In [1]:
```
pwd
```

Out[1]:
```
'C:\\Users\\sethw\\Desktop\\Caltech Bootcamp\\Machine Learning'
```

In [2]:
```
cd Practice Projects Datasets
```

```
C:\Users\sethw\Desktop\Caltech Bootcamp\Machine Learning\Practice Projects Datasets
```

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
```

In [4]:
```
#import housing dataset
df = pd.read_excel('housing.xlsx')
```

In [5]:
```
df.head()
```

Out[5]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	m
0	-122.23	37.88	41	880	129.0	322	126	
1	-122.22	37.86	21	7099	1106.0	2401	1138	
2	-122.24	37.85	52	1467	190.0	496	177	
3	-122.25	37.85	52	1274	235.0	558	219	
4	-122.25	37.85	52	1627	280.0	565	259	

In [6]:
```
df.tail()
```

Out[6]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	household
20635	-121.09	39.48	25	1665	374.0	845	33(
20636	-121.21	39.49	18	697	150.0	356	11
20637	-121.22	39.43	17	2254	485.0	1007	43
20638	-121.32	39.43	18	1860	409.0	741	34
20639	-121.24	39.37	16	2785	616.0	1387	53(

In [7]:
```
df.shape
```

Out[7]:
```
(20640, 10)
```

In [8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  int64  
 3   total_rooms         20640 non-null  int64  
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  int64  
 6   households          20640 non-null  int64  
 7   median_income       20640 non-null  float64
 8   ocean_proximity     20640 non-null  object 
 9   median_house_value  20640 non-null  int64  
dtypes: float64(4), int64(5), object(1)
memory usage: 1.6+ MB
```

In [9]:
```
#check columns for null values
df.isnull().sum()
```

Out[9]:
```
longitude               0
latitude                0
housing_median_age      0
total_rooms             0
total_bedrooms        207
population              0
households              0
median_income           0
ocean_proximity         0
median_house_value      0
dtype: int64
```

In [10]:
```
#replace null values in total_bedrooms column with mean value
df['total_bedrooms'] = df['total_bedrooms'].fillna(df['total_bedrooms'].mean())
```

In [11]:
```
#check success of implementation of mean for null values
df.isnull().sum()
```

Out[11]:
```
longitude             0
latitude              0
housing_median_age    0
total_rooms           0
total_bedrooms        0
population            0
households            0
median_income         0
ocean_proximity       0
median_house_value    0
dtype: int64
```

In [12]:
```
# ocean_proximity is categorical. Implemnent one hot encoding
df = pd.get_dummies(df, columns=['ocean_proximity'])
```

In [13]:
```
#view new dataset
df.head()
```

Out[13]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	m
0	-122.23	37.88	41	880	129.0	322	126	
1	-122.22	37.86	21	7099	1106.0	2401	1138	
2	-122.24	37.85	52	1467	190.0	496	177	
3	-122.25	37.85	52	1274	235.0	558	219	
4	-122.25	37.85	52	1627	280.0	565	259	

In [14]:
```
df.shape
```

Out[14]:
```
(20640, 14)
```

In [15]:
```
df.columns
```

Out[15]:
```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'median_house_value', 'ocean_proximity_<1H OCEAN',
       'ocean_proximity_INLAND', 'ocean_proximity_ISLAND',
       'ocean_proximity_NEAR BAY', 'ocean_proximity_NEAR OCEAN'],
      dtype='object')
```

In [16]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
```

In [17]:
```
target = df['median_house_value']
df = df.drop(['median_house_value'], axis=1)
```

In [18]:
```
target.head()
```

Out[18]:
```
0    452600
1    358500
2    352100
3    341300
4    342200
Name: median_house_value, dtype: int64
```

In [19]:
```
df.head()
```

Out[19]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	m
0	-122.23	37.88	41	880	129.0	322	126	
1	-122.22	37.86	21	7099	1106.0	2401	1138	
2	-122.24	37.85	52	1467	190.0	496	177	
3	-122.25	37.85	52	1274	235.0	558	219	
4	-122.25	37.85	52	1627	280.0	565	259	

In [20]:
```
#split the dataset into 80% training set
x = df
y = target

x_train,x_test,y_train,y_test = train_test_split(x,y, test_size=0.2, random_state=42)
```

In [21]:
```
#standardize training and test data
scaler = StandardScaler()

scaler.fit(x_train)

x_train = scaler.transform(x_train)
x_test = scaler.transform(x_test)
```

Linear Regression

In [22]:
```
#perform Linear Regression
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from math import sqrt

lm = LinearRegression()
lm.fit(x_train,y_train)
```

Out[22]:
```
LinearRegression()
```

In [23]:
```
print(lm.intercept_)
print(lm.coef_)
```

```
207194.69373788772
[-53826.64801649 -54415.6961445   13889.86618856 -13094.25116219
  43068.18184187 -43403.43242732  18382.19632373  75167.77476625
   6424.35562855 -12492.68755954   2319.63426583   2459.94570874
   5435.0100562 ]
```

In [24]:
```
preds = lm.predict(x_test)
print(sqrt(mean_squared_error(y_test, preds)))
```

```
70031.41991955662
```

In [25]:
```
# taking a look at house price data again
target.describe()
```

Out[25]:
```
count     20640.000000
mean     206855.816909
std      115395.615874
min       14999.000000
25%      119600.000000
50%      179700.000000
75%      264725.000000
max      500001.000000
Name: median_house_value, dtype: float64
```

Decision Tree

In [29]:
```
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import accuracy_score

decision_tree = DecisionTreeRegressor()

decision_tree.fit(x_train,y_train)
```

Out[29]:
```
DecisionTreeRegressor()
```

In [30]:
```
y_preds = decision_tree.predict(x_test)

accuracy = accuracy_score(y_preds, y_test)
print(accuracy)
```

```
0.023982558139534885
```

In [31]:
```
print(sqrt(mean_squared_error(y_test, y_preds)))
```

```
68972.1937160397
```

Inference from aboce^

1. Bad Accuracy
2. Smaller MSE, thus more accurate than linear regression model

Random Forest

In [34]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import accuracy_score

random_forest = RandomForestRegressor()

random_forest.fit(x_train, y_train)
```

Out[34]:
```
RandomForestRegressor()
```

In [36]:
```
y_predictions = random_forest.predict(x_test)

print(sqrt(mean_squared_error(y_test, y_preds)))
```

```
48951.208660767086
```

Inferences from above^

1. accuracy has imporved
2. MSE is smaller, indicating that random forest regressor is best model

In [ ]:
```

```