Analysis of Classifier Performance for the Prediction of Medical Conditions Compared to a Small Dimensional Data Set

Miguel Garcia
University of California, San Diego
mig053@ucsd.edu

Abstract

Here in this paper, I am going to apply three classifiers to three different data sets. The classifiers that I will apply are the following: K-nearest neighbors, decision trees with a specified depth list, and random forests with a specified depth list. I am going to evaluate which classifiers out of these three are the most reliable when it comes to scientific and medical data and see if there seems to be a drastic difference when the dimensionality and size of these data sets. K-nearest neighbors resulted in the most fluctuations across these data sets, giving inconsistent results hence not being a reliable classifier to use for these data sets.

1 Introduction

There are many arising problems in the 21st century typically addressed in a pragmatic way. Something that has been also arising is the recording of important data that we can take advantage of in order to address a specific problem. There are many different types of data sets, political, qualitative data sets, quantitative data sets, etc. For this paper I use quantitative data sets, first for iris of plants, then for breast cancer and mammographic masses data sets. I will train the three classifiers with the data sets split into training and testing. Then I can compare the accuracies and hyper-parameters given by each trained classifier. After, observations can be made to determine whether or not these classifiers can be helpful when it comes to predicting important medical conditions such as cancer.

2 Method

2.1 Data Sets

In this section, you can see the summary of the classifiers and parameters used for each data set. You can also see how these data sets were cleaned of null data and then split to see the results given by the classifiers. The data sets used can be found at the UCI Machine Learning Repository [1] and all the details about each of them and which features they these have.

The data sets were extracted using pandas data frames then convert them to list so that the classifier algorithms can handle them. The iris data set [2] contains 160 data points of 4 features each. The breast cancer data set [3] contains 568 data points with 31 features each and the mammographic masses data set [4] has 829 data points with 4 features each. Information for the data sets attributes information can be found in the reference links. I will apply three different classifiers to these data sets to evaluate which classifiers are the most reliable to use when it comes to these types of data sets.

The cleaning that had to be made is done to the breast cancer data set where the ‘y’ labels were not given in integers of the form ‘1.0’ and ‘0.0’. Similarly, cleaning had to be done to the mammographic masses data set to remove all rows that have null values.

Before applying the algorithms to each data set, they were shuffled then divided into training and test samples for ‘x’ and ‘y’. The split is done as 80% of the data points for training and 20% for testing.

2.2 Other Methods and Implementations Used

A helper function that converts labels from strings to numeric values is implemented for the purpose of converting the ‘y’ values in the iris dataset and similarly for the breast cancer data set.

A cross-validation function [5] is implemented adapted for the use of the k-nearest neighbor classifier algorithm. Grid search [6] is also implemented adapted for the use of the k-nearest neighbor classifier algorithm. I also used an scikit-learn implementation of grid search [7]. Heat-map functions [8] were also used in order to show the parameters and accuracies.

2.3 Classifiers Used

K-nearest neighbors classifier algorithm and its implementation as given by scikit-learn [9] with a 'k-list' that includes [1,2,3,4,5,6] is used in order to find the hyper-parameters and to determine the validation and training accuracy for each value of the list, results are then reported using a heat map resulting from the six-fold cross-validation. Then after that testing accuracy is calculated.

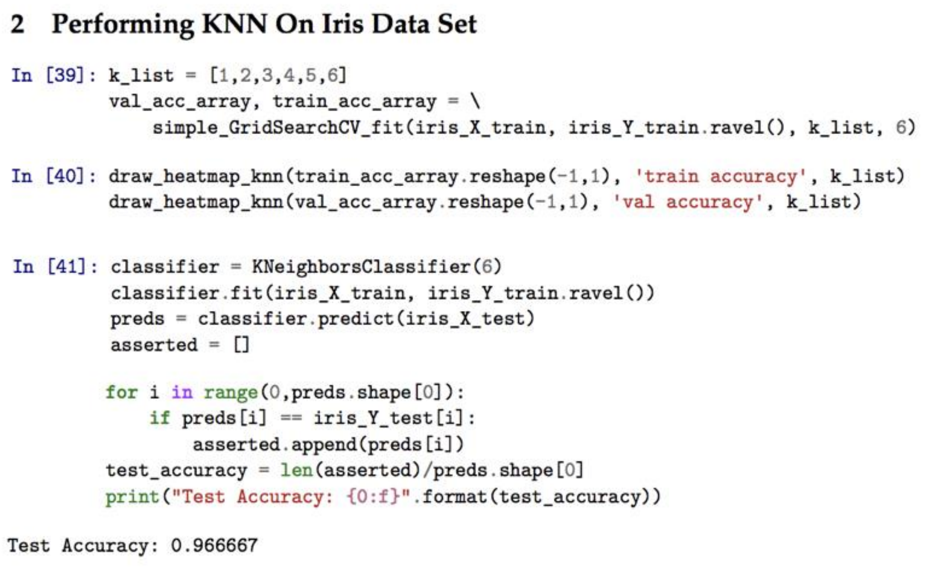
Decision trees classifier algorithm and its implementation as given by scikit-learn [10] with a 'depth list' that includes [1,2,3,4,5] using an entropy criterion. After that grid search with ten folds cross-validation (five for mammographic data set) is performed and the hyper-parameters are reported with a heat map. Then testing accuracy is calculated using the test data sets that were initially split.

Random forest classifier algorithm and its implementation as given by scikit-learn [11] with a 'depth list' that includes [1,2,3,4,5] and has a maximum depth of five. Then grid search with ten-fold cross-validation is performed and hyper-parameters are reported with a heat map. Then testing accuracy is calculated using the test data sets that were initially split.

3 Experiment

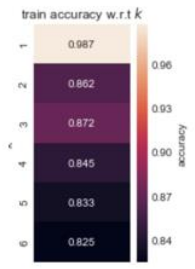
In this section, I will run several experiments on the data sets with the given classifiers. For the first part, I will apply k-nearest neighbors on each data set then determine the hyper-parameters using grid search with cross-validation. Look at the attached file for more details on the implementation and details of these experiments

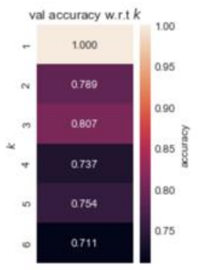
3.1 K-nearest neighbors Classifier

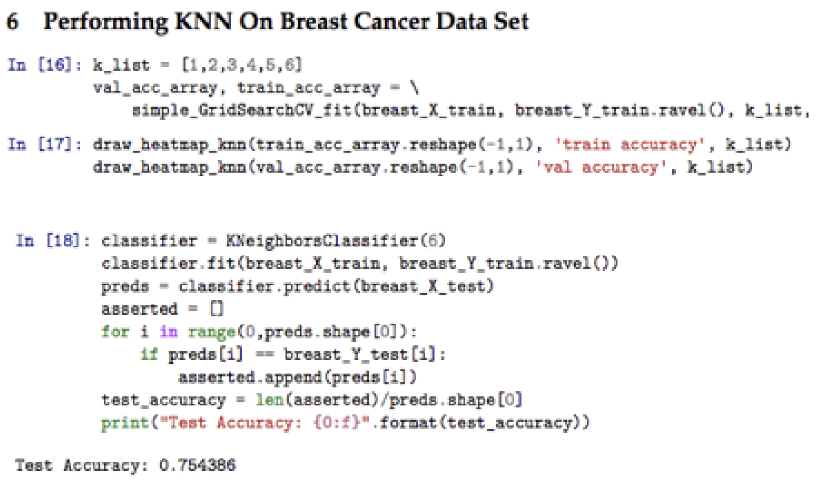


2 Performing KNN On Iris Data Set

```
In [39]: k_list = [1,2,3,4,5,6]
         val_acc_array, train_acc_array = \
             simple_GridSearchCV_fit(iris_X_train, iris_Y_train.ravel(), k_list, 6)

In [40]: draw_heatmap_knn(train_acc_array.reshape(-1,1), 'train accuracy', k_list)
         draw_heatmap_knn(val_acc_array.reshape(-1,1), 'val accuracy', k_list)


In [41]: classifier = KNeighborsClassifier(6)
         classifier.fit(iris_X_train, iris_Y_train.ravel())
         preds = classifier.predict(iris_X_test)
         asserted = []

        for i in range(0,preds.shape[0]):
            if preds[i] == iris_Y_test[i]:
                asserted.append(preds[i])
        test_accuracy = len(asserted)/preds.shape[0]
        print("Test Accuracy: {0:f}".format(test_accuracy))

Test Accuracy: 0.966667
```

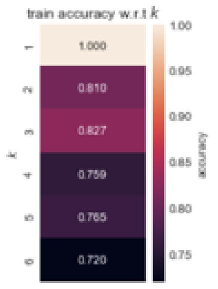


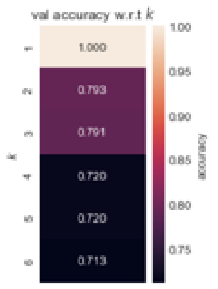


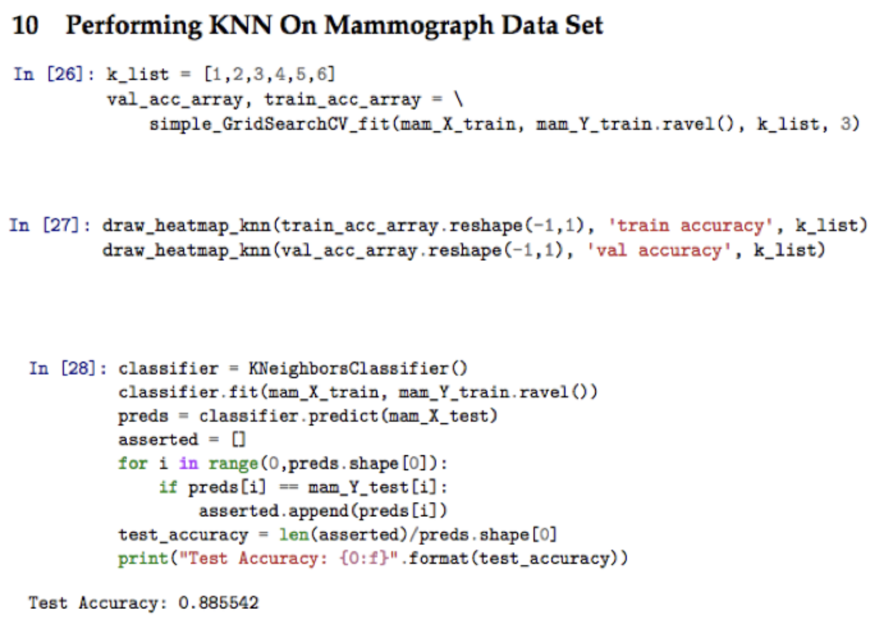


6 Performing KNN On Breast Cancer Data Set

```
In [16]: k_list = [1,2,3,4,5,6]
         val_acc_array, train_acc_array = \
             simple_GridSearchCV_fit(breast_X_train, breast_Y_train.ravel(), k_list,

In [17]: draw_heatmap_knn(train_acc_array.reshape(-1,1), 'train accuracy', k_list)
         draw_heatmap_knn(val_acc_array.reshape(-1,1), 'val accuracy', k_list)


 In [18]: classifier = KNeighborsClassifier(6)
          classifier.fit(breast_X_train, breast_Y_train.ravel())
          preds = classifier.predict(breast_X_test)
          asserted = []
          for i in range(0,preds.shape[0]):
              if preds[i] == breast_Y_test[i]:
                  asserted.append(preds[i])
          test_accuracy = len(asserted)/preds.shape[0]
          print("Test Accuracy: {0:f}".format(test_accuracy))

 Test Accuracy: 0.754386
```





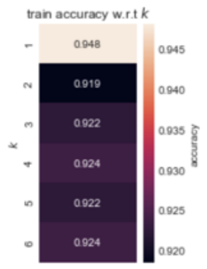


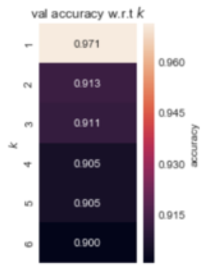
10 Performing KNN On Mammograph Data Set

```
In [26]: k_list = [1,2,3,4,5,6]
         val_acc_array, train_acc_array = \
             simple_GridSearchCV_fit(mam_X_train, mam_Y_train.ravel(), k_list, 3)
```

```
In [27]: draw_heatmap_knn(train_acc_array.reshape(-1,1), 'train accuracy', k_list)
         draw_heatmap_knn(val_acc_array.reshape(-1,1), 'val accuracy', k_list)
```

```
In [28]: classifier = KNeighborsClassifier()
         classifier.fit(mam_X_train, mam_Y_train.ravel())
         preds = classifier.predict(mam_X_test)
         asserted = []
         for i in range(0,preds.shape[0]):
             if preds[i] == mam_Y_test[i]:
                 asserted.append(preds[i])
         test_accuracy = len(asserted)/preds.shape[0]
         print("Test Accuracy: {0:f}".format(test_accuracy))

Test Accuracy: 0.885542
```

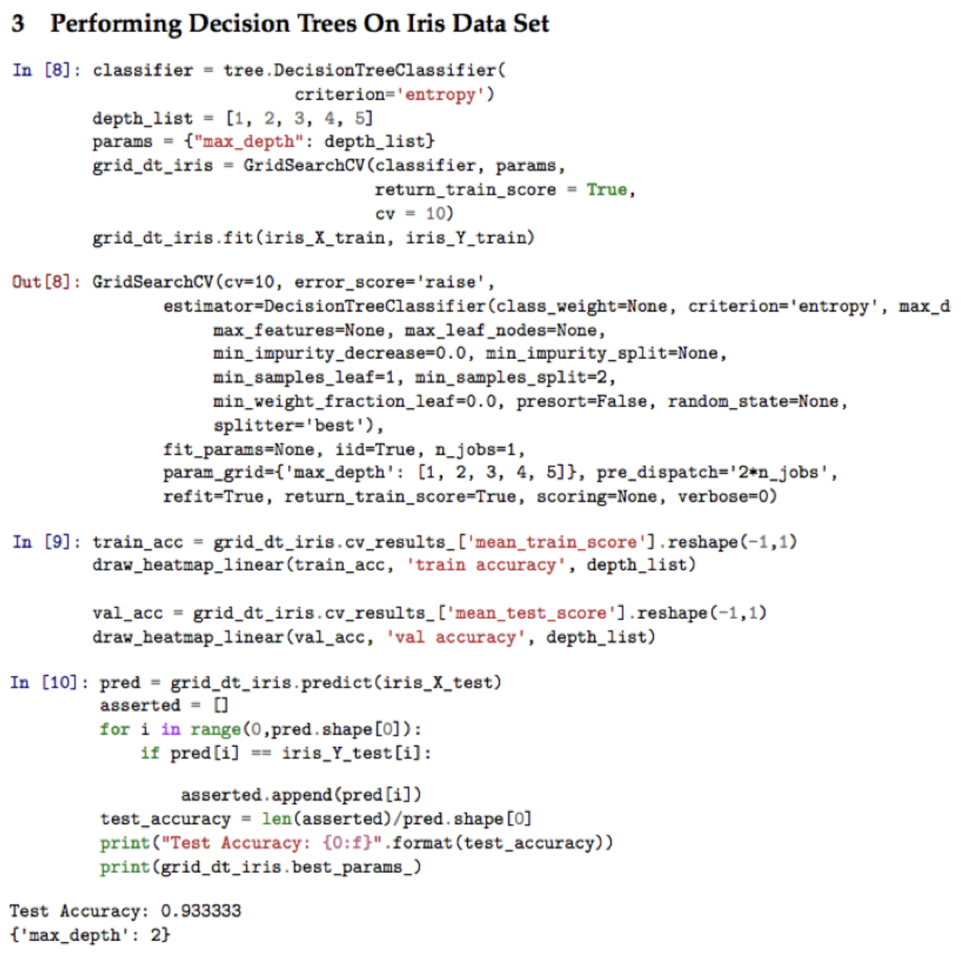




3.1.1 K-nearest neighbors Classifier Performance Analysis

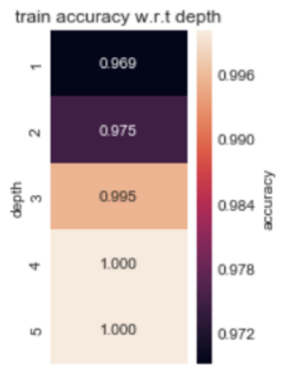
In here you can see how k-nearest neighbors' test accuracy fluctuates from data set to data set, the pattern it follows is that the less features a data set has, and the fewer data points a data set has, the more accurate will k-nearest neighbors be. Now, another observation that can be made is that this pattern also follows for validation and training accuracy. Even then, the test accuracy is not high enough as expected. For the breast cancer data set, here you can observe that it has a really low-test accuracy (~.75) that could result from the fact that it is highly dimensional and has too many features as compared to the iris and mammographic masses data set.

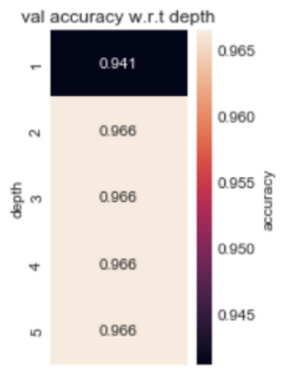
3.2 Decision Trees Classifier



3 Performing Decision Trees On Iris Data Set

```
In [8]: classifier = tree.DecisionTreeClassifier(
                            criterion='entropy')
        depth_list = [1, 2, 3, 4, 5]
        params = {"max_depth": depth_list}
        grid_dt_iris = GridSearchCV(classifier, params,
                                   return_train_score = True,
                                   cv = 10)
        grid_dt_iris.fit(iris_X_train, iris_Y_train)

Out[8]: GridSearchCV(cv=10, error_score='raise',
               estimator=DecisionTreeClassifier(class_weight=None, criterion='entropy', max_d
                   max_features=None, max_leaf_nodes=None,
                   min_impurity_decrease=0.0, min_impurity_split=None,
                   min_samples_leaf=1, min_samples_split=2,
                   min_weight_fraction_leaf=0.0, presort=False, random_state=None,
                   splitter='best'),
               fit_params=None, iid=True, n_jobs=1,
               param_grid={'max_depth': [1, 2, 3, 4, 5]}, pre_dispatch='2*n_jobs',
               refit=True, return_train_score=True, scoring=None, verbose=0)

In [9]: train_acc = grid_dt_iris.cv_results_['mean_train_score'].reshape(-1,1)
        draw_heatmap_linear(train_acc, 'train accuracy', depth_list)

        val_acc = grid_dt_iris.cv_results_['mean_test_score'].reshape(-1,1)
        draw_heatmap_linear(val_acc, 'val accuracy', depth_list)

In [10]: pred = grid_dt_iris.predict(iris_X_test)
         asserted = []
         for i in range(0,pred.shape[0]):
             if pred[i] == iris_Y_test[i]:

                 asserted.append(pred[i])
         test_accuracy = len(asserted)/pred.shape[0]
         print("Test Accuracy: {0:f}".format(test_accuracy))
         print(grid_dt_iris.best_params_)

Test Accuracy: 0.933333
{'max_depth': 2}
```



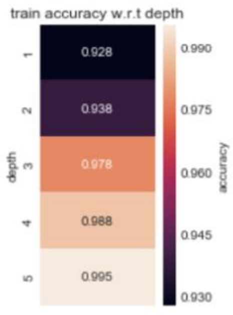


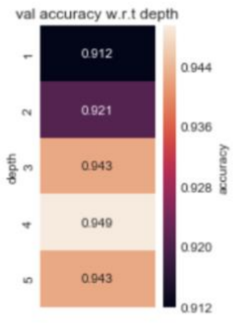
7 Performing Decision Trees On Breast Cancer Data Set

```
In [19]: classifier = tree.DecisionTreeClassifier(criterion='entropy')
         depth_list = [1, 2, 3, 4, 5]
         params = {"max_depth": depth_list}
         grid_dt_breast = GridSearchCV(classifier, params, return_train_score = True, cv = 10)
         grid_dt_breast.fit(breast_X_train, breast_Y_train)
```

```
Out[19]: GridSearchCV(cv=10, error_score='raise',
                estimator=DecisionTreeClassifier(class_weight=None, criterion='entropy', max_d
                     max_features=None, max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, presort=False, random_state=None,
                     splitter='best'),
                fit_params=None, iid=True, n_jobs=1,
                param_grid={'max_depth': [1, 2, 3, 4, 5]}, pre_dispatch='2*n_jobs',
                refit=True, return_train_score=True, scoring=None, verbose=0)
```

```
In [20]: train_acc = grid_dt_breast.cv_results_['mean_train_score'].reshape(-1,1)
         draw_heatmap_linear(train_acc, 'train accuracy', depth_list)

         val_acc = grid_dt_breast.cv_results_['mean_test_score'].reshape(-1,1)
         draw_heatmap_linear(val_acc, 'val accuracy', depth_list)
```





```
In [21]: pred = grid_dt_breast.predict(breast_X_test)
         asserted = []
         for i in range(0,pred.shape[0]):
             if pred[i] == breast_Y_test[i]:

                 asserted.append(pred[i])
        test_accuracy = len(asserted)/pred.shape[0]
        print(test_accuracy)
        print(grid_dt_breast.best_params_)
```

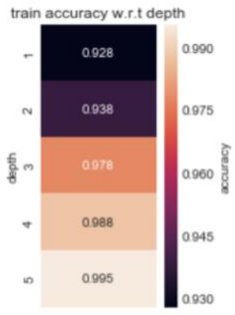
```
0.9298245614035088
{'max_depth': 4}
```

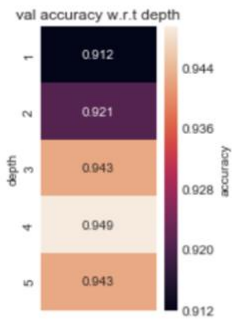
11 Performing Decision Trees On Mammograph Data Set

```
In [29]: classifier = tree.DecisionTreeClassifier(criterion='entropy')
         depth_list = [1, 2, 3, 4, 5]
         params = {"max_depth": depth_list}
         grid_dt_mam = GridSearchCV(classifier, params, return_train_score = True, cv = 5)
         grid_dt_mam.fit(mam_X_train, mam_Y_train)
```

```
Out[29]: GridSearchCV(cv=5, error_score='raise',
                estimator=DecisionTreeClassifier(class_weight=None, criterion='entropy', max_d
                     max_features=None, max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, presort=False, random_state=None,
                     splitter='best'),
                fit_params=None, iid=True, n_jobs=1,
                param_grid={'max_depth': [1, 2, 3, 4, 5]}, pre_dispatch='2*n_jobs',
                refit=True, return_train_score=True, scoring=None, verbose=0)
```

```
In [30]: train_acc = grid_dt_breast.cv_results_['mean_train_score'].reshape(-1,1)
         draw_heatmap_linear(train_acc, 'train accuracy', depth_list)

         val_acc = grid_dt_breast.cv_results_['mean_test_score'].reshape(-1,1)
         draw_heatmap_linear(val_acc, 'val accuracy', depth_list)
```





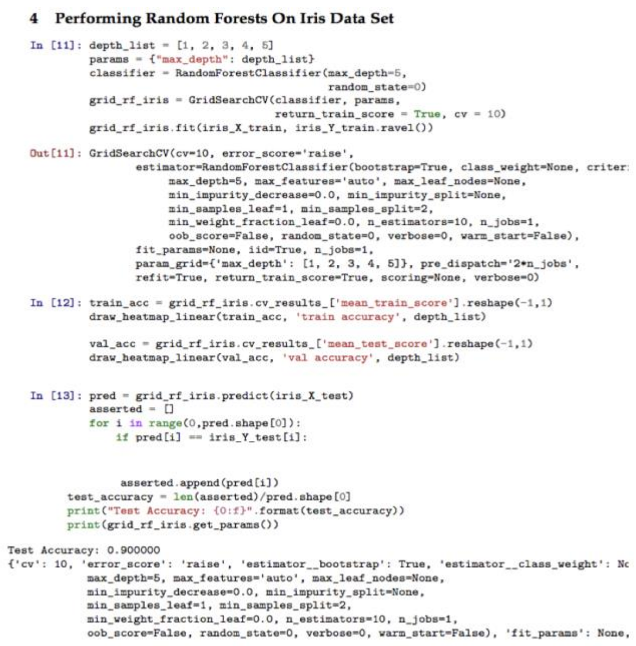
```
In [31]: pred = grid_dt_mam.predict(mam_X_test)
         asserted = []
         for i in range(0,pred.shape[0]):
             if pred[i] == mam_Y_test[i]:

                 asserted.append(pred[i])
          test_accuracy = len(asserted)/pred.shape[0]
          print("Test Accuracy: {0:f}".format(test_accuracy))
          print(grid_dt_mam.best_params_)
```

```
Test Accuracy: 0.891566
{'max_depth': 1}
```

3.2.1 Decision Trees Classifier Performance Analysis

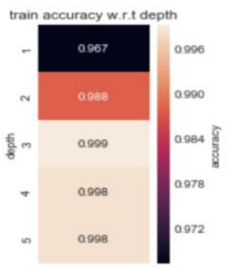
There is a really interesting observation that can be made in this part. Decision trees gave a reliable test accuracy for the iris data set. (~.96), similarly for the breast cancer data set (~.92). But the observation that I want to mention is that for the mammographic masses data set, it gave approximately the same test accuracy as the k-nearest neighbors classifier. One way to explain this is the low dimensionality of the mammographic masses data set. K-nearest neighbors classifier is observed to be not reliable for high dimensionality data sets of this type. It is the complete opposite, it is reliable when the data sets had low dimensionality (iris data set). The performance for decision trees classifier resulted as about the same efficiency as k-nearest neighbors classifier when it comes to low dimensionality data medical data sets. It did perform well for the iris data set, which is similar in dimension to the mammographic masses data set, but for the mammographic masses, its performance is probably not greater than .90 test accuracy since it is a medical data set that might require more features if you want to predict a tumor as benign or malign. More information is needed (features) so that the prediction can be accurate for such a medical condition. The last observation I want to make is that decision trees works efficiently when it comes to highly dimensional data sets (breast cancer data set) and it can be used as a good predictor to determine if a person's given conditions can be predicted as positive or negative for breast cancer diagnosis using decision trees classifier.

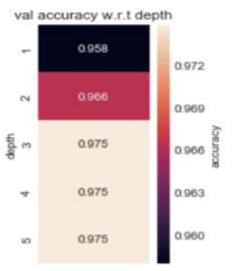
3.3 Random Forests Classifier



4 Performing Random Forests On Iris Data Set

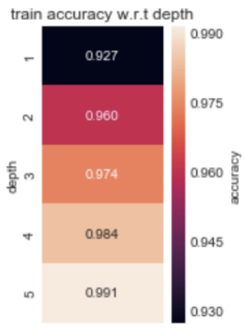
```
In [11]: depth_list = [1, 2, 3, 4, 5]
         params = {"max_depth": depth_list}
         classifier = RandomForestClassifier(max_depth=5,
                                             random_state=0)
         grid_rf_iris = GridSearchCV(classifier, params,
                                     return_train_score = True, cv = 10)
         grid_rf_iris.fit(iris_X_train, iris_Y_train.ravel())

Out[11]: GridSearchCV(cv=10, error_score='raise',
                estimator=RandomForestClassifier(bootstrap=True, class_weight=None, criter
                     max_depth=5, max_features='auto', max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
                     oob_score=False, random_state=0, verbose=0, warm_start=False),
                fit_params=None, iid=True, n_jobs=1,
                param_grid={'max_depth': [1, 2, 3, 4, 5]}, pre_dispatch='2*n_jobs',
                refit=True, return_train_score=True, scoring=None, verbose=0)

In [12]: train_acc = grid_rf_iris.cv_results_['mean_train_score'].reshape(-1,1)
         draw_heatmap_linear(train_acc, 'train accuracy', depth_list)

         val_acc = grid_rf_iris.cv_results_['mean_test_score'].reshape(-1,1)
         draw_heatmap_linear(val_acc, 'val accuracy', depth_list)


In [13]: pred = grid_rf_iris.predict(iris_X_test)
         asserted = []
         for i in range(0,pred.shape[0]):
             if pred[i] == iris_Y_test[i]:


               asserted.append(pred[i])
      test_accuracy = len(asserted)/pred.shape[0]
      print("Test Accuracy: {0:f}".format(test_accuracy))
      print(grid_rf_iris.get_params())

Test Accuracy: 0.900000
{'cv': 10, 'error_score': 'raise', 'estimator__bootstrap': True, 'estimator__class_weight': Nc
          max_depth=5, max_features='auto', max_leaf_nodes=None,
          min_impurity_decrease=0.0, min_impurity_split=None,
          min_samples_leaf=1, min_samples_split=2,
          min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
          oob_score=False, random_state=0, verbose=0, warm_start=False), 'fit_params': None,
```

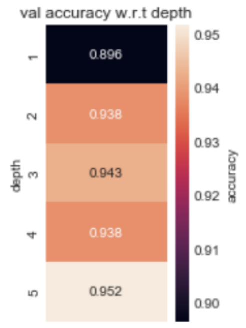




8 Performing Random Forests On Breast Cancer Data Set

```
In [22]: depth_list = [1, 2, 3, 4, 5]
         params = {"max_depth": depth_list}
         classifier = RandomForestClassifier(max_depth=5, random_state=0)
         grid_rf_breast = GridSearchCV(classifier, params, return_train_score = True, cv = 10)
         grid_rf_breast.fit(breast_X_train, breast_Y_train.ravel())

Out[22]: GridSearchCV(cv=10, error_score='raise',
                estimator=RandomForestClassifier(bootstrap=True, class_weight=None, criterion=
                     max_depth=5, max_features='auto', max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
                     oob_score=False, random_state=0, verbose=0, warm_start=False),
                fit_params=None, iid=True, n_jobs=1,
                param_grid={'max_depth': [1, 2, 3, 4, 5]}, pre_dispatch='2*n_jobs',
                refit=True, return_train_score=True, scoring=None, verbose=0)

In [23]: train_acc = grid_rf_breast.cv_results_['mean_train_score'].reshape(-1,1)
         draw_heatmap_linear(train_acc, 'train accuracy', depth_list)

         val_acc = grid_rf_breast.cv_results_['mean_test_score'].reshape(-1,1)
         draw_heatmap_linear(val_acc, 'val accuracy', depth_list)
```

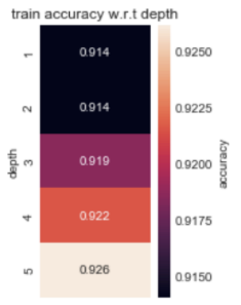


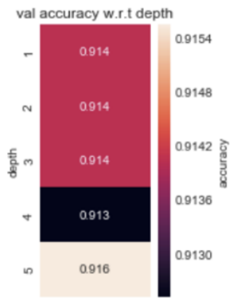


```
In [24]: pred = grid_rf_breast.predict(breast_X_test)
         asserted = []
         for i in range(0,pred.shape[0]):
             if pred[i] == breast_Y_test[i]:

              asserted.append(pred[i])
     test_accuracy = len(asserted)/pred.shape[0]
     print("Test Accuracy: {0:f}".format(test_accuracy))
     print(grid_rf_breast.get_params())

Test Accuracy: 0.956140
{'cv': 10, 'error_score': 'raise', 'estimator__bootstrap': True, 'estimator__class_weight': No
            max_depth=5, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
            oob_score=False, random_state=0, verbose=0, warm_start=False), 'fit_params': None,
```

12 Performing Random Forests On Mammograph Data Set

```
In [32]: depth_list = [1, 2, 3, 4, 5]
         params = {"max_depth": depth_list}
         classifier = RandomForestClassifier(max_depth=5, random_state=0)
         grid_rf_mam = GridSearchCV(classifier, params, return_train_score = True)
         grid_rf_mam.fit(mam_X_train, mam_Y_train.ravel())

Out[32]: GridSearchCV(cv=None, error_score='raise',
                estimator=RandomForestClassifier(bootstrap=True, class_weight=None, criterion=
                     max_depth=5, max_features='auto', max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
                     oob_score=False, random_state=0, verbose=0, warm_start=False),
                fit_params=None, iid=True, n_jobs=1,
                param_grid={'max_depth': [1, 2, 3, 4, 5]}, pre_dispatch='2*n_jobs',
                refit=True, return_train_score=True, scoring=None, verbose=0)

In [33]: train_acc = grid_rf_mam.cv_results_['mean_train_score'].reshape(-1,1)
         draw_heatmap_linear(train_acc, 'train accuracy', depth_list)

         val_acc = grid_rf_mam.cv_results_['mean_test_score'].reshape(-1,1)
         draw_heatmap_linear(val_acc, 'val accuracy', depth_list)
```





```
In [34]: pred = grid_rf_mam.predict(mam_X_test)
         asserted = []
         for i in range(0,pred.shape[0]):
             if pred[i] == mam_Y_test[i]:

             asserted.append(pred[i])
     test_accuracy = len(asserted)/pred.shape[0]
     print("Test Accuracy: {0:f}".format(test_accuracy))
     print(grid_rf_mam.get_params())

Test Accuracy: 0.885542
{'cv': None, 'error_score': 'raise', 'estimator__bootstrap': True, 'estimator__class_weight':
            max_depth=5, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
            oob_score=False, random_state=0, verbose=0, warm_start=False), 'fit_params': None,
```

3.3.1 Random Forests Classifier Performance Analysis

Observing the results of test accuracy of random forests classifier, one observation that can be made is that this classifier is the most effective when it comes to high dimensionality data sets. It gave a test accuracy of ~.95 for the breast cancer data set, which is the one with the highest dimension. This tells us that the most accurate way to predict a disease or a tumor, random forests classifier will work the most effectively out of these three classifiers. Another observation I want to reiterate is the fact that the test accuracy for the mammographic data set stays almost unchanged for these three classifiers.

4 Conclusion

Running these three classifiers helped with giving a good observation on how to handle machine learning in combination with medical techniques. The clearest observation is that when working with medical data sets, high dimensionality is helpful in order to effectively predict a medical condition. More information (features) is needed so that machine learning can effectively help the field of medicine. Another important result to point out is that k-nearest neighbors classifier is good for making predictions within small data sets and a bad predictor within large data sets. Whereas random forests classifier is the most effective for making important predictions of medical conditions given that the data set has high dimensionality and large enough data points. Medical conditions such as cancer, tumors, etc., can be effectively predicted with the use of machine learning. Not necessarily rely completely on machine learning methods, but I mean incorporating the techniques adds a complement to the prediction of such medical conditions. For more information on the hyper-parameters and how the experiments ran and how all this was done, take a look at the jupyter notebooks file attached which was written in python.

5 References

[1] Dua, D. and Karra Taniskidou, E. (2017). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[2] R.A. Fisher, Michael Marshall (1936). UCI Machine Learning Repository – Iris Data Set [https://archive.ics.uci.edu/ml/datasets/iris]. Irvine, CA: University of California, School of Information and Computer Science.

[3] Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian (1996). UCI Machine Learning Repository – Breast Cancer Wisconsin [https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)]. Irvine, CA: University of California, School of Information and Computer Science.

[4] Matthias Elter, Prof. Dr. Rüdiger Schulz-Wendtland (2007). UCI Machine Learning Repository – Mammographic Mass Data Set [http://archive.ics.uci.edu/ml/datasets/mammographic+mass]. Irvine, CA: University of California, School of Information and Computer Science.

[5] Miguel Garcia, Dr. Tu (2018). Assignment #6 – Implementation of cross validation done by me with the help of Dr. Tu's starter code. Implementation is included in attached file and another version can be found in: [https://sites.google.com/site/ucsdcogs118awinter2018/assignments/homework6]. La Jolla, CA: University of California, San Diego

[6] Miguel Garcia, Dr. Tu (2018). Assignment #6 – Implementation of grid search done by me with the help of Dr. Tu's starter code. Implementation is included in attached file and another version can be found in: [https://sites.google.com/site/ucsdcogs118awinter2018/assignments/homework6]. La Jolla, CA: University of California, San Diego

[7] Scikit-learn: GridSearchCV, (2018). [http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html].

[8] Dr. Tu (2018). Assignment #6 and Assignment #5 – Implementation of heat map drawing written in the assignments' starter code by Dr. Tu. For knn heat map: [https://sites.google.com/site/ucsdcogs118awinter2018/assignments/homework6], and for the linear heat map: [https://sites.google.com/site/ucsdcogs118awinter2018/assignments/homework5] La Jolla, CA: University of California, San Diego

[9] Scikit-learn: KNeighborsClassifiers, (2018). [http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html].

[10] Scikit-learn: DecisionTreeClassifier, (2018). [http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html].

[11] Scikit-learn: RandomForestClassifier, (2018). [http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html].