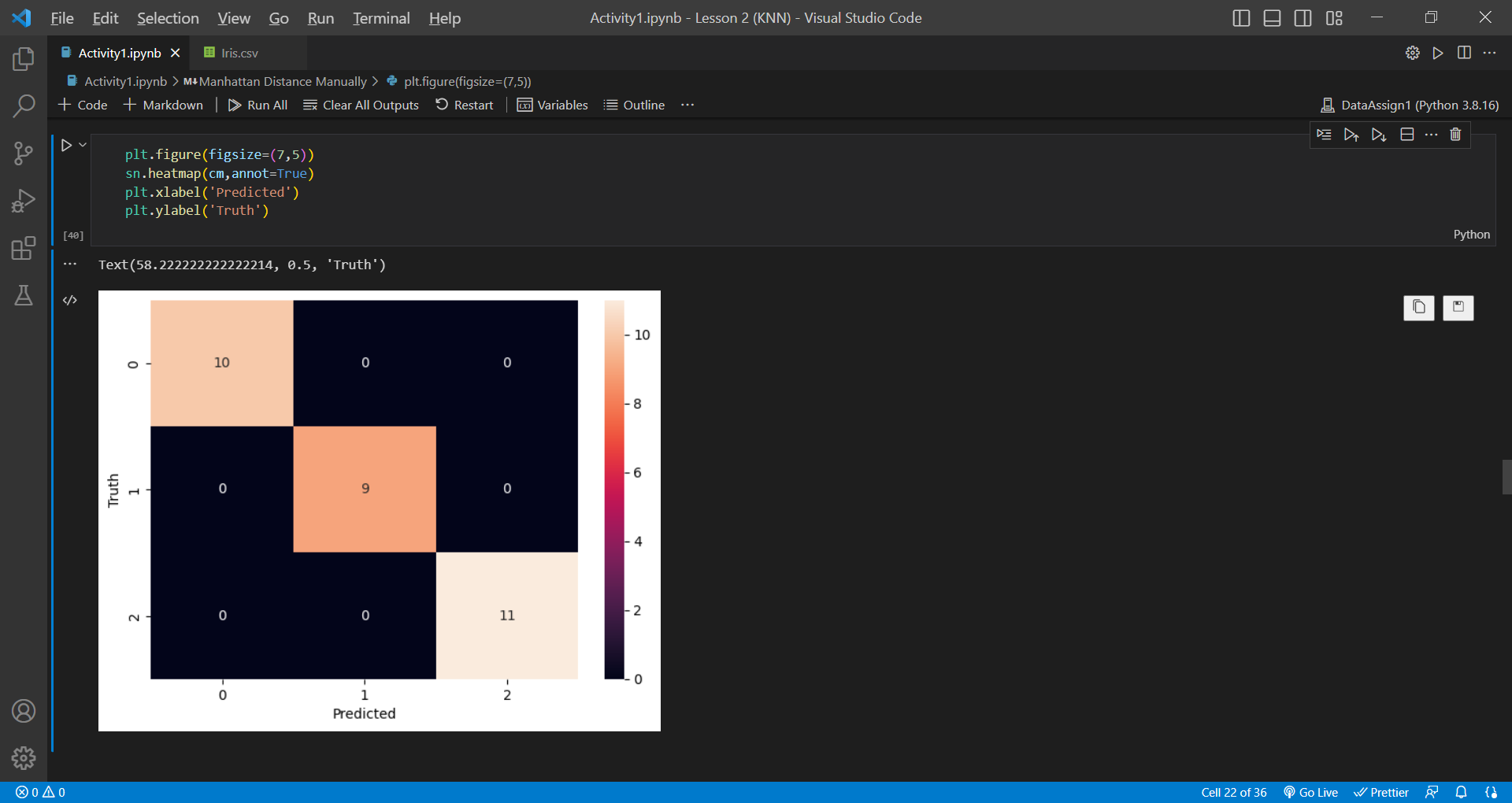


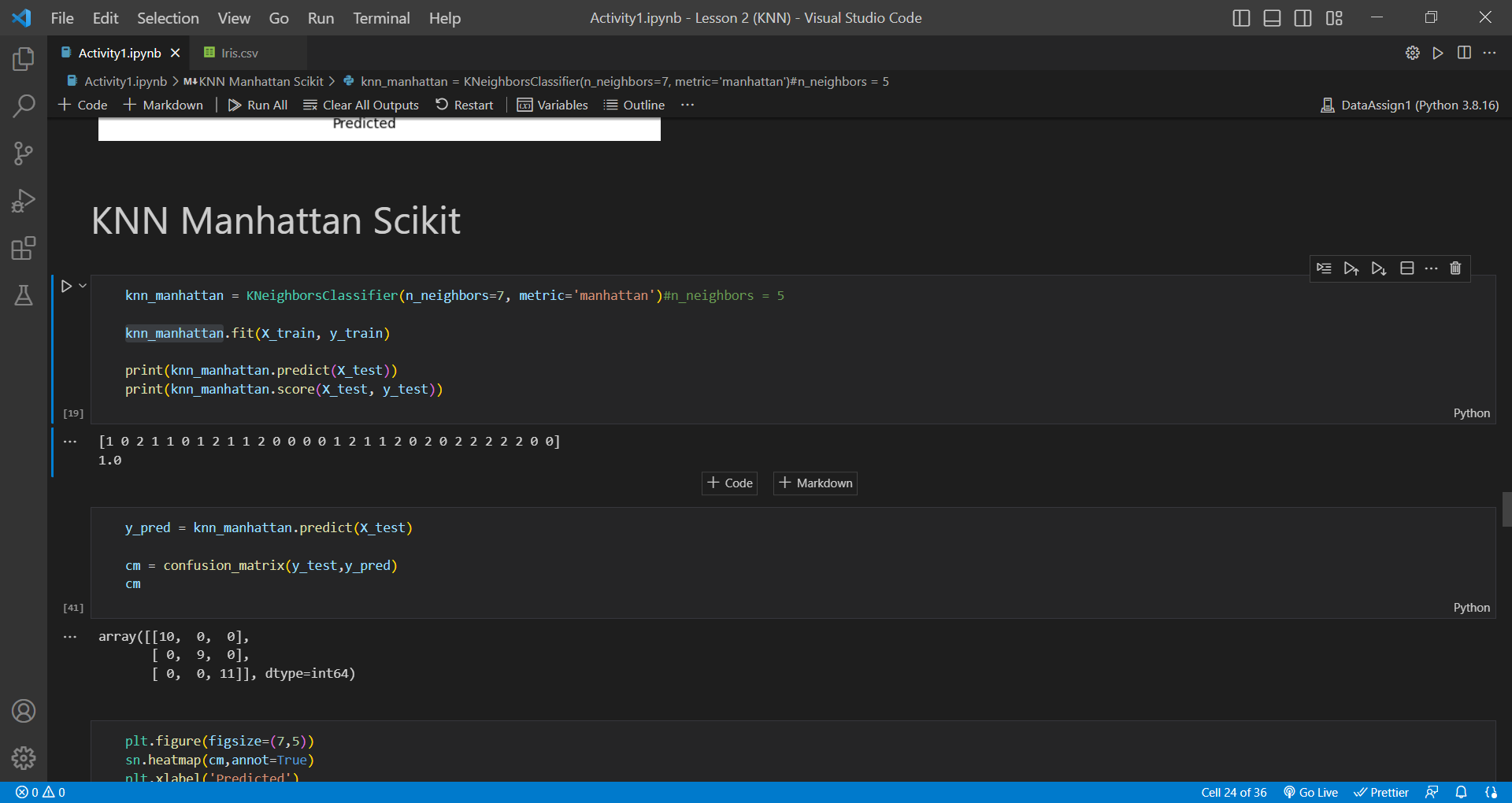


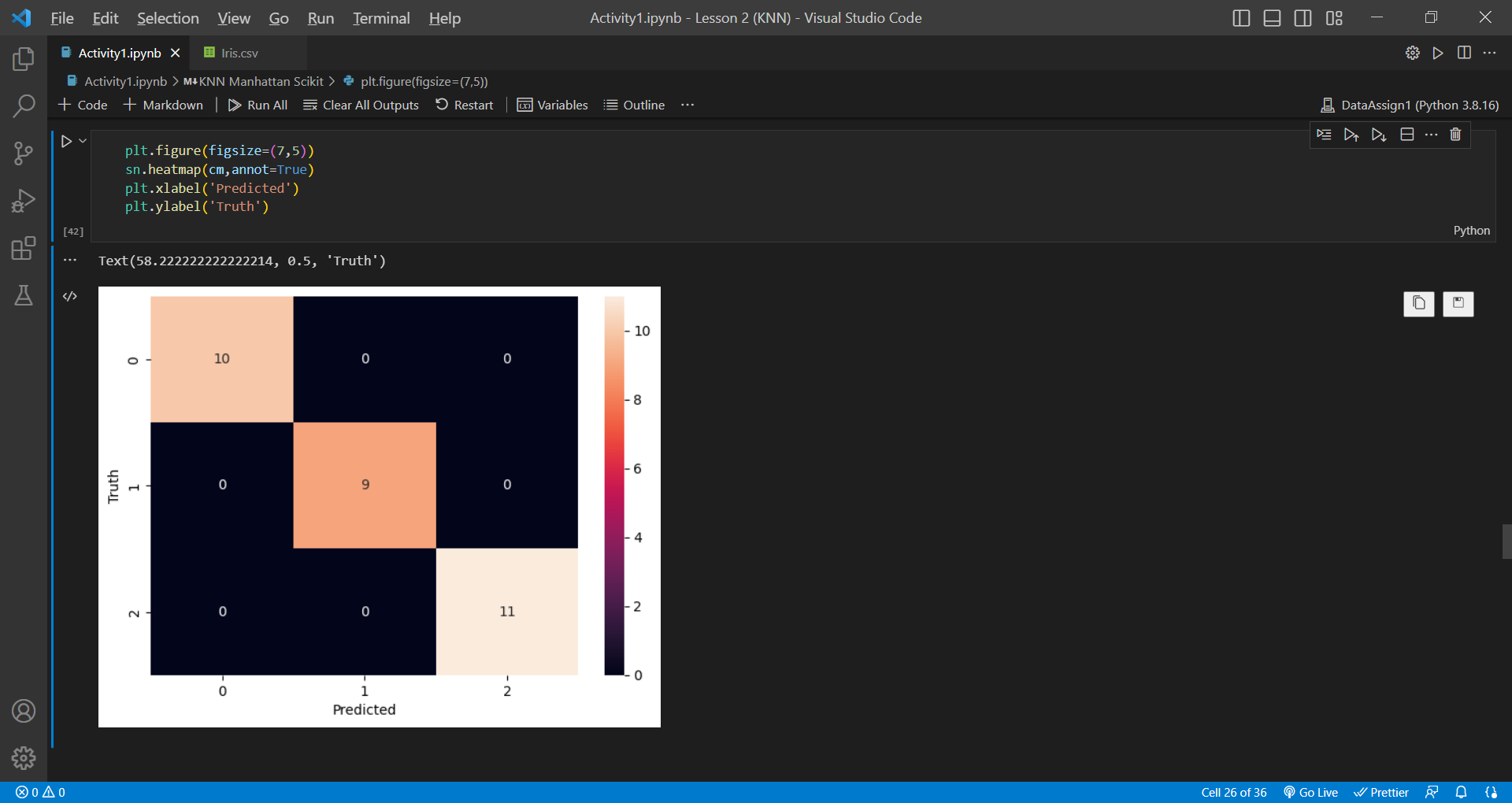


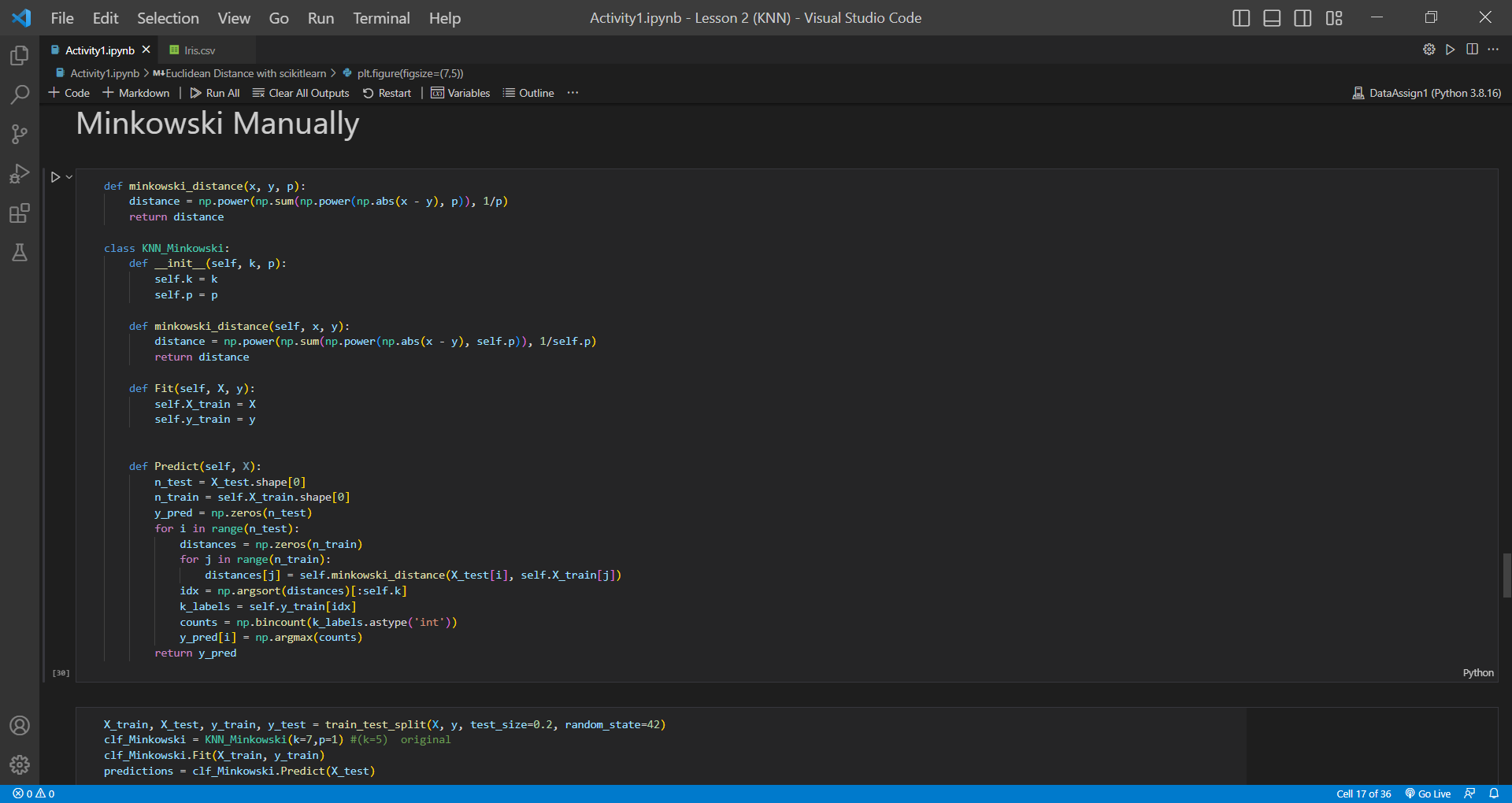


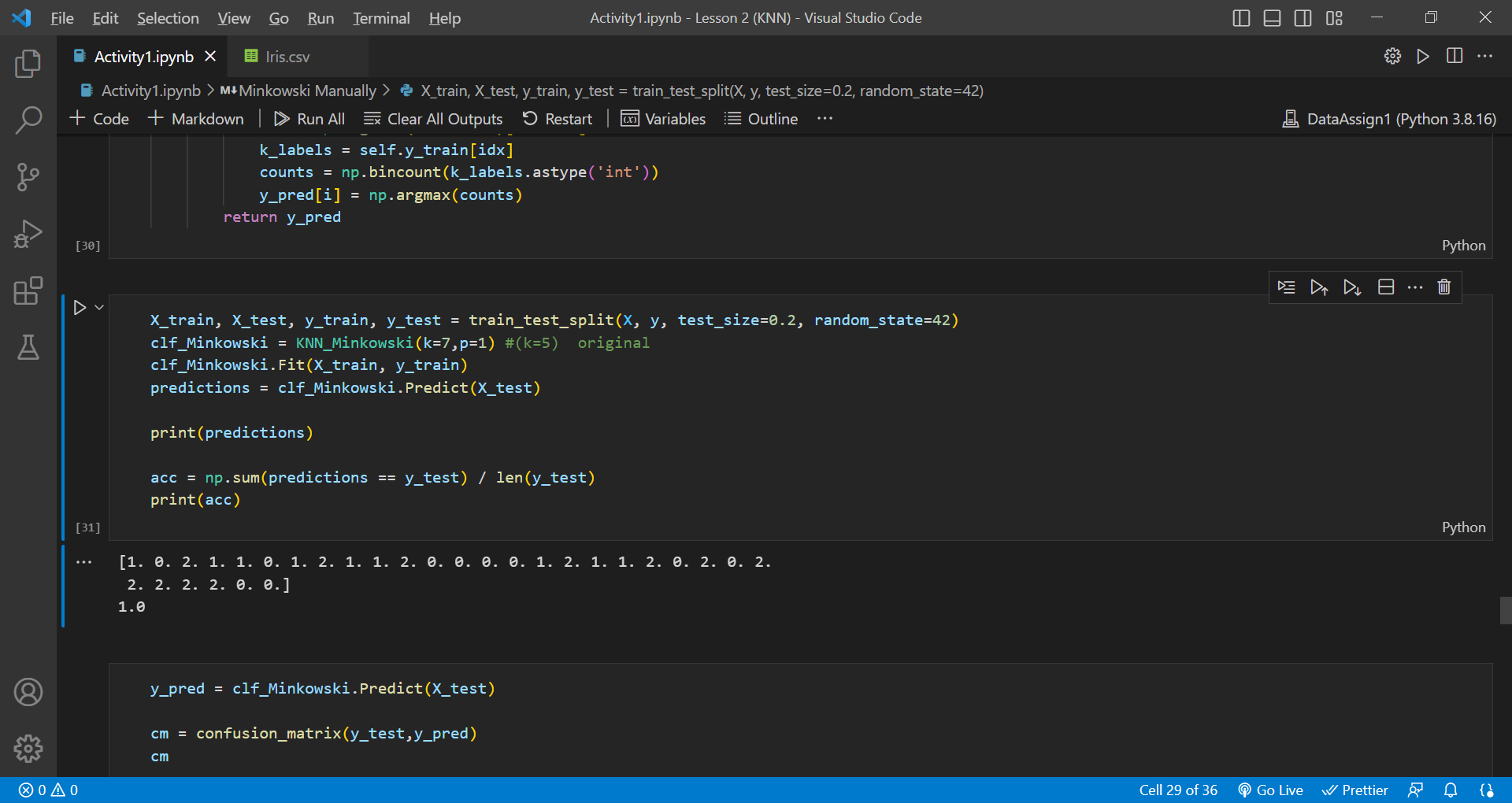


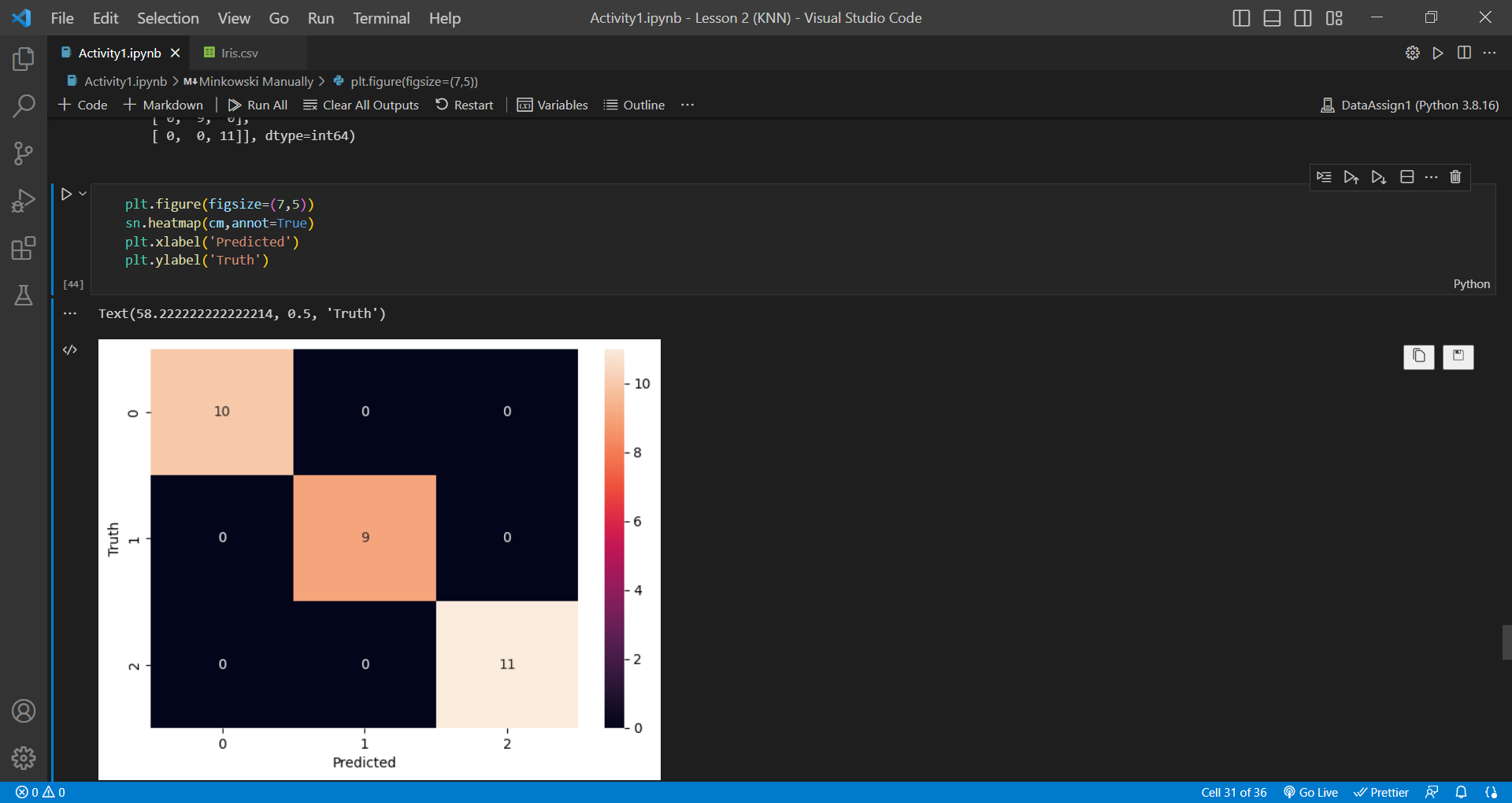


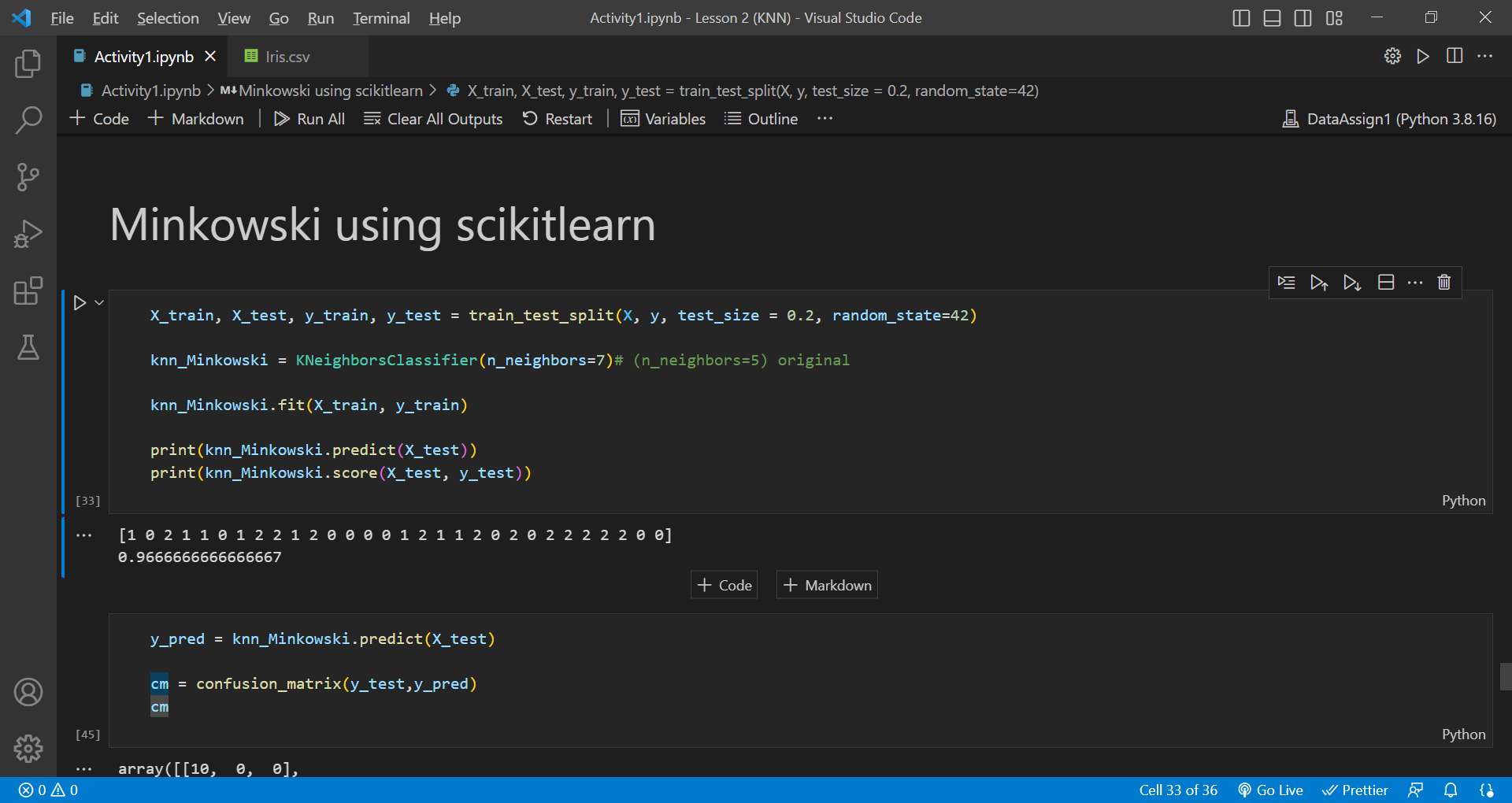




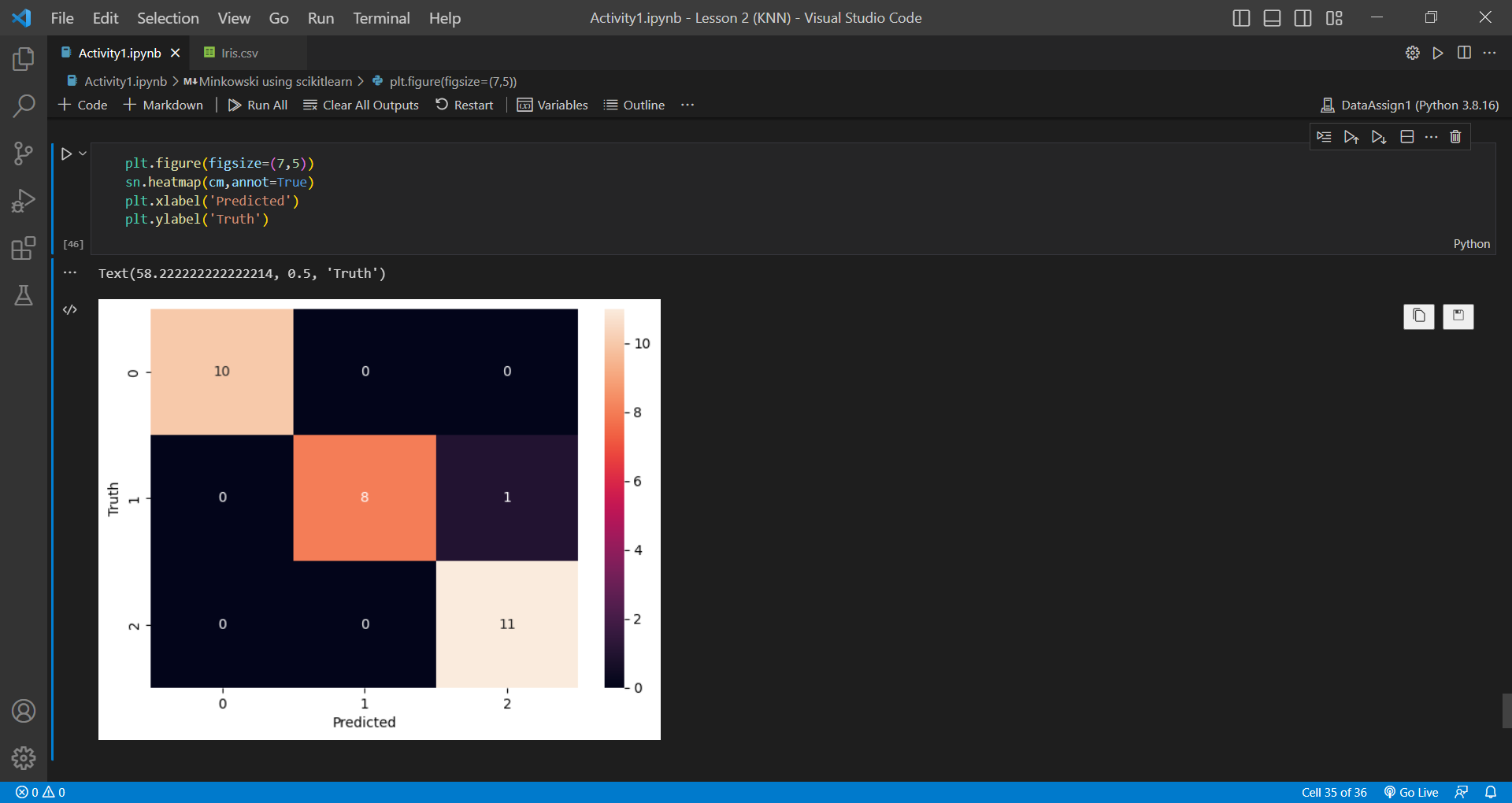








The bug is that the Minkowski distance using the SciKit Learn function has different accuracy than the manual Minkowski distance



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

NOT SCREENSHOT

|

V

X = data[['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm']].values

y = data['Species'].values

X is the independent values or the features of the model.

y is the dependent values or the labels/output of the model

data['Species'] = data['Species'].apply(lambda x: 0 if x == 'Iris-setosa' else 1 if x == 'Iris-versicolor' else 2)

data['Species'].unique()

this code snippet is used to convert 'Iris-setosa' to 0, 'Iris-versicolor' to 1 and else or 'Iris-virginica' to 2

MANUAL CODES

clf = KNN(k=7)#(k=5) original

clf.Fit(X\_train, y\_train)

predictions = clf.Predict(X\_test)

print(predictions)

The code snippet above is used by the manual codes for Euclidean, Manhattan and Minkowski methods of KNN Model to train the datasets, and then get the predictions that classifies if Species are Iris-Setosa, Iris-Versicolor or Iris Virginica

acc = np.sum(predictions == y\_test) / len(y\_test)

print(acc)

This code snippet is used to get the accuracy of the model using the manual codes for Euclidean, Manhattan and Minkowski methods of KNN Model.

SCIKIT CODES:

knn\_ = KNeighborsClassifier(n\_neighbors=7, metric='')

**n\_neighbors** is the parameter Number of neighbors to use for **[kneighbors](https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html" \l "sklearn.neighbors.KNeighborsClassifier.kneighbors" \o "sklearn.neighbors.KNeighborsClassifier.kneighbors)** queries.

**metric** is the parameter to use for distance computation. Default is “minkowski” but it can also be changed to “manhattan” or “euclidean” to get the results from those methods

knn\_.fit(X\_train, y\_train)

This code snippet is used to train the dataset for the model.

print(knn\_.predict(X\_test))

print(knn\_.score(X\_test, y\_test))

**.predict(X\_test)** is the method in Python is used to make predictions using a trained machine learning model.

**.score(X\_test, y\_test)** method returns a score that indicates how well the model was able to predict the values in the test data.

y\_pred = clf\_Manhattan.Predict(X\_test)

cm = confusion\_matrix(y\_test,y\_pred)

cm

and

plt.figure(figsize=(7,5))

sn.heatmap(cm,annot=True)

plt.xlabel('Predicted')

plt.ylabel('Truth')

Both of the code snippets are used to show the confusion matrix. The second code snippet shows an actual graph while the first only prints a matrix without a graph. The confusion matrix can be used to evaluate the performance of a machine learning model by providing information about the number of correct and incorrect predictions made by the model.

The output in the diagonal line of the confusion matrix represents the number of correct predictions made by the model. The off-diagonal elements of the confusion matrix represent the number of incorrect predictions made by the model. The diagonal line of the confusion matrix can be used to calculate the overall accuracy of the model, while the off-diagonal elements can be used to calculate the false positive and false negative rates.

RESULTS:

As seen from the prediction, accuracy and confusion matrix each model shows that the both the manual and Sci-kit codes models are performing well. With the exception of the Minkowski Distance manual and Sci-kit codes producing different accuracies and the Sci-kit code producing one incorrect prediction in its confusion matrix, all the models are performing well.

<https://www.figma.com/file/DyV97k19rKBqb0NXRiwinA/Activity0-and-Activity1_KNN-flowchart?type=whiteboard&node-id=0%3A1&t=6BnNaepQYN390yzs-1>