Untitled2

October 22, 2017

In [1]:
```
# Loading required libraries
```

In [2]:
```
from sklearn import metrics, svm, datasets, tree
from sklearn.svm import SVR
from sklearn.ensemble import GradientBoostingClassifier, BaggingClassifier
from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, classification_report
from sklearn.model_selection import train_test_split
```

In [3]:
```
# QUESTION 1: Use SVR (SVM for regression) for Boston dataset as following:
#Fit a regression three different models with kernels rbf,poly and linear
#C=100, GAMMA=0.1 AND DEGREE=2.
#Which model has lower error?
```

In [4]:
```
# Loading Boston dataset
```

In [5]:
```
from sklearn.datasets import load_boston
X,Y = load_boston(return_X_y=True)
```

In [6]:
```
# Splitting dataset into train and test in 70:30 ratio respectively
```

In [7]:
```
X_train_label, X_test_label, Y_train_label,
Y_test_label = train_test_split(X, Y, test_size=0.30,
                                                    random_state=1234)
```

In [8]:
```
# Common Parameters for SVM model
```

In [9]:
```
C = 100.0
gamma = 0.1
degree = 2
```

In [10]:
```
# KERNEL = rbf
```

In [11]:
```
svr_reg_rbf = svm.SVR(kernel='rbf', C=C, gamma=gamma, degree=degree)
svr_reg_rbf = svr_reg_rbf.fit(X_train_label, Y_train_label)
Y_pred_svr_reg_rbf = svr_reg_rbf.predict(X_test_label)
```

In [12]:
```
# Loss with rbf kernel
```

```
In [13]: metrics.mean_squared_error(Y_test_label, Y_pred_svr_reg_rbf)

Out[13]: 82.583167317229325

In [14]: # KERNEL = poly

In [15]: svr_reg_poly = svm.SVR(kernel='poly', C=C, gamma=gamma, degree=degree)
         svr_reg_poly = svr_reg_poly.fit(X_train_label, Y_train_label)
         Y_pred_svr_reg_poly = svr_reg_poly.predict(X_test_label)

In [16]: # Loss with poly kernel

In [17]: metrics.mean_squared_error(Y_test_label, Y_pred_svr_reg_poly)

Out[17]: 1016892792.815606

In [18]: # KERNEL = linear

In [19]: svr_reg_linear = svm.SVR(kernel='linear', C=C, gamma=gamma, degree=degree)
         svr_reg_linear = svr_reg_linear.fit(X_train_label, Y_train_label)
         Y_pred_svr_reg_linear = svr_reg_linear.predict(X_test_label)

In [20]: # Loss with linear kernel

In [21]: metrics.mean_squared_error(Y_test_label, Y_pred_svr_reg_linear)

Out[21]: 41.707352129515357

In [22]: # ANSWER: The SVM model using the 'linear'Kernel has the lowest error

In [23]: # QUESTION 2: Use Breast cancer dataset to fit a gradient boosting model
         # Use Gridsearch to find the best
         # n_estimators=[10,100,200,500] and max_depth=[2,3,5,7].
         #What are the best parameters?
         #Use classification_report to report the accuracy of classification.

In [24]: # Loading Breast Cancer dataset

In [25]: from sklearn.datasets import load_breast_cancer
         X,Y = load_breast_cancer(return_X_y=True)

In [26]: # Splitting dataset into train and test in 70:30 ratio respectively

In [27]: X_train_label, X_test_label, Y_train_label,
         Y_test_label = train_test_split(X, Y, test_size=0.30,
                                                          random_state=1234)

In [28]: # parameter list for GradientBoostingClassifier

In [29]: params_GradientBoosted = {'n_estimators':[10,100,200,500],
                                   'max_depth': [2,3,5,7]}
```

```
In [30]: # Fiitting the Gradient Boosted Model on training dataset

In [31]: optimized_GradientBoosted = GridSearchCV(GradientBoostingClassifier(),
                                                  params_GradientBoosted)
         model_GradientBossted = optimized_GradientBoosted.fit(X_train_label, Y_train_label)

In [32]: # Best Parameters for Gradient Boosted Model
         print(model_GradientBossted.best_estimator_)

GradientBoostingClassifier(criterion='friedman_mse', init=None,
              learning_rate=0.1, loss='deviance', max_depth=2,
              max_features=None, max_leaf_nodes=None,
              min_impurity_split=1e-07, min_samples_leaf=1,
              min_samples_split=2, min_weight_fraction_leaf=0.0,
              n_estimators=500, presort='auto', random_state=None,
              subsample=1.0, verbose=0, warm_start=False)


In [33]: # Accuracies for Gradient Boosted Model
         Outcome_GradientBoosted=model_GradientBossted.predict(X_test_label)
         print (classification_report(Outcome_GradientBoosted,Y_test_label))

             precision    recall  f1-score   support

          0       0.82      1.00      0.90        54
          1       1.00      0.90      0.95       117

avg / total       0.94      0.93      0.93       171


In [34]: #QUESTION 3: Using Breast cancer dataset apply bagging method
         #with decision tree for the following:
         # Use Gridsearch to find the best parameters for
         # n_estimator=[10,100,200,500], max_depth=[2,3,5,7].
         #What are the best parameters?
         #Use classification_report to report the accuracy.

In [35]: # parameter list for BaggingClassifier

In [36]: params_Bagging = {'n_estimators':[10,100,200,500]}

In [37]: # Fiitting the Bagging model on training dataset

In [38]: optimized_Bagging = GridSearchCV(BaggingClassifier(tree.DecisionTreeClassifier()),
                                          params_Bagging)
         model_Bagging = optimized_Bagging.fit(X_train_label, Y_train_label)

In [39]: # Best Parameters for Bagging
         print(model_Bagging.best_estimator_)
```

```
BaggingClassifier(base_estimator=DecisionTreeClassifier(class_weight=None, criterion='gini', ma
            max_features=None, max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            presort=False, random_state=None, splitter='best'),
          bootstrap=True, bootstrap_features=False, max_features=1.0,
          max_samples=1.0, n_estimators=500, n_jobs=1, oob_score=False,
          random_state=None, verbose=0, warm_start=False)
```

In [40]:
```
# Accuracies for Bagging
Outcome_Bagging=model_Bagging.predict(X_test_label)
print (classification_report(Outcome_Bagging,Y_test_label))
```

```
             precision    recall  f1-score   support

          0       0.83      0.96      0.89        57
          1       0.98      0.90      0.94       114

avg / total       0.93      0.92      0.93       171
```

In [41]:
```
# QUESTION 4: Using Breast cancer dataset apply random forest
# for the following:
# Use Gridsearch to find the best parameters for
# n_estimator=[10,100,200,500], max_depth=[2,3,5,7].
# What are the best parameters?
# Use classification_report to report the accuracy.
```

In [42]:
```
# parameter list for RandomForestClassifier
```

In [43]:
```
params_RandomForest = {'n_estimators':[10,100,200,500],'max_depth': [2,3,5,7]}
```

In [44]:
```
optimized_RandomForest = GridSearchCV(RandomForestClassifier(), params_RandomForest)
model_RandomForest = optimized_RandomForest.fit(X_train_label, Y_train_label)
```

In [45]:
```
# Best Parameters for RandmForest
print(model_RandomForest .best_estimator_)
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=7, max_features='auto', max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            n_estimators=100, n_jobs=1, oob_score=False, random_state=None,
            verbose=0, warm_start=False)
```

In [46]:
```
# Accuracies for RandomForest
Outcome_RandomForest=model_RandomForest .predict(X_test_label)
print (classification_report(Outcome_RandomForest,Y_test_label))
```

	precision	recall	f1-score	support
0	0.83	0.98	0.90	56
1	0.99	0.90	0.95	115
avg / total	0.94	0.93	0.93	171