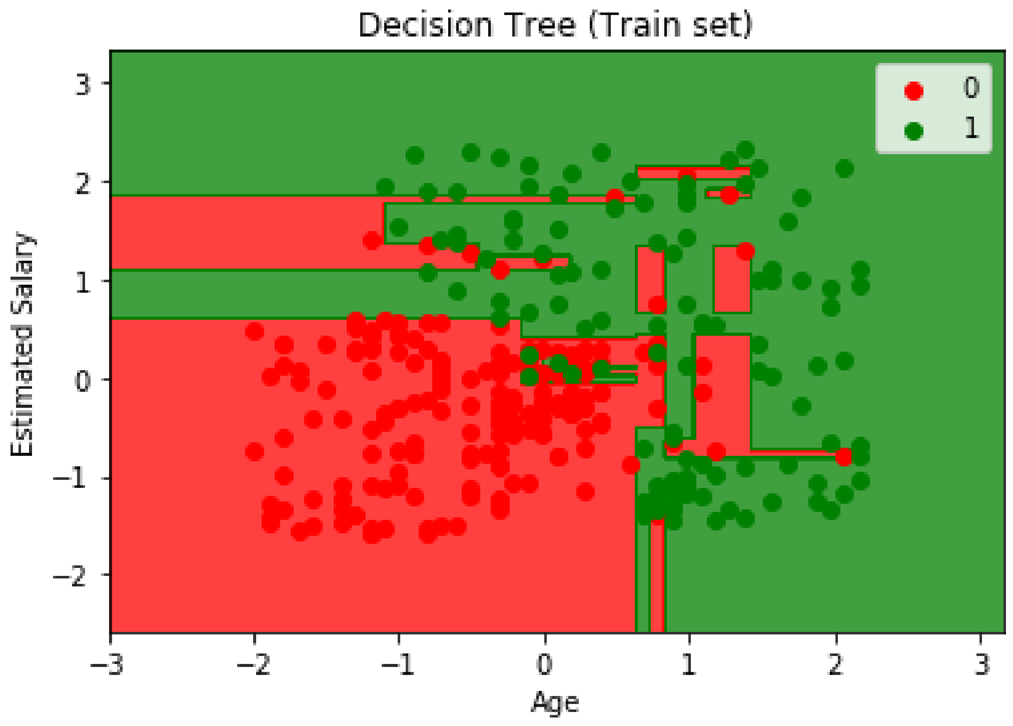
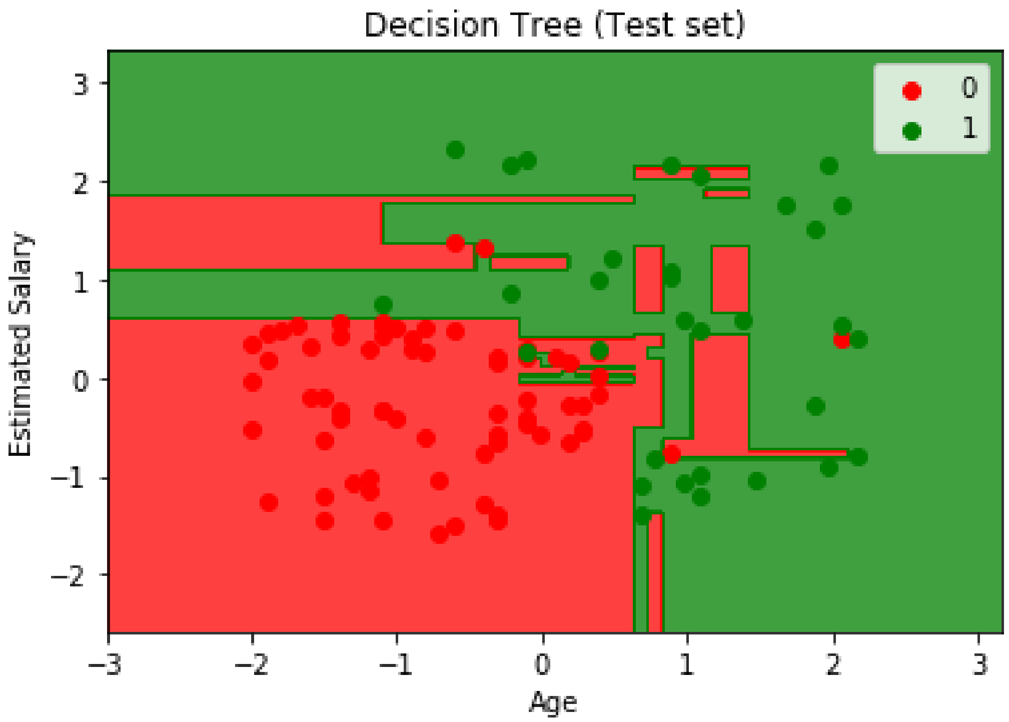
Decision Tree

September 21, 2018

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt

In [2]: dataset = pd.read_csv("/media/coea/A4F698A8F6987BEC/A-Z/SVM/SVM/Social_Network_Ads.csv")

        x = dataset.iloc[:,[2,3]].values

        y = dataset.iloc[:,4].values

In [3]: from sklearn.cross_validation import train_test_split

        x_train,x_test,y_train,y_test = train_test_split(x,y, test_size = .25,random_state = 0)

/home/coea/virtualenvironment/vikram/local/lib/python2.7/site-packages/sklearn/cross_validation.
  "This module will be removed in 0.20.", DeprecationWarning)


In [4]: from sklearn.preprocessing import StandardScaler
        sc_x = StandardScaler()
        x_train = sc_x.fit_transform(x_train)
        x_test = sc_x.transform(x_test)

/home/coea/virtualenvironment/vikram/local/lib/python2.7/site-packages/sklearn/utils/validation.
  warnings.warn(msg, DataConversionWarning)


In [5]: from sklearn.tree import DecisionTreeClassifier
        classifier = DecisionTreeClassifier(criterion = 'entropy',random_state=0)
        classifier.fit(x_train,y_train)

Out[5]: DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
                    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=0,
                    splitter='best')

In [6]: y_pred = classifier.predict(x_test)
In [7]: from sklearn.metrics import confusion_matrix
        cn = confusion_matrix(y_test,y_pred)

In [12]: cn

Out[12]: array([[62,  6],
                [ 3, 29]])

In [8]: # Visualising the Test set results
        from matplotlib.colors import ListedColormap
        X_set, y_set = x_train, y_train
        X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() +
                             np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() +
        plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1
                     alpha = 0.75, cmap = ListedColormap(('red', 'green')))
        plt.xlim(X1.min(), X1.max())
        plt.ylim(X2.min(), X2.max())
        for i, j in enumerate(np.unique(y_set)):
            plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                        c = ListedColormap(('red', 'green'))(i), label = j)
        plt.title('Decision Tree (Train set)')
        plt.xlabel('Age')
        plt.ylabel('Estimated Salary')
        plt.legend()
        plt.show()
```



```
In [10]: # Visualising the Test set results
         from matplotlib.colors import ListedColormap
         X_set, y_set = x_test, y_test
         X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max()
                              np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max()
         plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X
                      alpha = 0.75, cmap = ListedColormap(('red', 'green')))
         plt.xlim(X1.min(), X1.max())
         plt.ylim(X2.min(), X2.max())
         for i, j in enumerate(np.unique(y_set)):
             plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                         c = ListedColormap(('red', 'green'))(i), label = j)
         plt.title('Decision Tree (Test set)')
         plt.xlabel('Age')
         plt.ylabel('Estimated Salary')
         plt.legend()
         plt.show()
```


Decision Tree (Test set)
● 0
● 1
Estimated Salary
Age