Joelle Fitzgerald
HIDS 509 HW #1 Task 2

Homework Task 2: Use at least two additional classification algorithms from SKlearn package and classify the same two groups of images (tumor vs normal)

- You can get additional points if you run more than two classification algorithms.
- Compare the accuracy of the all classification runs (including those done in the classroom), determine which algorithm performed the best

Summary of steps:

In Task 2, after loading the pathology image classification libraries in python, the selected images are loaded into a structured directory using a 'load_image_files' function. This function is run on specified image files and assigned to a data frame. The data frame type allows us to split the data into train and test datasets for predictive classification analysis using the sklearn library package and the 'train_test_split' function. Using various training models with parameter optimization, our data will be trained to predict the intended outcome - classifying images as 'normal' or 'tumor'. The outcome is then predicted using clf.predict using the features of the test dataset (X_test). Classification metrics of predictions from the trained model are printed after running the code chunk. The first model I used to train the data was logistic regression. From this classification model, a 95% accuracy rate was produced for correct classification of tumor versus normal tissue. Using a SVM model, a 95% accuracy rate was also obtained with parameters kernel set to RBF and gamma set to auto. Using random forest to train the dataset, a 93% accuracy was obtained with n_estimators set to 100, max_depth = 30, and random state at 0. The lowest performing models were neural networks (hideen_layer_size = 300 and random state of 1) with an accuracy of 58% and decision tree model (max_depth = 30, random state =1) with an accuracy of 77%. K-means nearest neighbor model performed the best and produced a 98% accuracy rate with n_neighbors set to 3.

(See screenshots of results below with classification report)

Predict

```
[ ]  1 y_pred = clf.predict(X_test)
```

Report

```
1 print("Classification report for - \n{}:\n{}\n".format(
2     clf, metrics.classification_report(y_test, y_pred)))
```

```
Classification report for -
GridSearchCV(estimator=SVC(),
             param_grid=[{'C': [1, 10, 100, 1000], 'kernel': ['linear']},
                         {'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001],
                          'kernel': ['rbf']}]):
              precision    recall  f1-score   support

           0       0.96      0.96      0.96        25
           1       0.94      0.94      0.94        18

    accuracy                           0.95        43
   macro avg       0.95      0.95      0.95        43
weighted avg       0.95      0.95      0.95        43
```

```
1 ##### Logistic Regression #####
2 from sklearn.linear_model import LogisticRegression
3 model = LogisticRegression(random_state=45, max_iter=1500)
4 model.fit(X_train, y_train)
5 y_pred_LR = model.predict(X_test)
6 print("Logistic Regression Classification report for - \n{}:\n{}\n".format(
7     model, metrics.classification_report(y_test, y_pred_LR)))
8
```

```
Logistic Regression Classification report for -
LogisticRegression(max_iter=1500, random_state=45):
              precision    recall  f1-score   support

           0       0.96      0.96      0.96        25
           1       0.94      0.94      0.94        18

    accuracy                           0.95        43
   macro avg       0.95      0.95      0.95        43
weighted avg       0.95      0.95      0.95        43
```

```
#### SVM ####
from sklearn.svm import SVC
model_svm = SVC(kernel='rbf', gamma='auto')
model_svm.fit(X_train, y_train)
y_pred_svm = model_svm.predict(X_test)
print("SVM report for - \n{}:\n{}\n".format(
    model_svm, metrics.classification_report(y_test, y_pred_svm)))

```

```
SVM report for -
SVC(gamma='auto'):
              precision    recall  f1-score   support

           0       1.00      0.92      0.96        25
           1       0.90      1.00      0.95        18

    accuracy                           0.95        43
   macro avg       0.95      0.96      0.95        43
weighted avg       0.96      0.95      0.95        43
```

```
#### Decision Tree Classifier #####
from sklearn.tree import DecisionTreeClassifier
model_decisionTree = DecisionTreeClassifier(max_depth=30, random_state=1) # max_depth s
model_decisionTree.fit(X_train, y_train)
y_pred_decisionTree = model_decisionTree.predict(X_test)
print("Decision Tree report for - \n{}:\n{}\n".format(
    model_decisionTree, metrics.classification_report(y_test, y_pred_decisionTree)))

```

```
Decision Tree report for -
DecisionTreeClassifier(max_depth=30, random_state=1):
              precision    recall  f1-score   support

           0       0.76      0.88      0.81        25
           1       0.79      0.61      0.69        18

    accuracy                           0.77        43
   macro avg       0.77      0.75      0.75        43
weighted avg       0.77      0.77      0.76        43
```

```
##### Random Forest #####
from sklearn.ensemble import RandomForestClassifier
model_RF = RandomForestClassifier(n_estimators=100, max_depth=30, random_state=0)
model_RF.fit(X_train, y_train)
y_pred_RF = model_RF.predict(X_test)
print("Random Forest Classification report for - \n{}:\n{}\n".format(
    model_RF, metrics.classification_report(y_test, y_pred_RF)))

```

```
Random Forest Classification report for -
RandomForestClassifier(max_depth=30, random_state=0):
              precision    recall  f1-score   support

           0       0.92      0.96      0.94        25
           1       0.94      0.89      0.91        18

    accuracy                           0.93        43
   macro avg       0.93      0.92      0.93        43
weighted avg       0.93      0.93      0.93        43
```

```
#### Neural Network #####
from sklearn import neural_network
from sklearn.neural_network import MLPClassifier
model_NeuralNet = MLPClassifier(hidden_layer_sizes=(300,),random_state=1) #, max_iter=1500)
model_NeuralNet.fit(X_train, y_train)
y_pred_NeuralNet = model_NeuralNet.predict(X_test)
print("Neural Network Classification report for - \n{}:\n{}\n".format(
    model_NeuralNet, metrics.classification_report(y_test, y_pred_NeuralNet)))

```

```
Neural Network Classification report for -
MLPClassifier(hidden_layer_sizes=(300,), random_state=1):
              precision    recall  f1-score   support

           0       0.58      1.00      0.74        25
           1       0.00      0.00      0.00        18

    accuracy                           0.58        43
   macro avg       0.29      0.50      0.37        43
weighted avg       0.34      0.58      0.43        43
```

```
#### k-Means Nearest Neighbor #####
from sklearn.neighbors import KNeighborsClassifier
model_kmeans = KNeighborsClassifier(n_neighbors=3)
model_kmeans.fit(X_train, y_train)
y_pred_kmeans = model_kmeans.predict(X_test)
print("k-Means Nearest Neighbor Classification report for -Loading...-\n".format(
    model_kmeans, metrics.classification_report(y_test, y_pred_kmeans)))

```

```
k-Means Nearest Neighbor Classification report for -
KNeighborsClassifier(n_neighbors=3):
              precision    recall  f1-score   support

           0       1.00      0.96      0.98        25
           1       0.95      1.00      0.97        18

    accuracy                           0.98        43
   macro avg       0.97      0.98      0.98        43
weighted avg       0.98      0.98      0.98        43
```