5293 hw4

April 12, 2018

```
In [1]: import pandas as pd
        import numpy as np
        from sklearn.model_selection import train_test_split,cross_val_score,GridSearchCV
        from sklearn.tree import DecisionTreeRegressor,DecisionTreeClassifier
        from sklearn.ensemble import RandomForestRegressor,RandomForestClassifier
        from sklearn.metrics import mean_squared_error
```

```
In [2]: df = pd.read_csv("Flight Delays Data.csv")
        df.head(5)
```

Out[2]:

	Year	Month	DayofMonth	DayOfWeek	Carrier	OriginAirportID	DestAirportID	\
0	2013	4	19	5	DL	11433	13303	
1	2013	4	19	5	DL	14869	12478	
2	2013	4	19	5	DL	14057	14869	
3	2013	4	19	5	DL	15016	11433	
4	2013	4	19	5	DL	11193	12892	

	CRSDepTime	DepDelay	DepDel15	CRSArrTime	ArrDelay	ArrDel15	Cancelled
0	837	-3.0	0.0	1138	1.0	0.0	0.0
1	1705	0.0	0.0	2336	-8.0	0.0	0.0
2	600	-4.0	0.0	851	-15.0	0.0	0.0
3	1630	28.0	1.0	1903	24.0	1.0	0.0
4	1615	-6.0	0.0	1805	-11.0	0.0	0.0

1 Step 1: Remove Unnecessary Rows and Columns

```
In [3]: df2 = df[df["Cancelled"]!=1]
        df3 = df2.drop(df2.columns[[9,12,13]],axis = 1)
        df3.head(5)
```

Out[3]:

	Year	Month	DayofMonth	DayOfWeek	Carrier	OriginAirportID	DestAirportID	\
0	2013	4	19	5	DL	11433	13303	
1	2013	4	19	5	DL	14869	12478	
2	2013	4	19	5	DL	14057	14869	
3	2013	4	19	5	DL	15016	11433	
4	2013	4	19	5	DL	11193	12892	

```
   CRSDepTime  DepDelay  CRSArrTime  ArrDelay
0         837      -3.0        1138       1.0
1        1705       0.0        2336      -8.0
2         600      -4.0         851     -15.0
3        1630      28.0        1903      24.0
4        1615      -6.0        1805     -11.0
```

2 Step2: Variable checking

3 a) Convert time values to datetime objects

```
In [4]: import datetime
        def fix_time(row):
            x = row["CRSDepTime"]
            time_str = str(int(x)).zfill(4)
            time_str_hour = int(time_str[0:2])
            if time_str_hour == 24:
                return 0
            time_str_minute = int(time_str[2:])

            month = int(row["Month"])
            day = int(row["DayofMonth"])
            return datetime.datetime(2013,month,day,time_str_hour,time_str_minute)

In [5]: df3["CRSDepTime"] = df3[['CRSDepTime','Month','DayofMonth']].apply(lambda row : fix_ti

In [6]: def fix_time2(row):
            x = row["CRSArrTime"]
            time_str = str(int(x)).zfill(4)
            time_str_hour = int(time_str[0:2])
            if time_str_hour == 24:
                return 0
            time_str_minute = int(time_str[2:])

            month = int(row["Month"])
            day = int(row["DayofMonth"])
            return datetime.datetime(2013,month,day,time_str_hour,time_str_minute)

In [7]: df3["CRSArrTime"] = df3[['CRSArrTime','Month','DayofMonth']].apply(lambda row : fix_ti
```

4 b) Add FlightTime variable & Convert Carrier Variable

```
In [8]: df3["FlightTime"] = 0
        count = 0
        for i in set(df3["Carrier"]):
            df3.loc[df3["Carrier"] == i, "Carrier"] = count
            count+=1
```

```
In [9]: df3.head(10)
```

Out[9]:

	Year	Month	DayofMonth	DayOfWeek	Carrier	OriginAirportID	DestAirportID	\
0	2013	4	19	5	4	11433	13303	
1	2013	4	19	5	4	14869	12478	
2	2013	4	19	5	4	14057	14869	
3	2013	4	19	5	4	15016	11433	
4	2013	4	19	5	4	11193	12892	
5	2013	4	19	5	4	10397	15016	
6	2013	4	19	5	4	15016	10397	
7	2013	4	19	5	4	10397	14869	
8	2013	4	19	5	4	10397	10423	
9	2013	4	19	5	4	11278	10397	

	CRSDepTime	DepDelay	CRSArrTime	ArrDelay	FlightTime
0	2013-04-19 08:37:00	-3.0	2013-04-19 11:38:00	1.0	0
1	2013-04-19 17:05:00	0.0	2013-04-19 23:36:00	-8.0	0
2	2013-04-19 06:00:00	-4.0	2013-04-19 08:51:00	-15.0	0
3	2013-04-19 16:30:00	28.0	2013-04-19 19:03:00	24.0	0
4	2013-04-19 16:15:00	-6.0	2013-04-19 18:05:00	-11.0	0
5	2013-04-19 17:26:00	-1.0	2013-04-19 18:18:00	-19.0	0
6	2013-04-19 19:00:00	0.0	2013-04-19 21:33:00	-1.0	0
7	2013-04-19 21:45:00	15.0	2013-04-19 23:56:00	24.0	0
8	2013-04-19 21:57:00	33.0	2013-04-19 23:33:00	34.0	0
9	2013-04-19 19:00:00	323.0	2013-04-19 20:55:00	322.0	0

```
In [10]: df4=df3.values

In [11]: for i in range(df4.shape[0]):
             df4[i,11]=(df4[i,9]-df4[i,7]).total_seconds()/60
```

5 c) Convert negative DepDelay to 0

```
In [12]: for i in range(df4.shape[0]):
                 if df4[i,8] < 0:
                     df4[i,8] = 0
```

6 d) Delete Timestamp Variables

```
In [13]: df5 = np.delete(df4,[7,9],1)
```

7 Regression & RandomForest

```
In [14]: y = df5[:,7]
         X = np.delete(df5,7,1)

In [15]: X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.2, random_state =
```

```
In [16]: dt1 = DecisionTreeRegressor(max_depth = 8,min_samples_split = 20,min_impurity_decrease
         rf1 = RandomForestRegressor(max_depth = 8,min_samples_split = 20,min_impurity_decrease
         dt1.fit(X_train,y_train)
         rf1.fit(X_train,y_train)

Out[16]: RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=8,
                    max_features='auto', max_leaf_nodes=None,
                    min_impurity_decrease=0.01, min_impurity_split=None,
                    min_samples_leaf=1, min_samples_split=20,
                    min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
                    oob_score=False, random_state=None, verbose=0, warm_start=False)

In [17]: dt1_scores = cross_val_score(dt1,X_train,y_train,cv=5)
         rf1_scores = cross_val_score(rf1,X_train,y_train,cv=5)
         print("Average cross-validation score for Regression Tree: {:.4f}".format(dt1_scores.
         print("Average cross-validation score for Random Forest(Regression): {:.4f}".format(r

Average cross-validation score for Regression Tree: 0.9269
Average cross-validation score for Random Forest(Regression): 0.9274
```

8 Classification & RandomForest

```
In [18]: y2 = np.where(y>0,1,0)
         X_train,X_test,y_train2,y_test2 = train_test_split(X,y2,test_size = 0.2, random_state

In [19]: dt2 = DecisionTreeClassifier(max_depth = 8,min_samples_split = 20,min_impurity_decreas
         rf2 = RandomForestClassifier(max_depth = 8,min_samples_split = 20,min_impurity_decreas
         dt2.fit(X_train,y_train2)
         rf2.fit(X_train,y_train2)

Out[19]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                     max_depth=8, max_features='auto', max_leaf_nodes=None,
                     min_impurity_decrease=0.01, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=20,
                     min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
                     oob_score=False, random_state=None, verbose=0,
                     warm_start=False)

In [20]: dt2_scores = cross_val_score(dt2,X_train,y_train2,cv=5)
         rf2_scores = cross_val_score(rf2,X_train,y_train2,cv=5)
         print("Average cross-validation score for Classification Tree: {:.4f}".format(dt2_sco
         print("Average cross-validation score for Random Forest(Classification): {:.4f}".forma

Average cross-validation score for Classification Tree: 0.7813
Average cross-validation score for Random Forest(Classification): 0.7561
```

9 Evaluation

10 a) Regression Tree + Random Forest

```
In [21]: y_pred_dt1 = dt1.predict(X_test)
         y_pred_rf1 = rf1.predict(X_test)
         MSE_dt1 = mean_squared_error(y_test,y_pred_dt1)
         MSE_rf1 = mean_squared_error(y_test,y_pred_rf1)
         print("MSE for Regression Tree: {:.4f}".format(MSE_dt1))
         print("MSE for Random Forest (Regression): {:.4f}".format(MSE_rf1))
```

```
MSE for Regression Tree: 89.4430
MSE for Random Forest (Regression): 88.6433
```

11 b) Classification Tree + Random Forest

```
In [22]: y_pred_dt2 = dt2.predict(X_test)
         y_pred_rf2 = rf2.predict(X_test)
         MSE_dt2 = mean_squared_error(y_test2,y_pred_dt2)
         MSE_rf2 = mean_squared_error(y_test2,y_pred_rf2)
         print("MSE for Regression Tree: {:.4f}".format(MSE_dt2))
         print("MSE for Random Forest (Regression): {:.4f}".format(MSE_rf2))
```

```
MSE for Regression Tree: 0.2184
MSE for Random Forest (Regression): 0.2161
```