NEW YORK INSTITUTE OF TECHNOLOGY

DTSC 620: Statistics for Data Science (Spring 2023)

Project Assignment 2

Ramis Rawnak, ID: 1319750

Data: A data set containing two classes spam or ham, classifies email messages. The dataset has a total of 4601 instances and there are 57 attributes encoding the number of times some words and or characters appear.

Classification Task: Trained classifiers using the first 1000 instances and utilized the remaining 3601 for testing.

Metrics: Three classifiers i.e. Decision Tree, Gaussian Naïve Bayes, and Logistic Regression are fused using the majority voting rule .Then it is compared with the accuracy of the fused model with: AdaBoost Ensemble with Decision Trees as the base learner, and Random Forests.

Task: With the given sorted dataset, after being loaded into a data frame, the missing feature values can be filled in with the most popular value/ used values in each column. It is then split for training and testing sets and the features and target variables are also separated for both training and testing the sets.

Then the first classifier i.e. Decision Tree is used to produce as given below classification accuracy, and confusion matrix on the test instances.

1) **Decision Tree:** It is a type of supervised machine learning which is both used for classification and regression tasks. It is also a tree-like structure which shows a series of decisions and their possible outcomes. With the help of sklearn library, decision trees, classification of accuracy, and confusion matrix has been produced as shown below:

```
[31] spam_input = spam.drop("Class", axis = 1)
```

```
[32] spam_output = spam["Class"]
```

```
# Split for training and testing set
from sklearn.model_selection import train_test_split
train_spam_input = spam_input[:1000]
test_spam_input = spam_input[1000:]
train_spam_output = spam_output[:1000]
test_spam_output = spam_output[1000:]
```

```
#Decision Tree Classifier
clf_dt = DecisionTreeClassifier()
#Train the Decision Tree Classifier
clf_dt.fit(train_spam_input,train_spam_output)
# Predict the testing
Y_pred_dt = clf_dt.predict(test_spam_input)
# Classification Accuracy of the Decision Tree Classifier
acc_dt = accuracy_score(test_spam_output, Y_pred_dt)
print("Decision tree accuracy: ", acc_dt)
print(confusion_matrix(test_spam_output, Y_pred_dt))
```

```
Decision tree accuracy:  0.8764232157733963
[[1930  252]
 [ 193 1226]]
```

As seen above the classification accuracy of the **Decision Tree Classifier** is **87.64%**.

2. **Gaussian Naïve Bayes:** It is a classification technique which is based on the probabilistic approach and Gaussian distribution used in Machine Learning.

```
#Gaussian Naive Classifier
clf_gnb = GaussianNB()
#Train the Gaussian Naive Classifier
clf_gnb.fit(train_spam_input,train_spam_output)
# Predict the testing
Y_pred_gnb = clf_gnb.predict(test_spam_input)
# Classification Accuracy of the Gaussian Naive Classifier
acc_gnb = accuracy_score(test_spam_output, Y_pred_gnb)
print("Gaussian Naive accuracy: ", acc_gnb)
print(confusion_matrix(test_spam_output, Y_pred_gnb))
```

```
Gaussian Naive accuracy:  0.830047209108581
[[1657  525]
 [  87 1332]]
```

As seen above the classification accuracy of the **Gaussian Naïve Bayes Classifier** is **83.00%**.

2. **Logistic Regression:** It is a statistical model often used for classification and predictive analytics. Logistic regression estimates the probability of an event occurring, i.e. true or false, based on a given dataset of independent variables

```
#Logistic Regression
clf_lr = LogisticRegression()
#Train the Logistic Regression
clf_lr.fit(train_spam_input,train_spam_output)
# Predict the testing
Y_pred_lr = clf_lr.predict(test_spam_input)
# Classification Accuracy of the Gaussian Naive Classifier
acc_lr = accuracy_score(test_spam_output, Y_pred_lr)
print("Logistic Regression: ", acc_lr)
print(confusion_matrix(test_spam_output, Y_pred_lr))
```

```
Logistic Regression:  0.9050263815606776
[[2043  139]
 [ 203 1216]]
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

As seen above the classification accuracy of the **Logistic Regression** is **90.50%**.

Fusion Classifier with Majority voting rule:

```
# Fuse the classifier with Decision Tree, Gaussian Naive Bayes and Logistic Regression by using majority voting rule
clf_dt_model = DecisionTreeClassifier()
clf_dt_score = clf_dt_model.fit(train_spam_input,train_spam_output)
clf_gnb_model = GaussianNB()
clf_gnb_score = clf_gnb_model.fit(train_spam_input,train_spam_output)
clf_lr_model = LogisticRegression()
clf_lr_score = clf_lr_model.fit(train_spam_input,train_spam_output)
Fusion_model = VotingClassifier(estimators=[('DT', clf_dt_model), ('GNB', clf_gnb_model), ('LR', clf_lr_model)], voting='har
Fusion_score = Fusion_model.fit(train_spam_input,train_spam_output)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
```

```
[92] Y_pred = Fusion_score.predict(test_spam_input)
     accuracy_score(test_spam_output, Y_pred)
     print("Accuracy:",accuracy_score(test_spam_output, Y_pred))
     target_names = ['class 0', 'class 1']
     print(classification_report(test_spam_output, Y_pred, target_names=target_names, digits=5))
     print(confusion_matrix(test_spam_output, Y_pred))
```

```
Accuracy: 0.9389058594834768
              precision    recall  f1-score   support

     class 0    0.96670   0.93126   0.94865      2182
     class 1    0.89993   0.95067   0.92461      1419

    accuracy                        0.93891      3601
   macro avg    0.93332   0.94096   0.93663      3601
weighted avg    0.94039   0.93891   0.93917      3601

[[2032  150]
 [  70 1349]]
```

As seen above the classification accuracy is **93.89%**.

Here we can conclude that **Fusion Classifier with majority voting** rule is more accurate.

AdaBoost Ensemble with Decision Trees as the base learner

```
# AdaBoost Ensemble with Decision Trees as the base learner
abc = AdaBoostClassifier(n_estimators=200, base_estimator=DecisionTreeClassifier())
model = abc.fit(train_spam_input, train_spam_output)
Y_pred_abc = model.predict(test_spam_input)
abc_acc = accuracy_score(test_spam_output, Y_pred_abc)
print("Accuracy of AdaBoost Ensemble with Decision Trees as the base learner: {:.3f}%".format(abc_acc * 100 ))
```

```
Accuracy of AdaBoost Ensemble with Decision Trees as the base learner: 87.948%
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_base.py:166: FutureWarning: `base_estimator` was renamed to `estima
  warnings.warn(
```

As seen above the classification accuracy is **87.95%**.

Random Forest: It is a machine learning algorithm that combines the results of multiple decision trees to conclude with a single output. It is capable of handling both classification and regression problems.
Below is a Random Forest Classifier using 1000 as base learners and max_features="auto":

```
#Random Forest Classifier
clf_rf = RandomForestClassifier(n_estimators=1000,max_features="auto")
#Train the Random Tree Classifier
clf_rf.fit(train_spam_input, train_spam_output)
# Predict the testing
Y_pred_rf = clf_rf.predict(test_spam_input)
# Classification Accuracy of the Random Forest Classifier
acc_rf = accuracy_score(test_spam_output, Y_pred_rf)
print("Random forest accuracy with 1000 base learners: ", acc_rf)
print(confusion_matrix(test_spam_output, Y_pred_rf))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_features='auto'`
  warn(
Random forest accuracy with 1000 base learners:  0.9336295473479589
[[2096   86]
 [ 153 1266]]
```

As seen above for Random Forest with 1000 base learners, the classification accuracy is **93.36%**.
Here we can conclude that **Fusion Classifier with majority voting** rule is more accurate.

Training-Test size using: 50%- 50%

```
#Adaboost classifier with Decision Tree as base learner with different training and testing sizes
```

```
[76] #Training & Testing splits: 50%-50%
     from sklearn.model_selection import train_test_split
     X_train, X_test, Y_train, Y_test = train_test_split(spam_input,spam_output, test_size = 0.5, random_s
     #Initialize Decision Tree Classifier and fit into training data
     clf_dt = DecisionTreeClassifier()
     clf_dt.fit(X_train, Y_train)
     #Predict
     Y_pred = clf_dt.predict(X_test)
     #Accuracy of the classifier
     acc_dt= accuracy_score(Y_test, Y_pred)
     print("Accuracy:", acc_dt)
     #Confusion_Matrix
     print(confusion_matrix(Y_test, Y_pred))
```

```
Accuracy: 0.8974358974358975
[[1299  122]
 [ 114  766]]
```

Accuracy given above is **89.74%**.

Fusion Classifier with Majority voting rule:

```
[81] # Fuse the classifier with Decision Tree, Gaussian Naive Bayes and Logistic Regression by using majority voting rule
     clf_dt_model = DecisionTreeClassifier()
     clf_dt_score = clf_dt_model.fit(X_train,Y_train)
     clf_gnb_model = GaussianNB()
     clf_gnb_score = clf_gnb_model.fit(X_train,Y_train)
     clf_lr_model = LogisticRegression()
     clf_lr_score = clf_lr_model.fit(X_train,Y_train)
     Fusion_model = VotingClassifier(estimators=[('DT', clf_dt_model), ('GNB', clf_gnb_model), ('LR', clf_lr_model)], voting='har
     Fusion_score = Fusion_model.fit(X_train,Y_train)
```

```
Y_pred = Fusion_score.predict(X_test)
accuracy_score(Y_test, Y_pred)
print("Accuracy:",accuracy_score(Y_test, Y_pred))
target_names = ['class 0', 'class 1']
print(classification_report(Y_test, Y_pred, target_names=target_names, digits=5))
print(confusion_matrix(Y_test, Y_pred))
```

```
Accuracy: 0.9282920469361148
              precision    recall  f1-score   support

     class 0    0.96450   0.91766   0.94050      1421
     class 1    0.87671   0.94545   0.90979       880

    accuracy                        0.92829      2301
   macro avg    0.92060   0.93156   0.92514      2301
weighted avg    0.93092   0.92829   0.92875      2301

[[1304  117]
 [  48  832]]
```

Accuracy given above is **92.83%.**

Here, we can conclude that **Training-Test size using: 50%- 50%** is less accurate with Adaboost classifier with decision tree (**89.74%**) than Fusion classifier with decision tree as the base estimator (**92.83%**) .

Training-Test size using: 60%- 40%

```
[72] #Training & Testing splits: 60%-40%
     from sklearn.model_selection import train_test_split
     X_train, X_test, Y_train, Y_test = train_test_split(spam_input,spam_output, test_size = 0.4, random_state = 0)
     #Initialize Decision Tree Classifier and fit into training data
     clf_dt = DecisionTreeClassifier()
     clf_dt.fit(X_train, Y_train)
     #Predict
     Y_pred = clf_dt.predict(X_test)
     #Accuracy of the classifier
     acc_dt= accuracy_score(Y_test, Y_pred)
     print("Accuracy:", acc_dt)
     #Confusion_Matrix
     print(confusion_matrix(Y_test, Y_pred))

     Accuracy: 0.9071156979902227
     [[1055   88]
      [  83  615]]
```

Accuracy given above is **90.71%.**

Fusion Classifier with Majority voting rule:

```
[81] # Fuse the classifier with Decision Tree, Gaussian Naive Bayes and Logistic Regression by using majority voting rule
     clf_dt_model = DecisionTreeClassifier()
     clf_dt_score = clf_dt_model.fit(X_train,Y_train)
     clf_gnb_model = GaussianNB()
     clf_gnb_score = clf_gnb_model.fit(X_train,Y_train)
     clf_lr_model = LogisticRegression()
     clf_lr_score = clf_lr_model.fit(X_train,Y_train)
     Fusion_model = VotingClassifier(estimators=[('DT', clf_dt_model), ('GNB', clf_gnb_model), ('LR', clf_lr_model)], voting='har
     Fusion_score = Fusion_model.fit(X_train,Y_train)
```

```
Y_pred = Fusion_score.predict(X_test)
accuracy_score(Y_test, Y_pred)
print("Accuracy:",accuracy_score(Y_test, Y_pred))
target_names = ['class 0', 'class 1']
print(classification_report(Y_test, Y_pred, target_names=target_names, digits=5))
print(confusion_matrix(Y_test, Y_pred))
```

```
Accuracy: 0.9282920469361148
              precision    recall  f1-score   support

     class 0    0.96176   0.92048   0.94067      1421
     class 1    0.87991   0.94091   0.90939       880

    accuracy                        0.92829      2301
   macro avg    0.92084   0.93069   0.92503      2301
weighted avg    0.93046   0.92829   0.92871      2301

[[1308  113]
 [  52  828]]
```

Accuracy given above is **92.83%**.

Here, we can conclude that **Training-Test size using: 60%- 40%** is less accurate with Adaboost classifier with decision tree (**90.71%**) than Fusion classifier with decision tree as the base estimator (**92.83%**) .

Training-Test size using: 70%- 30%

```
#Training & Testing splits: 70%-30%
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(spam_input,spam_output, test_size = 0.3, random_state = 0)
#Initialize Decision Tree Classifier and fit into training data
clf_dt = DecisionTreeClassifier()
clf_dt.fit(X_train, Y_train)
#Predict
Y_pred = clf_dt.predict(X_test)
#Accuracy of the classifier
acc_dt= accuracy_score(Y_test, Y_pred)
print("Accuracy:", acc_dt)
#Confusion_Matrix
print(confusion_matrix(Y_test, Y_pred))
```

```
Accuracy: 0.9044170890658942
[[786  65]
 [ 67 463]]
```

Accuracy given above is **90.44%**.

Fusion Classifier with Majority voting rule:

```
[81] # Fuse the classifier with Decision Tree, Gaussian Naive Bayes and Logistic Regression by using majority voting rule
     clf_dt_model = DecisionTreeClassifier()
     clf_dt_score = clf_dt_model.fit(X_train,Y_train)
     clf_gnb_model = GaussianNB()
     clf_gnb_score = clf_gnb_model.fit(X_train,Y_train)
     clf_lr_model = LogisticRegression()
     clf_lr_score = clf_lr_model.fit(X_train,Y_train)
     Fusion_model = VotingClassifier(estimators=[('DT', clf_dt_model), ('GNB', clf_gnb_model), ('LR', clf_lr_model)], voting='har
     Fusion_score = Fusion_model.fit(X_train,Y_train)
```

```
Y_pred = Fusion_score.predict(X_test)
accuracy_score(Y_test, Y_pred)
print("Accuracy:",accuracy_score(Y_test, Y_pred))
target_names = ['class 0', 'class 1']
print(classification_report(Y_test, Y_pred, target_names=target_names, digits=5))
print(confusion_matrix(Y_test, Y_pred))
```

```
Accuracy: 0.9265536723163842
              precision    recall  f1-score   support

     class 0    0.96165   0.91766   0.93914      1421
     class 1    0.87619   0.94091   0.90740       880

    accuracy                        0.92655      2301
   macro avg    0.91892   0.92929   0.92327      2301
weighted avg    0.92897   0.92655   0.92700      2301

[[1304  117]
 [  52  828]]
```

Accuracy given above is **92.66%**.

Here, we can conclude that **Training-Test size using: 70%- 30%** is less accurate with Adaboost classifier with decision tree (**90.44%**) than Fusion classifier with decision tree as the base estimator (**92.66%**) .

Training-Test size using: 80%- 20%

```
#Training & Testing splits: 80%-20%
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(spam_input,spam_output, test_size = 0.4, random_state = 0)
#Initialize Decision Tree Classifier and fit into training data
clf_dt = DecisionTreeClassifier()
clf_dt.fit(X_train, Y_train)
#Predict
Y_pred = clf_dt.predict(X_test)
#Accuracy of the classifier
acc_dt= accuracy_score(Y_test, Y_pred)
print("Accuracy:", acc_dt)
#Confusion_Matrix
print(confusion_matrix(Y_test, Y_pred))
```

```
Accuracy: 0.9043997827267789
[[1050   93]
 [  83  615]]
```

Accuracy given above is **90.44%**.

Fusion Classifier with Majority voting rule:

```
[81] # Fuse the classifier with Decision Tree, Gaussian Naive Bayes and Logistic Regression by using majority voting rule
     clf_dt_model = DecisionTreeClassifier()
     clf_dt_score = clf_dt_model.fit(X_train,Y_train)
     clf_gnb_model = GaussianNB()
     clf_gnb_score = clf_gnb_model.fit(X_train,Y_train)
     clf_lr_model = LogisticRegression()
     clf_lr_score = clf_lr_model.fit(X_train,Y_train)
     Fusion_model = VotingClassifier(estimators=[('DT', clf_dt_model), ('GNB', clf_gnb_model), ('LR', clf_lr_model)], voting='har
     Fusion_score = Fusion_model.fit(X_train,Y_train)
```

```
[90] Y_pred = Fusion_score.predict(X_test)
     accuracy_score(Y_test, Y_pred)
     print("Accuracy:",accuracy_score(Y_test, Y_pred))
     target_names = ['class 0', 'class 1']
     print(classification_report(Y_test, Y_pred, target_names=target_names, digits=5))
     print(confusion_matrix(Y_test, Y_pred))
```

```
Accuracy: 0.9261190786614515
              precision    recall  f1-score   support

     class 0    0.96094   0.91766   0.93880      1421
     class 1    0.87606   0.93977   0.90680       880

    accuracy                        0.92612      2301
   macro avg    0.91850   0.92872   0.92280      2301
weighted avg    0.92848   0.92612   0.92656      2301

[[1304  117]
 [  53  827]]
```

Accuracy given above is 92.61%

Here, we can conclude that **Training-Test size using: 80%- 20%** is less accurate with Adaboost classifier with decision tree (**90.44%**) than Fusion classifier with decision tree as the base estimator (**92.61%**) .

Observations:

- The classification accuracy of the **Decision Tree Classifier** is **87.64%**.
- The classification accuracy of the **Gaussian Naïve Bayes Classifier** is **83.00%**.
- The classification accuracy of the **Logistic Regression** is **90.50%**.

  Comparing the three classifications, the **Logistic Regression** is the most accurate.

- The classification accuracy of the **Fusion Classifier with majority voting is 93.89%.**
- The classification accuracy of the **Adaboost classifier with Decision Tree** as base learner is **87.95%**.
- **Random Forest** with 1000 base learners, the classification accuracy is **93.36%**.

In this case **Fusion Classifier with majority voting** is the most accurate**.**

- **Training-Test size using: 50%- 50%** is less accurate with Adaboost classifier with decision tree (**89.74%**) than Fusion classifier with decision tree as the base estimator (**92.83%**).

- **Training-Test size using: 60%- 40%** is less accurate with Adaboost classifier with decision tree (**90.71%**) than Fusion classifier with decision tree as the base estimator (**92.83%**) .

- **Training-Test size using: 70%- 30%** is less accurate with Adaboost classifier with decision tree (**90.44%**) than Fusion classifier with decision tree as the base estimator (**92.66%**) .

- **Training-Test size using: 80%- 20%** is less accurate with Adaboost classifier with decision tree (**90.44%**) than Fusion classifier with decision tree as the base estimator (**92.61%**) .