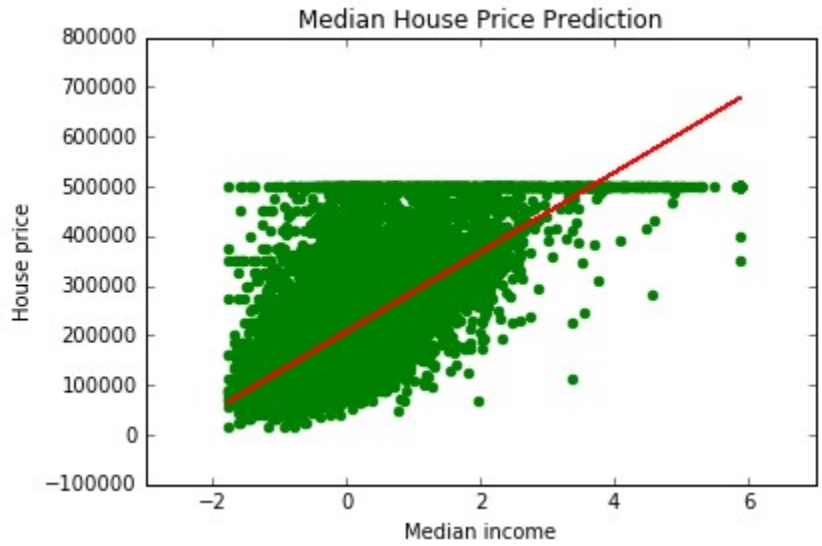
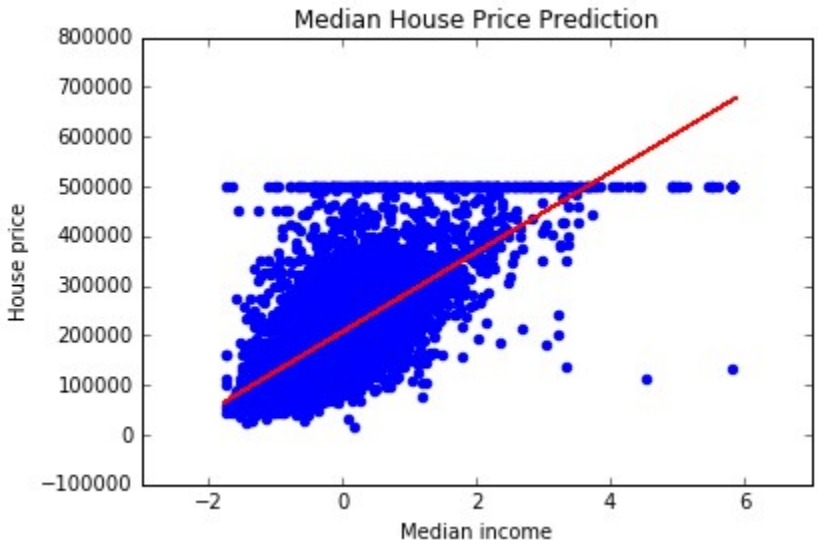
```
Python 3.5.6 |Anaconda 4.2.0 (64-bit)| (default, Aug 26 2018, 16:05:27) [MSC v.1900 64
bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 5.1.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]:

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

In [2]: houseData = pd.read_csv('housing.csv')
   ...:
   ...: print(houseData.iloc[0:6,:].values)
   ...: X = houseData.iloc[:,:-1].values
   ...: y = houseData.iloc[:,9].values
[[-122.23 37.88 41 880 129.0 322 126 8.3252 'NEAR BAY' 452600]
 [-122.22 37.86 21 7099 1106.0 2401 1138 8.3014 'NEAR BAY' 358500]
 [-122.24 37.85 52 1467 190.0 496 177 7.2574 'NEAR BAY' 352100]
 [-122.25 37.85 52 1274 235.0 558 219 5.6431 'NEAR BAY' 341300]
 [-122.25 37.85 52 1627 280.0 565 259 3.8462 'NEAR BAY' 342200]
 [-122.25 37.85 52 919 213.0 413 193 4.0368 'NEAR BAY' 269700]]

In [3]: from sklearn.preprocessing import Imputer
   ...: missingValues = Imputer(missing_values="NaN",strategy="mean",axis=0)

In [4]: X[:,0:8] = missingValues.fit_transform(X[:,0:8])
   ...:

In [5]: from sklearn.preprocessing import LabelEncoder
   ...: X_labelencoder = LabelEncoder()
   ...:
   ...: X[:,8] = X_labelencoder.fit_transform(X[:,8])

In [6]: X_labelencoder.classes_
   ...:
Out[6]: array(['<1H OCEAN', 'INLAND', 'ISLAND', 'NEAR BAY', 'NEAR OCEAN'],
dtype=object)

In [7]: from sklearn.preprocessing import OneHotEncoder
   ...: X_ohe = OneHotEncoder( categorical_features=[8])
   ...: X = X_ohe.fit_transform(X).toarray()
   ...:

In [8]: from sklearn.cross_validation import train_test_split
   ...: X_train,X_test,y_train,y_test = train_test_split
(X,y,test_size=0.2,random_state=0)

In [9]: from sklearn.preprocessing import StandardScaler
   ...: scale = StandardScaler()
   ...: X_train = scale.fit_transform(X_train)
   ...: X_test = scale.fit_transform(X_test)
   ...:

In [10]: from sklearn.linear_model import LinearRegression
    ...: linearRegressor = LinearRegression()
    ...: linearRegressor.fit(X_train,y_train)
```

```
    ...:
Out[10]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)

In [11]: LRpredict = linearRegressor.predict(X_test)
    ...:

In [12]: linearRegressor.score(X_train,y_train)
    ...:
Out[12]: 0.64696546198529004

In [13]: linearRegressor.score(X_test,y_test)
Out[13]: -1.4353247599973323e+21

In [14]: from sklearn.metrics import mean_squared_error
    ...: from math import sqrt
    ...: LRrms = sqrt(mean_squared_error(y_test,LRpredict))

In [15]: from sklearn.tree import DecisionTreeRegressor
    ...: DTregressor = DecisionTreeRegressor(random_state=4)
    ...: DTregressor.fit(X_train,y_train)
    ...:
Out[15]:
DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
           max_leaf_nodes=None, min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, presort=False, random_state=4,
           splitter='best')

In [16]: DTpredict = DTregressor.predict(X_test)
    ...:

In [17]: DTregressor.score(X_train,y_train)
Out[17]: 1.0

In [18]: DTregressor.score(X_test,y_test)
Out[18]: 0.5418211162276273

In [19]: from sklearn.metrics import mean_squared_error
    ...: from math import sqrt
    ...: DTrms = sqrt(mean_squared_error(y_test,DTpredict))

In [20]: from sklearn.ensemble import RandomForestRegressor
    ...: RFregressor = RandomForestRegressor(n_estimators=20,random_state=2)
    ...: RFregressor.fit(X_train,y_train)

Out[20]:
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', max_leaf_nodes=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           n_estimators=20, n_jobs=1, oob_score=False, random_state=2,
           verbose=0, warm_start=False)

In [21]: RFpredict = RFregressor.predict(X_test)
    ...:

In [22]: RFregressor.score(X_train,y_train)
Out[22]: 0.9699115208222413

In [23]: RFregressor.score(X_test,y_test)
Out[23]: 0.75218399106388856

In [24]: from sklearn.metrics import mean_squared_error
    ...: from math import sqrt
```

```
    ...: RFrms = sqrt(mean_squared_error(y_test,RFpredict))

In [25]: X = np.delete(X,[0,1,2,3,4,5,6,7,8,9,10,11],axis=1)

In [26]: from sklearn.cross_validation import train_test_split
    ...: X_train,X_test,y_train,y_test = train_test_split(X,y,
test_size=1/4,random_state=0)
    ...:

In [27]: X_train.shape
    ...:
Out[27]: (15480, 1)

In [28]: from sklearn.preprocessing import StandardScaler
    ...: scaler = StandardScaler()
    ...: X_train = scaler.fit_transform(X_train)
    ...: X_test = scaler.fit_transform(X_test)
    ...:

In [29]: from sklearn.linear_model import LinearRegression
    ...: linearRegressor1 = LinearRegression()
    ...: linearRegressor1.fit(X_train,y_train)
Out[29]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)

In [30]: linearRegressor1.predict(X_test)
    ...:
Out[30]:
array([ 219691.90105431,  287764.77990757,  227926.05134686, ...,
        197854.04791684,  276628.39482425,  245289.27735735])

In [31]: linearRegressor1.score(X_train,y_train)
Out[31]: 0.48061930819884535

In [32]: linearRegressor1.score(X_test,y_test)
    ...:
Out[32]: 0.45147717106069024

In [33]: plt.scatter(X_train,y_train,color='green')
    ...: plt.plot(X_train,linearRegressor1.predict(X_train),color='red')
    ...: plt.title('Median House Price Prediction')
    ...: plt.xlabel('Median income')
    ...: plt.ylabel('House price')
    ...: plt.show()
```



```
In [34]: plt.scatter(X_test,y_test,color='blue')
    ...: plt.plot(X_train,linearRegressor1.predict(X_train),color='red')
    ...: plt.title('Median House Price Prediction')
    ...: plt.xlabel('Median income')
    ...: plt.ylabel('House price')
    ...: plt.show()
```




```
In [35]:
```