CS3481 Fundamentals of Data Science

Assignment 2

Hyder Ali (54028087)

(a)
For my train/test split, I did a 70/30 % partition. Thus, training set had 70% of the data and test set had 30% of the data of the original set. Random forest models are constructed and tested using different number of component trees based on the same train/test split. The change in performance is evaluated among the different models. I created 6 models in total. The parameter, n_estimators, indicate number of trees and for the 6 models, they are set to, 5, 10, 20, 30, 50 and 100. These numbers are chosen arbitrarily to get the overall picture how increasing the number of component trees affect the performance of the random forest. Below shows the summary of the result.

Model	n_estimator	Accuracy Score (%)
1	5	84
2	10	88
3	20	86
4	30	86
5	50	88
6	100	89

As seen from the table above, as the number of trees (n_estimator) increase, the general trend is that, the accuracy score would increase. If the algorithm is run again, slightly different accuracy score will

result for all of the models as the random forest depends heavily on the random values chosen at subset and training data. Due to this, the models with closer value of n_estimator have a more overlapping accuracy score, however if the algorithm is run many times, the general trend I have seen is that, the larger the number of component trees (larger n_estimator value), the higher the accuracy score will be.

(b)

The random forest, which has the best classification performance, is the one with 100 component trees (n_estimators=100) with the accuracy score of approximately 89%. For this random forest, I ran the algorithm again and also tested the performance of 3 component trees. As I mentioned earlier, running the algorithm a different time will yield a slightly different result due to the random values used in the algorithm. Thus, for this part, when I ran the algorithm again, the random forest with 100 component trees (n_estimators=100) gave an accuracy score of approximately 88%. The 3 component trees chosen arbitrarily are the 11th, 51th and the 91th component trees. The classification performance of these component trees is compared with the original random forest, which they belong to. The result is shown below.

Model	Tree (th)	**Accuracy Score (%)**
Random Forest	NA	88
Decision Tree	11	81
Decision Tree	51	82
Decision Tree	91	80

The classification performance for all the trees is lower than the original random forest model they belong to. And this is as predicted as the random forest utilizes the collective accuracy of all the trees and minimizes over-fitting on unseen data by providing generalization through utilizing all the different trees so it must have higher accuracy in general.

(c)

For the selected trees in (b) and for the overall random forest, the feature importance values is obtained and compared to understand and analyze how the 3 selected trees contribute to the overall classification performance resultant from the random forest model (n_estimators=100). Below shows the results.

11th tree:

[0.15054607 0.0223551 0.29845983 0.08816092 0.16113914 0.279
33893]

51th tree:

[0.0620108 0.06386526 0.09143613 0.09266998 0.06607688 0.623
94094]

91th tree:

[0.30618497 0.11535975 0.07547124 0.0825032 0.19577095 0.224
70989]

Original random forest (n_estimators=100):
[0.1396279 0.09276069 0.11922486 0.11911471 0.12320436 0.406
06747]

All the above rows are with respect to the same columns and thus it's not important to mention the column names. I will refer to the columns in terms of the number to which they are ordered from left to right (e.g. 1st,2nd,etc.). As seen from the data above, in the original random forest, the last attribute, 6th column, has the highest importance / contribution in classification performance. And it's much higher than the rest of the attributes. Thus, it is expected that, the trees would display this characteristics too. However, only 51th tree truly displays this characteristic. 51th tree also has the highest accuracy score so it comes to show how having this characteristic can help to influence to improve the overall accuracy. The 2nd attribute from the random forest has the least importance, and more or less all the trees shows this similar characteristic. However, it is shown that the reason why 11th and 51th trees perform better than 91th tree is because in 11th and 51th give less importance to 2nd attributes with respect to other attributes than 91th tree does. So 11th and 51th tree perform better as it can recognize which attribute is not as helpful. The 1st, 3rd, 4th and 5th attributes' feature importance values of the random forest are relatively similar as see from the result above. 11th and 51th tree both maintain this characteristic more properly than 91th tree does, as seen in the 91th tree, the values have more variance among them. Thus, the performance of 11th and 51th tree rightly is higher than that of 91th tree and helps to influence the final performance of the random forest. Among 11th and 51th trees, 51th tree maintains this characteristic more readily and thus can be seen to have better performance accuracy. In conclusion, the tree, which generalizes and give similar picture/result of that of the random

forest, will likely have higher accuracy score. This as a result will influence the performance accuracy of the random forest.

(d)

A Gaussian naïve Bayes classifier is built using the same original data. Gaussian distribution is used to handle the continuous nature of the data. The accuracy score obtained from this classifier is approximately 87% accuracy, which is very similar to the 89% accuracy on the first run and 88% accuracy on the second run using random forest algorithm. Although it still shows a little bit less accurate than random forest, but I argue that, naïve Bayes classifier is better as it is a simpler, faster and more efficient compared to random forest which needs to build many tress and many nodes in each trees recursively and each tree can grow very large especially when the maximum depth is not set. Also, unlike random forest, which depends heavily on random values to choose feature subset and training dataset, naïve Bayes classifier did not use any random value, and thus the accuracy score was consistent and it would not change no matter how many times the algorithm is re-run. So with regards to all the benefits, and only around 1-2% difference in accuracy between the two classifiers, I think Gaussian naïve Bayes classifier is doing a more impressive job for the current task.

In [590]:
```
import pandas as pd
import numpy as np
from sklearn import tree
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
import graphviz

# Assignment 2
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
```

In [591]:
```
data = pd.read_csv('../../Desktop/CS3481- Fundamentals of Data Scie
nce/vertebral_column_data/column_3C.dat', sep=' ', header=None)
data.columns = ['pelvic_incidence numeric', 'pelvic_tilt numeric',
'lumbar_lordosis_angle numeric', 'sacral_slope numeric', 'pelvic_ra
dius numeric','degree_spondylolisthesis numeric','class']
features = ['pelvic_incidence numeric', 'pelvic_tilt numeric', 'lum
bar_lordosis_angle numeric', 'sacral_slope numeric', 'pelvic_radius
numeric','degree_spondylolisthesis numeric']
classes = ['disk hernia (DH)', 'spondylolisthesis (SL)', 'normal (N
O)']
```

In [592]:
```
X=data.iloc[:,0:6].values
Y=data.iloc[:,6].values
#print (len(data))
#print (data.shape)
```

In [ ]:
```
```

In [593]:
```
# Random Forest
```

In [594]:
```
# (a)
```

In [595]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0
.3,random_state=10,shuffle=True)
```

In [596]:
```
RFclf = RandomForestClassifier(n_estimators=5)
RFclf = RFclf.fit(X_train, Y_train)
RFprediction = RFclf.predict(X_test)
print(accuracy_score(Y_test, RFprediction))
```

```
0.8387096774193549
```

In [597]:
```
RFclf = RandomForestClassifier(n_estimators=10)
RFclf = RFclf.fit(X_train, Y_train)
RFprediction = RFclf.predict(X_test)
print(accuracy_score(Y_test, RFprediction))
```

```
0.8817204301075269
```

In [598]:
```
RFclf = RandomForestClassifier(n_estimators=20)
RFclf = RFclf.fit(X_train, Y_train)
RFprediction = RFclf.predict(X_test)
print(accuracy_score(Y_test, RFprediction))
```

```
0.8602150537634409
```

In [599]:
```
RFclf = RandomForestClassifier(n_estimators=30)
RFclf = RFclf.fit(X_train, Y_train)
RFprediction = RFclf.predict(X_test)
print(accuracy_score(Y_test, RFprediction))
```

```
0.8602150537634409
```

In [600]:
```
RFclf = RandomForestClassifier(n_estimators=50)
RFclf = RFclf.fit(X_train, Y_train)
RFprediction = RFclf.predict(X_test)
print(accuracy_score(Y_test, RFprediction))
```

```
0.8817204301075269
```

In [601]:
```
RFclf = RandomForestClassifier(n_estimators=100)
RFclf = RFclf.fit(X_train, Y_train)
RFprediction = RFclf.predict(X_test)
print(accuracy_score(Y_test, RFprediction))
```

```
0.8924731182795699
```

In [ ]:
```
```

In [620]:
```
# (b)

# The best random forest is one with the most trees among all I tes
ted. --> 100 trees.
```

In [621]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0
.3,random_state=10,shuffle=True)
```

In [622]:
```
RFclf = RandomForestClassifier(n_estimators=100)
RFclf = RFclf.fit(X_train, Y_train)
print(RFclf.score(X_test,Y_test))
#RFprediction = RFclf.predict(X_test)
#print(accuracy_score(Y_test, RFprediction))
```

```
0.8817204301075269
```

In [646]:
```
# 11th component tree
DT1 = RFclf.estimators_[10]
DT1 = DT1.fit(X_train,Y_train)
print(DT1.score(X_test,Y_test))
```

```
0.8064516129032258
```

In [647]:
```
# 51th component tree
DT2 = RFclf.estimators_[50]
DT2 = DT2.fit(X_train,Y_train)
print(DT2.score(X_test,Y_test))
```

```
0.8172043010752689
```

In [666]:
```
# 91th component tree
DT3 = RFclf.estimators_[90]
DT3 = DT3.fit(X_train,Y_train)
print(DT3.score(X_test,Y_test))
```

```
0.7956989247311828
```

In [ ]:
```

```

In [631]:
```
# (c)
```

In [627]:
```
# for 11th tree -> n=10
```

In [632]:
```
print(features)
print(DT1.feature_importances_)    # the higher the value, the more
important the feature
```

```
['pelvic_incidence numeric', 'pelvic_tilt numeric', 'lumbar_lordos
is_angle numeric', 'sacral_slope numeric', 'pelvic_radius numeric'
, 'degree_spondylolisthesis numeric']
[0.15054607 0.0223551  0.29845983 0.08816092 0.16113914 0.27933893
]
```

In [633]:
```
# for 51th tree -> n=50
```

In [634]:
```
print(features)
print(DT2.feature_importances_)
```

```
['pelvic_incidence numeric', 'pelvic_tilt numeric', 'lumbar_lordos
is_angle numeric', 'sacral_slope numeric', 'pelvic_radius numeric'
, 'degree_spondylolisthesis numeric']
[0.0620108  0.06386526 0.09143613 0.09266998 0.06607688 0.62394094
]
```

In [635]:
```
# for 91th tree -> n=90
```

In [636]:
```
print(features)a
print(DT3.feature_importances_)
```

```
['pelvic_incidence numeric', 'pelvic_tilt numeric', 'lumbar_lordos
is_angle numeric', 'sacral_slope numeric', 'pelvic_radius numeric'
, 'degree_spondylolisthesis numeric']
[0.30618497 0.11535975 0.07547124 0.0825032  0.19577095 0.22470989
]
```

In [637]:
```
# the feature important of the complete random forest
```

In [638]:
```
print(features)
print(RFclf.feature_importances_)
```

```
['pelvic_incidence numeric', 'pelvic_tilt numeric', 'lumbar_lordos
is_angle numeric', 'sacral_slope numeric', 'pelvic_radius numeric'
, 'degree_spondylolisthesis numeric']
[0.1396279  0.09276069 0.11922486 0.11911471 0.12320436 0.40606747
]
```

In [ ]:
```
```

In [659]:
```
# (d) Naive Bayes Classifier
```

In [660]:
```
NBclf = GaussianNB()
```

In [661]:
```
NBclf = NBclf.fit(X_train, Y_train)
```

In [662]:
```
print(NBclf.score(X_test, Y_test))
#NBprediction = NBclf.predict(X_test)
#print(accuracy_score(Y_test, NBprediction))
```

```
0.8709677419354839
```

In [ ]:
```
```

In [ ]:
```
```