PMDS505P Data Mining and Machine Learning
Experiment 7

February 2025

1 Work to do today

Note: Make a single pdf file of the work you are doing in jupyter notebook. Upload with proper format. Please mention your name and roll no properly with Experiment number in the first page of your submission.

Decision Tree: Bagging, Boosting, RandomForestClassifier

Q1. Today we will try to see how bagging, boosting etc can be implemented.

- Download the liver patient dataset that you have with you.
- Perform Min-Max scaling
- Do the train test split of the data with test size 20%.
- Fit the LogisticRegression model to the training data.
- Print the testing accuracy.
- Next for the same data fit a Decision Tree with the same training data and check for the testing accuracy.
- Note down your accuracy scores obtained.

Bagging: Bagging Classfier and Bagging regressor.

- Let's try to fit the ensemble techniques BaggingClassifier and a RandomForestClassifier and check for the test set accuracies. Keep random_state as a fixed number. otherwise, the accuracies may vary.

- Import the two classifiers from sklearn.ensemble and fit them using the training data we had.

```
from sklearn.ensemble import BaggingClassifier, RandomForestClassifier
BC = BaggingClassifier(n_estimators = 100,random_state = 0)
BC.fit(X_train,y_train)
print(BC.predict(X_test))
print(accuracy_score(BC.predict(X_test),y_test))
```

# here n_estimators are the number of trees we are fitting as a part of bagging. Here n_estimators is a hyperparameter also.

```
# To implement pasting you can do this. we make the parameter bootstrap to False.
pBC = BaggingClassifier(n_estimators = 100,random_state = 0, bootstrap = False)
pBC.fit(X_train,y_train)
print(pBC.predict(X_test))
print(accuracy_score(pBC.predict(X_test),y_test))
```

- To implement RandomForestClassifier you can perform the following.

```
RFC = RandomForestClassifier(n_estimators = 100,max_features = "sqrt", random_state = 0)
RFC.fit(X_train,y_train)
print(RFC.predict(X_test))
print(accuracy_score(RFC.predict(X_test),y_test))
```

- If you want to see the most relevant features in your modeling you can use the following code to print the scores. when you have a large score it means those features were crucial in the modeling.

```
print(RFC.feature_importances_)
```

Q2. Form a synthetic dataset using the make_classification class which we have used in the previous labs with two features and 3 classes. Fit the decisiontreeclassfier,baggingclassifier and randomforestclassifier and print the decsion boundaries for the different classifiers. Visualizing in the previous case is not possible so we can see how the nonlinear boundaries are getting created in these models except for a linear boundary that we have seen by other classifiers in the previous labs.

VotingClassifier

- Next we we will see how a voting classifier can be fitted using different models using Hard voting and soft voting. Here we are building a voting classifier with three models logistic regression, random forest classifier and softmax regression.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.linear_model import LogisticRegression

log_clf = LogisticRegression()
rnd_clf = RandomForestClassifier(random_state = 0)
sm_clf = model = LogisticRegression(multi_class='multinomial')
voting_clf = VotingClassifier( estimators=[('lr', log_clf), ('rf', rnd_clf), ('sm', sm_clf)],
voting='hard')
voting_clf.fit(X_train, y_train)
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test, voting_clf.predict(X_test)))
```

- Change to voting criteria to soft by setting voting = 'soft' and check the output in the above case. In the case of soft voting the models output a probability that is averaged and used for prediction

- You can see the accuracy of this voting classifier. when we are using ensemble learning the accuracy is increasing for the model.
- Next we will see how boosting can be implemented especially AdaboostClassifier.

  # **Here i have fixed the different trees we are fitting inside to be decision trees of depth 1 or height 1, which are called decision stumps**.

```
from sklearn.ensemble import AdaBoostClassifier
base_model = DecisionTreeClassifier(max_depth=1)
ABC = AdaBoostClassifier(base_estimator=base_model,n_estimators=500, random_state=0)
ABC.fit(X_train, y_train)
pred_ABC = ABC.predict(X_test)
print("AdaBoost Accuracy:", accuracy_score(y_test, pred_ABC))
```

- Next perform hyperparameter tuning using GridSearchCV for the random forest classifier with the parameters as max_depth, n_estimators, min_samples_leaf with the values given below. You can also try to figure out the maximum accuracy that you can derive using a different parameter grid. Find the best combination of the hyperparameters.

  n_estimators: [10, 50, 100, 200, 300, 400, 500, 700]
  max_depth: [3,5,7]
  min_samples_leaf: [1, 2, 4]

  n_estimators → Number of trees in the forest.
  max_depth → Maximum depth of each tree.
  min_samples_leaf → Minimum samples required in leaf nodes.

- Again try to tune the parameters in Adaboostclassifier so that we can get maximum accuracy with the following hyperparameters.

n_estimators: [10,50, 75,100, 125, 150, 200, 400, 500]

base_estimator__max_depth: [1, 2, 3]

- Now you can try to fit the multiple regression model, DecisionTreeRegressor, BaggingRegressor, RandomForestRegressor and AdaboostRegressor on the Book1.csv file and use the mean squared error to see how these ensemble models perform compared to the basic models.