In [1]:

```
import pandas as pd
import numpy as np
```

In [2]:

```
from sklearn.preprocessing import Imputer
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor

from sklearn import metrics
```

In [3]:

```
housing = pd.read_csv('housing1.csv')
housing
```

Out[3]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	housel
0	-122.23	37.88	41	880	129.0	322	
1	-122.22	37.86	21	7099	1106.0	2401	
2	-122.24	37.85	52	1467	190.0	496	
3	-122.25	37.85	52	1274	235.0	558	
4	-122.25	37.85	52	1627	280.0	565	
5	-122.25	37.85	52	919	213.0	413	
6	-122.25	37.84	52	2535	489.0	1094	
7	-122.25	37.84	52	3104	687.0	1157	
8	-122.26	37.84	42	2555	665.0	1206	
9	-122.25	37.84	52	3549	707.0	1551	
10	-122.26	37.85	52	2202	434.0	910	
11	-122.26	37.85	52	3503	752.0	1504	
12	-122.26	37.85	52	2491	474.0	1098	
13	-122.26	37.84	52	696	191.0	345	
14	-122.26	37.85	52	2643	626.0	1212	
15	-122.26	37.85	50	1120	283.0	697	
16	-122.27	37.85	52	1966	347.0	793	
17	-122.27	37.85	52	1228	293.0	648	
18	-122.26	37.84	50	2239	455.0	990	
19	-122.27	37.84	52	1503	298.0	690	
20	-122.27	37.85	40	751	184.0	409	
21	-122.27	37.85	42	1639	367.0	929	
22	-122.27	37.84	52	2436	541.0	1015	
23	-122.27	37.84	52	1688	337.0	853	
24	-122.27	37.84	52	2224	437.0	1006	
25	-122.28	37.85	41	535	123.0	317	
26	-122.28	37.85	49	1130	244.0	607	
27	-122.28	37.85	52	1898	421.0	1102	
28	-122.28	37.84	50	2082	492.0	1131	
29	-122.28	37.84	52	729	160.0	395	
...	...	...	...	...	...	...	
20610	-121.56	39.10	28	2130	484.0	1195	
20611	-121.55	39.10	27	1783	441.0	1163	
20612	-121.56	39.08	26	1377	289.0	761	
20613	-121.55	39.09	31	1728	365.0	1167	

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	housel
20614	-121.54	39.08	26	2276	460.0	1455	
20615	-121.54	39.08	23	1076	216.0	724	
20616	-121.53	39.08	15	1810	441.0	1157	
20617	-121.53	39.06	20	561	109.0	308	
20618	-121.55	39.06	25	1332	247.0	726	
20619	-121.56	39.01	22	1891	340.0	1023	
20620	-121.48	39.05	40	198	41.0	151	
20621	-121.47	39.01	37	1244	247.0	484	
20622	-121.44	39.00	20	755	147.0	457	
20623	-121.37	39.03	32	1158	244.0	598	
20624	-121.41	39.04	16	1698	300.0	731	
20625	-121.52	39.12	37	102	17.0	29	
20626	-121.43	39.18	36	1124	184.0	504	
20627	-121.32	39.13	5	358	65.0	169	
20628	-121.48	39.10	19	2043	421.0	1018	
20629	-121.39	39.12	28	10035	1856.0	6912	
20630	-121.32	39.29	11	2640	505.0	1257	
20631	-121.40	39.33	15	2655	493.0	1200	
20632	-121.45	39.26	15	2319	416.0	1047	
20633	-121.53	39.19	27	2080	412.0	1082	
20634	-121.56	39.27	28	2332	395.0	1041	
20635	-121.09	39.48	25	1665	374.0	845	
20636	-121.21	39.49	18	697	150.0	356	
20637	-121.22	39.43	17	2254	485.0	1007	
20638	-121.32	39.43	18	1860	409.0	741	
20639	-121.24	39.37	16	2785	616.0	1387	

20640 rows × 10 columns

In [4]:
```
housing.info()
```

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

In [5]:
```
print(housing.head())
```

```
   longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0    -122.23     37.88                  41          880           129.0
1    -122.22     37.86                  21         7099          1106.0
2    -122.24     37.85                  52         1467           190.0
3    -122.25     37.85                  52         1274           235.0
4    -122.25     37.85                  52         1627           280.0

   population  households  median_income ocean_proximity  median_house_value
0         322         126         8.3252        NEAR BAY              452600
1        2401        1138         8.3014        NEAR BAY              358500
2         496         177         7.2574        NEAR BAY              352100
3         558         219         5.6431        NEAR BAY              341300
4         565         259         3.8462        NEAR BAY              342200
```

In [6]:
```
housing.isnull().sum()
```

Out[6]:
```
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 [7]:

```
print ("Total_bedrooms column Mode is  "+str(housing["total_bedrooms"].mode())
+"\n")
print(housing["total_bedrooms"].describe())
```

```
Total_bedrooms column Mode is  0    280.0
dtype: float64

count    20433.000000
mean       537.870553
std        421.385070
min          1.000000
25%        296.000000
50%        435.000000
75%        647.000000
max       6445.000000
Name: total_bedrooms, dtype: float64
```

In [8]:

```
print(housing.iloc[:,4:5].head())
imputer = Imputer(np.nan,strategy ="median")
imputer.fit(housing.iloc[:,4:5])
housing.iloc[:,4:5] = imputer.transform(housing.iloc[:,4:5])
housing.isnull().sum()
```

```
   total_bedrooms
0           129.0
1          1106.0
2           190.0
3           235.0
4           280.0

E:\New folder (2)\lib\site-packages\sklearn\utils\deprecation.py:58: Deprecat
ionWarning: Class Imputer is deprecated; Imputer was deprecated in version 0.
20 and will be removed in 0.22. Import impute.SimpleImputer from sklearn inst
ead.
  warnings.warn(msg, category=DeprecationWarning)
```

Out[8]:

```
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 [9]:

```
labelEncoder = LabelEncoder()
print(housing["ocean_proximity"].value_counts())
housing["ocean_proximity"] = labelEncoder.fit_transform(housing["ocean_proximity"])
housing["ocean_proximity"].value_counts()
housing.describe()
```

```
<1H OCEAN     9136
INLAND        6551
NEAR OCEAN    2658
NEAR BAY      2290
ISLAND           5
Name: ocean_proximity, dtype: int64
```

Out[9]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	popul
count	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.0C
mean	-119.569704	35.631861	28.639486	2635.763081	536.838857	1425.47
std	2.003532	2.135952	12.585558	2181.615252	419.391878	1132.46
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.0C
25%	-121.800000	33.930000	18.000000	1447.750000	297.000000	787.0C
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	1166.0C
75%	-118.010000	37.710000	37.000000	3148.000000	643.250000	1725.0C
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	35682.0C

In [11]:

```
housing_ind = housing.drop("median_house_value",axis=1)
print(housing_ind.head())
housing_dep = housing["median_house_value"]
print("Medain Housing Values")
print(housing_dep.head())
```

```
   longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0    -122.23     37.88                  41          880           129.0
1    -122.22     37.86                  21         7099          1106.0
2    -122.24     37.85                  52         1467           190.0
3    -122.25     37.85                  52         1274           235.0
4    -122.25     37.85                  52         1627           280.0

   population  households  median_income  ocean_proximity
0         322         126         8.3252                3
1        2401        1138         8.3014                3
2         496         177         7.2574                3
3         558         219         5.6431                3
4         565         259         3.8462                3
Medain Housing Values
0    452600
1    358500
2    352100
3    341300
4    342200
Name: median_house_value, dtype: int64
```

In [12]:

```
#check for rand_state
X_train,X_test,y_train,y_test = train_test_split(housing_ind,housing_dep,test_
size=0.2,random_state=42)
#print(X_train.head())
#print(X_test.head())
#print(y_train.head())
#print(y_test.head())
print("X_train shape {} and size {}".format(X_train.shape,X_train.size))
print("X_test shape {} and size {}".format(X_test.shape,X_test.size))
print("y_train shape {} and size {}".format(y_train.shape,y_train.size))
print("y_test shape {} and size {}".format(y_test.shape,y_test.size))
```

```
X_train shape (16512, 9) and size 148608
X_test shape (4128, 9) and size 37152
y_train shape (16512,) and size 16512
y_test shape (4128,) and size 4128
```

In [13]:

```
X_train.head()
```

Out[13]:

```
       longitude  latitude  housing_median_age  total_rooms  total_bedrooms  population  housel
14196    -117.03     32.71                  33         3126           627.0        2300
 8267    -118.16     33.77                  49         3382           787.0        1314
17445    -120.48     34.66                   4         1897           331.0         915
14265    -117.11     32.69                  36         1421           367.0        1418
 2271    -119.80     36.78                  43         2382           431.0         874
```

In [14]:

```
independent_scaler = StandardScaler()
X_train = independent_scaler.fit_transform(X_train)
X_test = independent_scaler.transform(X_test)
print(X_train[0:5,:])
print("test data")
print(X_test[0:5,:])
```

```
E:\New folder (2)\lib\site-packages\sklearn\preprocessing\data.py:625: DataCo
nversionWarning: Data with input dtype int32, int64, float64 were all convert
ed to float64 by StandardScaler.
  return self.partial_fit(X, y)

[[ 1.27258656 -1.3728112   0.34849025  0.22256942  0.21122752  0.76827628
   0.32290591 -0.326196    2.00593172]
 [ 0.70916212 -0.87669601  1.61811813  0.34029326  0.59309419 -0.09890135
   0.6720272  -0.03584338  2.00593172]
 [-0.44760309 -0.46014647 -1.95271028 -0.34259695 -0.49522582 -0.44981806
  -0.43046109  0.14470145  2.00593172]
 [ 1.23269811 -1.38217186  0.58654547 -0.56148971 -0.40930582 -0.00743434
  -0.38058662 -1.01786438  2.00593172]
 [-0.10855122  0.5320839   1.14200767 -0.11956547 -0.25655915 -0.48587717
  -0.31496232 -0.17148831 -0.1124266 ]]
test data
[[ 0.28534728  0.1951     -0.28632369 -0.52286157 -0.24701249 -0.03030109
  -0.37008673 -1.15508475 -0.1124266 ]
 [ 0.06097472 -0.23549054  0.11043502  0.13841528 -0.24701249  0.12185077
   0.220532   -0.70865905 -0.1124266 ]
 [-1.42487026  1.00947776  1.85617335  0.54630997 -0.24701249 -0.10241931
   1.21539643 -0.21040155  1.29981228]
 [ 0.42994293 -0.63799909 -0.92113763  0.18808002 -0.24701249  0.24497944
  -0.01309052  0.97511311 -0.81854604]
 [-1.17058135  0.45719859  0.42784199 -0.13382109 -0.24701249 -0.31965346
  -0.18896365 -0.08179356  2.00593172]]

E:\New folder (2)\lib\site-packages\sklearn\base.py:462: DataConversionWarnin
g: Data with input dtype int32, int64, float64 were all converted to float64
by StandardScaler.
  return self.fit(X, **fit_params).transform(X)
E:\New folder (2)\lib\site-packages\ipykernel_launcher.py:3: DataConversionWa
rning: Data with input dtype int32, int64, float64 were all converted to floa
t64 by StandardScaler.
  This is separate from the ipykernel package so we can avoid doing imports u
ntil
```

In [15]:

```
#initantiate the linear regression
linearRegModel = LinearRegression(n_jobs=-1)
#fit the model to the training data (learn the coefficients)
linearRegModel.fit(X_train,y_train)
#print the intercept and coefficients
print("Intercept is "+str(linearRegModel.intercept_))
print("coefficients  is "+str(linearRegModel.coef_))
```

```
Intercept is 207194.69373788778
coefficients  is [-85854.94724101 -90946.06271148  14924.30655143 -17693.2340
5277
  48767.60670995 -43884.16852449  17601.31495096  77144.10164179
   -451.52015229]
```

In [16]:

```
y_pred = linearRegModel.predict(X_test)
```

In [17]:

```
print(np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
print(np.sqrt(metrics.mean_squared_error(y_train,linearRegModel.predict(X_trai
n))))
```

```
71147.87146118373
69361.0714290645
```

In [18]:

```
dtReg = DecisionTreeRegressor(max_depth=9)
dtReg.fit(X_train,y_train)
```

Out[18]:

```
DecisionTreeRegressor(criterion='mse', max_depth=9, 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=None, splitter='best')
```

In [19]:

```
dtReg_y_pred = dtReg.predict(X_test)
dtReg_y_pred
```

Out[19]:

```
array([ 60503.2556391 ,  75919.52054795, 478283.56097561, ...,
       488611.25      ,  75919.52054795, 211563.96963563])
```

In [20]:

```
print(np.sqrt(metrics.mean_squared_error(y_test,dtReg_y_pred)))
```

```
60981.93860192401
```

In [21]:

```
rfReg = RandomForestRegressor(30)
rfReg.fit(X_train,y_train)
```

Out[21]:

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', 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, n_estimators=30, n_jobs=None,
           oob_score=False, random_state=None, verbose=0, warm_start=False)
```

In [23]:

```
rfReg_y_pred = rfReg.predict(X_test)
print(len(rfReg_y_pred))
print(len(y_test))
print(rfReg_y_pred[0:5])
print(y_test[0:5])
```

```
4128
4128
[ 48646.66666667  71933.33333333 466903.7         278603.33333333
 245740.        ]
20046     47700
3024      45800
15663    500001
20484    218600
9814     278000
Name: median_house_value, dtype: int64
```

In [24]:

```
print(np.sqrt(metrics.mean_squared_error(y_test,rfReg_y_pred)))
```

```
50631.69730545329
```

In [ ]:

```

```