**Exercise 1:** Data Normalization, Classification and application of PCA

In this exercise we will experiment ideas learnt during the section “*Theory of Machine Learning*” using scikit-learn open source package. We will use classification of Iris flowers based on sepal and petal, width and length as our data. This data set is bundled in to the scikit-learn and can be loaded using load\_iris() API of scikit-learn. Data is also pre-classified and hence can be used with supervised learning. Also, this data is publicly available at <http://archive.ics.uci.edu/ml/datasets/Iris>

As an added note, <http://archive.ics.uci.edu/ml/datasets.html>, provide various data sets that you may use to experiment various machine learning models.

1. Load the data set and identify names of features and number of classes and name of the classes.

* Classes

Setosa, versicolor, virginica

* Features

Sepal length, Sepal width, Petal length, petal width

1. How many samples are available? How many classes ?

Samples: 150

Classes: 3

1. Normalize the data using StandardScaler
2. Divide 80% of data to training set and 20% to test set.
3. Go to scikit-learn API page and study the format and usage of KNeighborsClassifier
4. When using KNeighborsClassifier, what is the optimal number of neighbors for the Iris data set.
5. Train KNeighborsClassifier for 80% of Iris data set
6. What is the score (accuracy) of the KNeighborsClassifier classifier
7. Generate the classification report and review f1-score and precision
8. Change number of neighbors and see how accuracy changes
9. Repeat the above classification with VotingClassifier that uses LogisticRegression, GaussianNB and RandomForest as estimators
10. Generate the score and classification repots and compare that with that of the KNeighborsClassifier
11. Go to scikit-learn API page and study PCA
12. How do you find the variance ratio of each dimension?
    1. Make n\_components = <number of features>
    2. Fit PCA
    3. pca.explained\_variance\_ratio\_ gives the variance ratio of each feature
    4. Note that few features carry most of the variance
    5. Sample output of feature variances for Iris data set is below, as you may note 4th dimension has very little effect
       1. [0.88111947 0.07704014 0.03612976 0.00571063]
13. Transform Iris data set to a 3-dimensional space
14. Train KNeighborsClassifier with the new transformed data
15. Generate score (accuracy) and compare that with #8
16. What is the variance ratio of the most influential dimension?
17. Repeat 14 with number of components set 1
18. Generate score (accuracy) and compare that with #16

Sample code for Steps 1 – 10

**from** sklearn.datasets **import** load\_iris  
**from** sklearn.preprocessing **import** StandardScaler  
**from** sklearn.model\_selection **import** train\_test\_split  
**from** sklearn.neighbors **import** KNeighborsClassifier  
**from** sklearn.metrics **import** classification\_report  
  
data=load\_iris()  
  
X= data.data  
y=data.target  
s=StandardScaler()  
*#We are only transforming data not the labels*X1=s.fit\_transform(X)  
X1\_tr, X1\_tst, y\_tr, y\_tst = train\_test\_split(X1,y, shuffle=**True**, random\_state=32)  
*#There are 4 features*knn = KNeighborsClassifier(n\_neighbors=5)  
knn.fit(X1\_tr, y\_tr)  
sc=knn.score(X1\_tst, y\_tst)  
print(**'Score '**, sc)  
y\_pred = knn.predict(X1\_tst)  
c = classification\_report(y\_tst, y\_pred)  
  
print(**' Confusion Matrix '**, c )

Sample code for steps 11 – 12

**from** sklearn.linear\_model **import** LogisticRegression  
**from** sklearn.naive\_bayes **import** GaussianNB  
**from** sklearn.ensemble **import** RandomForestClassifier, VotingClassifier  
**from** sklearn.datasets **import** load\_iris  
**from** sklearn.preprocessing **import** StandardScaler  
**from** sklearn.model\_selection **import** train\_test\_split  
**from** sklearn.metrics **import** classification\_report  
  
data=load\_iris()  
  
X= data.data  
y=data.target  
s=StandardScaler()  
*#We are only transforming data not the labels*X1=s.fit\_transform(X)  
X1\_tr, X1\_tst, y\_tr, y\_tst = train\_test\_split(X1,y, shuffle=**True**, random\_state=32)  
  
clf1 = LogisticRegression(random\_state=1)  
clf2 = RandomForestClassifier(random\_state=1)  
clf3 = GaussianNB()  
  
est = [(**'LR'**, clf1), (**'RF'**, clf2), (**'GB'**, clf3)]  
  
vclf = VotingClassifier(estimators=est, voting=**'hard'**)  
  
vclf.fit(X1\_tr, y\_tr)  
*# based on the version you are using you will see a warning on empty array. It is a bugs you can ignore  
#https://github.com/scikit-learn/scikit-learn/issues/10449*y\_pred = vclf.predict(X1\_tst)  
  
s = vclf.score(X1\_tst, y\_tst)  
print(**'Score '**, s)  
  
c = classification\_report(y\_tst, y\_pred)  
  
print(**'Classification report '**, c)

Sample code for steps 13 - 20

**from** sklearn.datasets **import** load\_iris  
**from** sklearn.model\_selection **import** train\_test\_split  
**from** sklearn.neighbors **import** KNeighborsClassifier  
  
**from** sklearn.decomposition **import** PCA  
**from** sklearn.preprocessing **import** StandardScaler, MinMaxScaler  
  
*# Load the bundled iris flower data set*data = load\_iris()  
X= data.data  
y=data.target

# Try different normalizers and see the effect on final accuracy  
s=StandardScaler()  
s=MinMaxScaler()  
*#We are only transforming data not the labels*X1=s.fit\_transform(X)  
*#X1=X*

*# shuffle the data and split 50-50*X\_tr, X\_tst, y\_tr, y\_tst = train\_test\_split(X1, y, shuffle=**True**, test\_size=0.8, random\_state=32)  
  
  
  
pca = PCA(n\_components=3)  
pca.fit(X\_tr)  
print(**'Feature Variance ratio '**, pca.explained\_variance\_ratio\_)  
X\_tr\_pca = pca.transform(X\_tr)  
X\_tst\_pca = pca.transform(X\_tst)  
  
knn = KNeighborsClassifier(n\_neighbors=5)  
knn.fit(X\_tr\_pca, y\_tr)  
  
*# accuracy compared to predicted labels*score = knn.score(X\_tst\_pca, y\_tst)  
print(**"Accuracy "**, score)