5293_hw2_cm3700

March 22, 2018

```
In [1]: import pandas as pd
        import numpy as np
        from sklearn.model_selection import train_test_split
        from sklearn import linear_model
        from sklearn import preprocessing
        from sklearn.metrics import mean_squared_error
        from sklearn.tree import DecisionTreeRegressor
        from sklearn.tree import export_graphviz
```

```
In [2]: df = pd.read_excel("Assignment 2-3 Credit Model Data.xls")
```

1 1. Adjust CR column with 0/1

Delete unnecessary columns and convert column CR

```
In [3]: df1 = df.drop(["Exchange:Ticker","S&P Entity ID","Company Type","Geographic Locations"
        df1_x = df1.iloc[:,:-1]
        df1_x = df1_x.replace("-",np.nan)
        df1_x = df1_x.replace("NM",np.nan)
        df1_x = df1_x.as_matrix()
        n_row = df1_x.shape[0]
        n_col = df1_x.shape[1]
        y = df.iloc[:,-1].as_matrix()
        for i in range(n_row):
            if y[i] == "-":
                y[i] = 0
            else:
                y[i] = 1
```

2 2. Data Clean (Missing Values)

Missing Values - Entire Row

```
In [4]: for i in range(n_row):
            if not np.any(df1_x[i,]):
                df1_x = np.delete(df1_x,i,0)
```

Missing values - Entire Column

```
In [5]: for i in range(n_col):
            if not np.any(df1_x[:,i]):
                df1_x = np.delete(df1_x,i,1)
```

Missing value - if entire class is empty, randomly assign 0 or 1

```
In [6]: data_class = np.linspace(0,90,10).astype(int)
        for i in range(9):
            for j in range(n_row):
                if pd.isnull(df1_x[j,data_class[i]:data_class[i+1]]).all():
                    df1_x[j,data_class[i]:data_class[i+1]] =np.random.randint(2,size = 10)
```

Missing value - forward fill or backward fill

```
In [7]: for i in range(9):
            for j in range(n_row):
                T_value = pd.isnull(df1_x[j,data_class[i]:data_class[i+1]])
                F_value = np.logical_not(T_value)
                if T_value[0] == True:
                    forward = np.where(F_value)[0][0]
                    df1_x[j,data_class[i]:data_class[i+1]][0:forward] = df1_x[j,data_class[i]:c
                if T_value[9] == True:
                    backward = np.where(F_value)[0][-1]
                    df1_x[j,data_class[i]:data_class[i+1]][backward+1:] = df1_x[j,data_class[i
```

Missing value - fill average

```
In [8]: for i in range(9):
            for j in range(n_row):
                if np.isnan(df1_x[j,data_class[i]:data_class[i+1]]).any() == True:
                    T_value = np.isnan(df1_x[j,data_class[i]:data_class[i+1]])
                    idx = np.where(T_value)
                    df1_x[j,data_class[i]:data_class[i+1]][idx] = (df1_x[j,data_class[i]:data_c
```

Check if all points are filled and export X data

```
In [9]: np.isnan(df1_x).any()
        data_X = pd.DataFrame(df1_x)
        data_X.to_csv("outputX.csv")
```

3 3. Data split

```
In [ ]: x_train,x_test,y_train,y_test = train_test_split(df1_x, y,test_size=0.2,random_state =
```

4 4. Method 1: OLS

Data split and preparation

```
In [23]: #average variable
         OLS_x_train = []
         OLS_x_test = []
         for i in range(9):
             for j in range(len(x_train)):
                 OLS_x_train.append(np.average(x_train[j,data_class[i]:data_class[i+1]]))

         for i in range(9):
             for j in range(len(x_test)):
                 OLS_x_test.append(np.average(x_test[j,data_class[i]:data_class[i+1]]))

         OLS_x_train = np.reshape(OLS_x_train, (8000,9))
         OLS_x_test = np.reshape(OLS_x_test, (2000,9))

         # Last number
         # idx_col = np.linspace(9,89,9).astype(int)

         # OLS_x_train = []
         # OLS_x_test = []
         # for i in range(8):
         #     for j in range(len(x_train)):
         #                 OLS_x_train.append(x_train[j,idx_col[i]])
         # for i in range(8):
         #     for j in range(len(x_test)):
         #                 OLS_x_test.append(x_test[j,idx_col[i]])

         # OLS_x_train = np.reshape(OLS_x_train,(8000,8))

         # OLS_x_test = np.reshape(OLS_x_test,(2000,8))
```

Train OLS model

```
In [24]: OLS = linear_model.LinearRegression()
         OLS.fit(X=OLS_x_train,y=y_train)

Out[24]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)

In [25]: R2_train = OLS.score(OLS_x_train,y_train)
         print("R2 for training dataset: %.4f "% R2_train)

R2 for training dataset: 0.0003


In [26]: R2_test = OLS.score(OLS_x_test,y_test)
         y_predict = OLS.predict(OLS_x_test)
```

```
        MSE_OLS = mean_squared_error(y_true = y_test, y_pred = y_predict)
        print("R2 for H dataset: %.4f "% R2_test)
        print("MSE for H dataset: %.4f "% MSE_OLS)
```

```
R2 for H dataset: -0.0003
MSE for H dataset: 0.0319
```

5 5. Method 2: Regression Tree

```
In [27]: tree_model = DecisionTreeRegressor(max_depth=8,min_samples_split=20,min_impurity_decre
         tree_model.fit(x_train,y_train)
```

```
Out[27]: DecisionTreeRegressor(criterion='mse', max_depth=8, max_features=None,
                    max_leaf_nodes=None, min_impurity_decrease=0.001,
                    min_impurity_split=None, min_samples_leaf=1,
                    min_samples_split=20, min_weight_fraction_leaf=0.0,
                    presort=False, random_state=None, splitter='best')
```

```
In [28]: y_predict2 = tree_model.predict(x_test)
         MSE_tree = mean_squared_error(y_test,y_predict2)
         R2_tree_test = tree_model.score(x_test,y_test)
         print("R2 for H dataset: %.4f "% R2_tree_test)
         print("MSE for H dataset: %.4f "% MSE_tree)
```

```
R2 for H dataset: 0.4544
MSE for H dataset: 0.0174
```

6 6. Assess methods

1)OLS doesn't work with our dataset. It has a close to 0 R2. This is reasonable since our dataset has too many missing points and we don't have a good way to fill them.

2)Decision tree performs way better than OLS with 45.44% R2 and a almost 50% lower MSE.

3)R2: Decision Tree is 45% more than OLS; MSE: Decision tree is 48% lower than OLS