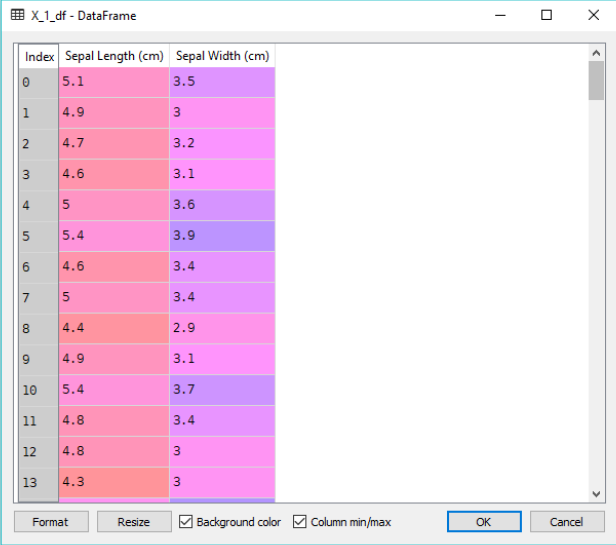


Week 5 Assignment – Classification and Clustering

Brian Mallari

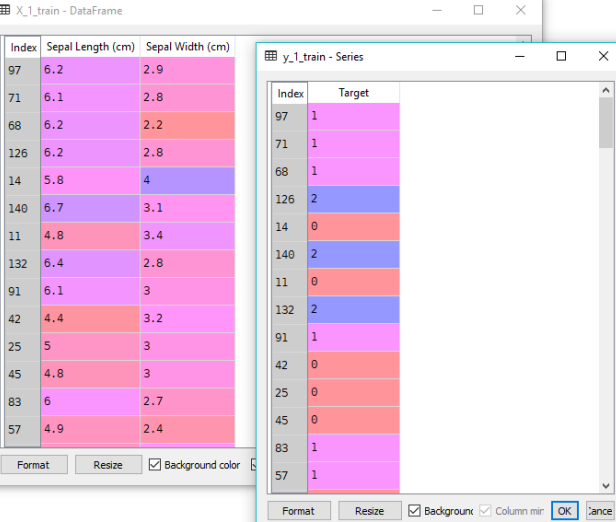
Part 1 – Classification

The first two columns from the iris dataset (as a data frame):



Index	Sepal Length (cm)	Sepal Width (cm)
0	5.1	3.5
1	4.9	3
2	4.7	3.2
3	4.6	3.1
4	5	3.6
5	5.4	3.9
6	4.6	3.4
7	5	3.4
8	4.4	2.9
9	4.9	3.1
10	5.4	3.7
11	4.8	3.4
12	4.8	3
13	4.3	3

70% of the extracted data for training:



Index	Sepal Length (cm)	Sepal Width (cm)
97	6.2	2.9
71	6.1	2.8
68	6.2	2.2
126	6.2	2.8
14	5.8	4
140	6.7	3.1
11	4.8	3.4
132	6.4	2.8
91	6.1	3
42	4.4	3.2
25	5	3
45	4.8	3
83	6	2.7
57	4.9	2.4

Index	Target
97	1
71	1
68	1
126	2
14	0
140	2
11	0
132	2
91	1
42	0
25	0
45	0
83	1
57	1

30% of the extracted data for testing:

Index	Sepal Length (cm)	Sepal Width (cm)
37	4.9	3.1
102	7.1	3
141	6.9	3.1
76	6.8	2.8
113	5.7	2.5
34	4.9	3.1
111	6.4	2.7
137	6.4	3.1
65	6.7	3.1
58	6.6	2.9
54	6.5	2.8
79	5.7	2.6
66	5.6	3
17	5.1	3.5

Index	Target
37	0
102	2
141	2
76	1
113	2
34	0
111	2
137	2
65	1
58	1
54	1
79	1
66	1
17	0

Allocation of the data points between the training and testing sets:

Name	Type	Size	Value
X_1_df	DataFrame	(150, 2)	Column names: Sepal Length (cm), Sepal Width (cm)
X_1_test	DataFrame	(45, 2)	Column names: Sepal Length (cm), Sepal Width (cm)
X_1_train	DataFrame	(105, 2)	Column names: Sepal Length (cm), Sepal Width (cm)
y_1	DataFrame	(150, 1)	Column names: Target
y_1_test	Series	(45,)	Series object of pandas.core.series module
y_1_train	Series	(105,)	Series object of pandas.core.series module

Resulting accuracies from kNN and three variations of SVM:

```
Using n_neighbors = 3:  
Number of correct predictions with kNN = 32  
Accuracy of kNN model = 0.711111111111  
  
Using n_neighbors = 5:  
Number of correct predictions with kNN = 36  
Accuracy of kNN model = 0.8  
  
Using n_neighbors = 7:  
Number of correct predictions with kNN = 36  
Accuracy of kNN model = 0.8  
  
Number of correct predictions with linear SVM = 35  
Accuracy of linear SVM model = 0.777777777778  
  
Number of correct predictions with radial basis function (RBF) SVM = 35  
Accuracy of RBF SVM model = 0.777777777778  
  
Number of correct predictions with polynomial (poly) SVM = 35  
Accuracy of poly SVM model = 0.777777777778
```

Python code for Part 1:

Week 5, Part 1 - Classification

```
import pandas as pd  
import numpy as np  
  
# for the kNN classification model  
from sklearn.neighbors import KNeighborsClassifier  
# from sklearn.cross_validation import train_test_split  
from sklearn.model_selection import train_test_split # using a new module in sklearn  
  
# for the iris dataset  
from sklearn import datasets  
iris = datasets.load_iris()  
  
# import the package  
from sklearn.svm import SVC  
  
# Inputs/features/predictors: X_1  
# Outcome/response: y_1  
# Split: train, test  
  
# extract the first two columns from the data set, along with the target information  
X_1 = iris.data[:, :2]  
X_1 = pd.DataFrame(X_1, columns = ["Sepal Length (cm)", "Sepal Width (cm)"])  
X_1_df = X_1
```

```

y_1 = iris.target
y_1 = pd.DataFrame(y_1, columns = ["Target"])

# 70% data for training, 30% for testing
X_1_train, X_1_test, y_1_train, y_1_test = train_test_split(
    X_1[['Sepal Length (cm)', 'Sepal Width (cm)']], y_1["Target"], test_size = 0.3 )

# kNN model with n_neighbors = 3
classifier_1_1a = KNeighborsClassifier(n_neighbors = 3)
classifier_1_1a.fit(X_1_train, y_1_train)
print("\nUsing n_neighbors = 3:")

prediction_1_1a = classifier_1_1a.predict(X_1_test)

correct_1_1a = np.where(prediction_1_1a == y_1_test, 1, 0).sum()
print("Number of correct predictions with kNN =", correct_1_1a)

accuracy_1_1a = correct_1_1a/len(y_1_test)
print("Accuracy of kNN model =", accuracy_1_1a)

# kNN model with n_neighbors = 5
classifier_1_1b = KNeighborsClassifier(n_neighbors = 5)
classifier_1_1b.fit(X_1_train, y_1_train)
print("\nUsing n_neighbors = 5:")

prediction_1_1b = classifier_1_1b.predict(X_1_test)

correct_1_1b = np.where(prediction_1_1b == y_1_test, 1, 0).sum()
print("Number of correct predictions with kNN =", correct_1_1b)

accuracy_1_1b = correct_1_1b/len(y_1_test)
print("Accuracy of kNN model =", accuracy_1_1b)

# kNN model with n_neighbors = 7
classifier_1_1c = KNeighborsClassifier(n_neighbors = 7)
classifier_1_1c.fit(X_1_train, y_1_train)
print("\nUsing n_neighbors = 7:")

prediction_1_1c = classifier_1_1c.predict(X_1_test)

correct_1_1c = np.where(prediction_1_1c == y_1_test, 1, 0).sum()
print("Number of correct predictions with kNN =", correct_1_1c)

accuracy_1_1c = correct_1_1c/len(y_1_test)
print("Accuracy of kNN model =", accuracy_1_1c)

# linear SVC classifier
classifier_1_2a = SVC(kernel = "linear")

```

```

classifier_1_2a.fit(X_1_train, y_1_train)

prediction_1_2a = classifier_1_2a.predict(X_1_test)

correct_1_2a = np.where(prediction_1_2a == y_1_test, 1, 0).sum()
print("\nNumber of correct predictions with linear SVM =", correct_1_2a)

accuracy_1_2a = correct_1_2a/len(y_1_test)
print("Accuracy of linear SVM model =", accuracy_1_2a)

# radial basis function (RBF) SVC classifier
classifier_1_3a = SVC(kernel = "rbf")
classifier_1_3a.fit(X_1_train, y_1_train)

prediction_1_3a = classifier_1_3a.predict(X_1_test)

correct_1_3a = np.where(prediction_1_3a == y_1_test, 1, 0).sum()
print("\nNumber of correct predictions with radial basis function (RBF) SVM =", correct_1_3a)

accuracy_1_3a = correct_1_3a/len(y_1_test)
print("Accuracy of RBF SVM model =", accuracy_1_3a)

# polynomial SVC classifier
classifier_1_4a = SVC(kernel = "poly")
classifier_1_4a.fit(X_1_train, y_1_train)

prediction_1_4a = classifier_1_4a.predict(X_1_test)


correct_1_4a = np.where(prediction_1_4a == y_1_test, 1, 0).sum()
print("\nNumber of correct predictions with polynomial (poly) SVM =", correct_1_4a)

accuracy_1_4a = correct_1_4a/len(y_1_test)
print("Accuracy of poly SVM model =", accuracy_1_4a)

```

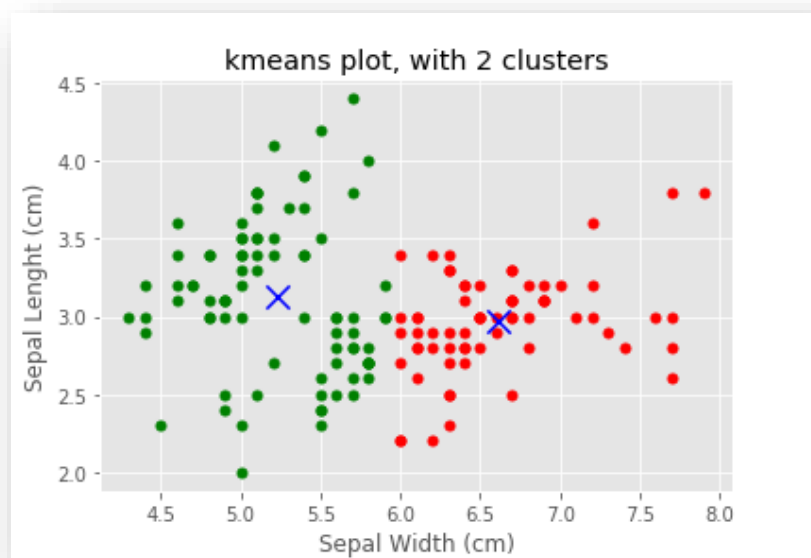
Part 2 – Clustering

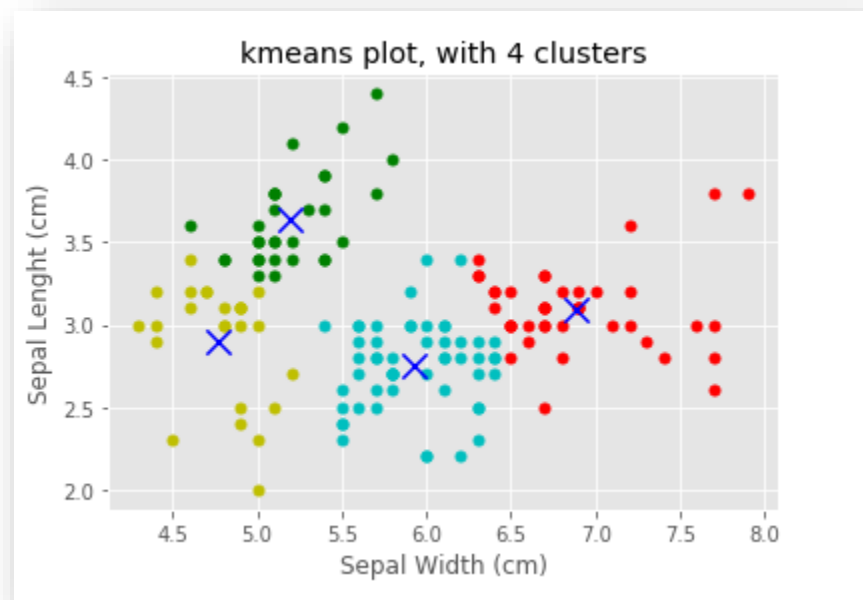
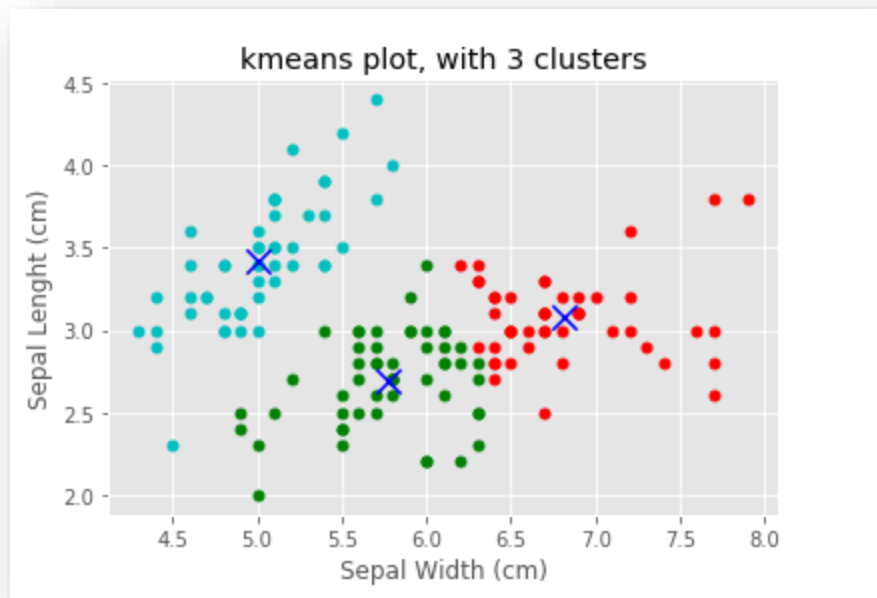
The first two columns from the iris dataset again (this time as an array):



	0	1
0	5.1	3.5
1	4.9	3
2	4.7	3.2
3	4.6	3.1
4	5	3.6
5	5.4	3.9
6	4.6	3.4
7	5	3.4
8	4.4	2.9
9	4.9	3.1
10	5.4	3.7
11	4.8	3.4
12	4.8	3

Plots with clusters using different numbers of clusters:





Python code for Part 2:

Week 5, Part 2 - Clustering

import matplotlib.pyplot as plt

for ggplot

from matplotlib import style

```

style.use("ggplot")

# for the iris dataset
from sklearn import datasets
iris = datasets.load_iris()

# for the KMeans function
from sklearn.cluster import KMeans

# extract the first two columns again from the iris data set
X_2 = iris.data[:, :2]
X_2_array = X_2

# n_clusters = 2
kmeans_a = KMeans(n_clusters = 2)
kmeans_a.fit(X_2)

centroids_a = kmeans_a.cluster_centers_
labels_a = kmeans_a.labels_

colors = ["g.", "r.", "c.", "y."]

for i in range(len(X_2)):
    plt.plot(X_2[i][0], X_2[i][1], colors[labels_a[i]], markersize = 10)

plt.scatter(centroids_a[:, 0], centroids_a[:, 1], marker = "x", s = 150,
            linewidths = 2, zorder = 10, color = 'blue')
plt.title("kmeans plot, with 2 clusters")
plt.ylabel("Sepal Length (cm)")
plt.xlabel("Sepal Width (cm)")
plt.show()

# n_clusters = 3
kmeans_b = KMeans(n_clusters = 3)
kmeans_b.fit(X_2)

centroids_b = kmeans_b.cluster_centers_
labels_b = kmeans_b.labels_

colors = ["g.", "r.", "c.", "y."]

for i in range(len(X_2)):
    plt.plot(X_2[i][0], X_2[i][1], colors[labels_b[i]], markersize = 10)

plt.scatter(centroids_b[:, 0], centroids_b[:, 1], marker = "x", s = 150,
            linewidths = 2, zorder = 10, color = 'blue')
plt.title("kmeans plot, with 3 clusters")
plt.ylabel("Sepal Length (cm)")

```



```

plt.xlabel("Sepal Width (cm)")
plt.show()

# n_clusters = 4
kmeans_c = KMeans(n_clusters = 4)
kmeans_c.fit(X_2)

centroids_c = kmeans_c.cluster_centers_
labels_c = kmeans_c.labels_

colors = ["g.", "r.", "c.", "y."]

for i in range(len(X_2)):
    plt.plot(X_2[i][0], X_2[i][1], colors[labels_c[i]], markersize = 10)

plt.scatter(centroids_c[:, 0], centroids_c[:, 1], marker = "x", s = 150,
            linewidths = 2, zorder = 10, color = 'blue')
plt.title("kmeans plot, with 4 clusters")
plt.ylabel("Sepal Length (cm)")
plt.xlabel("Sepal Width (cm)")
plt.show()

```

Thoughts on the data and techniques:

I liked working with the iris dataset. There were enough data points in the dataset to practice splitting the data between training and testing sets. Also, the data points themselves exhibited enough scatter to prevent a 100% accuracy for any of the classifying methods, which I think would be the most common scenario that I'll encounter when classifying data in the real world.

As for the techniques, I think it's pretty cool how there are different ways to perform classifying. I myself am most familiar with kNN as a function of this week's lesson; however, some awareness of SVM would help in the event that I find myself working with someone who's more inclined to work with that instead of kNN. I'll have to take some time to understand the mechanics of SVM, but I think it would be a worthwhile endeavor on my journey to become a capable data scientist.

References

1. Code for Part 1 of this assignment was based off of code featured in the video podcast, ML with Python - Introduction and Classification, by Dr. Chirag Shah, as well as the assignment description on Sakai also by Dr. Chirag Shah
2. Code for Part 2 of this assignment was based off of code featured in the video podcast, ML with Python - Clustering, by Dr. Chirag Shah
3. Information on extracting data from the iris dataset taken from the following website:
http://scikit-learn.org/stable/auto_examples/datasets/plot_iris_dataset.html
4. Information on radial basis function from the following website:
https://en.wikipedia.org/wiki/Radial_basis_function
5. Information on the 'rbf' kernel for the SVC function taken from the following website:
<http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>
6. Information on an error message that arises when running SVM code taken from the following website: <https://stackoverflow.com/questions/41238769/warning-messages-when-using-python>
7. Information regarding the changeover from the cross_validation module in scikit-learn to the model_selection model taken from the following website:
<https://github.com/EpistasisLab/tpot/issues/284>
8. Information on labeling plots taken from the following website:
https://matplotlib.org/users/pyplot_tutorial.html
9. Information to change the color of the cluster markers taken from the following website:
<https://stackoverflow.com/questions/12236566/setting-different-color-for-each-series-in-scatter-plot-on-matplotlib>