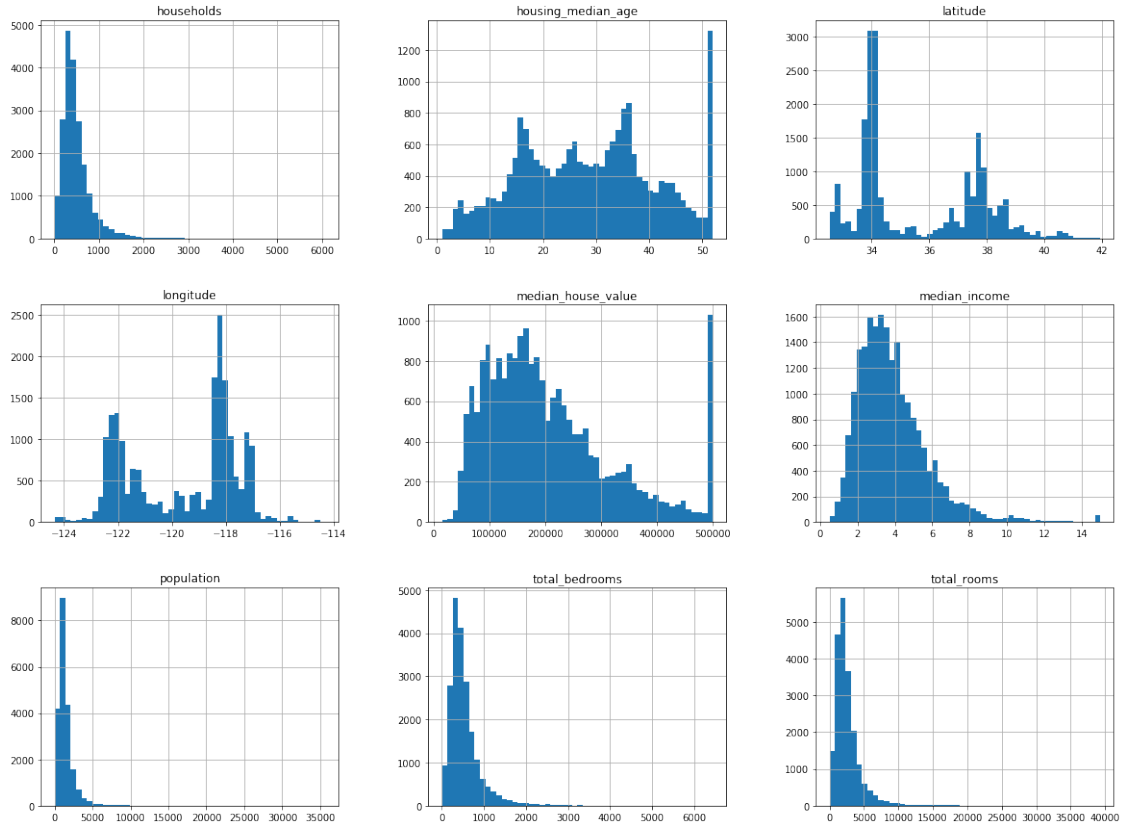
simplilearn project

May 30, 2019

In [129]:
```
import pandas as pd
import os
import numpy as np
import math
import matplotlib
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix
%matplotlib inline
import seaborn as sns
plt.show()
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
import statsmodels.api as sma
import statsmodels.formula.api as sm
from pandas.core import datetools
from statsmodels.stats.outliers_influence import variance_inflation_factor

from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn.metrics import roc_curve, auc
from statsmodels.tools.eval_measures import rmse
```

In [66]:
```
#1) Loading the Data

Housing = r"/Users/vaseekaranrajagopal/Desktop/housing.csv"
```

```
In [67]: House  = pd.read_csv(Housing)

In [69]: House.isnull().sum()

Out[69]: 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 [70]: #2) Handling Mising Values

         House['total_bedrooms'] = House['total_bedrooms'].fillna(House['total_bedrooms'].mean
         House.isnull().sum()

Out[70]: 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 [7]: %matplotlib inline
        import matplotlib.pyplot as plt
        House.hist(bins=50, figsize=(20,15))
        plt.show()
```



In [71]:
```
#3 Enable Categorica Data

lb_make = LabelEncoder()
House["proximity"] =lb_make.fit_transform(House["ocean_proximity"])
House[["proximity", "ocean_proximity"]].head(11)
```

Out[71]:

	proximity	ocean_proximity
0	3	NEAR BAY
1	3	NEAR BAY
2	3	NEAR BAY
3	3	NEAR BAY
4	3	NEAR BAY
5	3	NEAR BAY
6	3	NEAR BAY
7	3	NEAR BAY
8	3	NEAR BAY
9	3	NEAR BAY
10	3	NEAR BAY

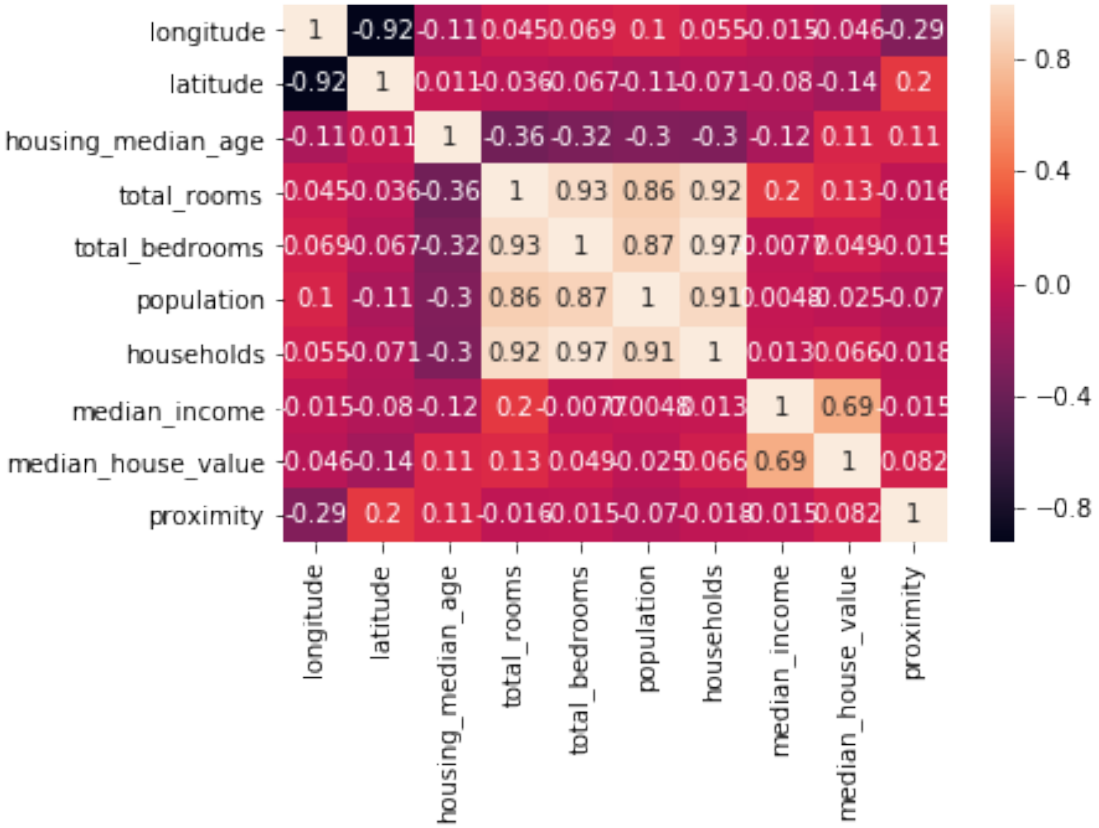
In [73]:
```
House.corr()
```

Out[73]:

	longitude	latitude	housing_median_age	total_rooms	\
longitude	1.000000	-0.924664	-0.108197	0.044568	

```
latitude            -0.924664  1.000000           0.011173    -0.036100
housing_median_age  -0.108197  0.011173           1.000000    -0.361262
total_rooms          0.044568 -0.036100          -0.361262     1.000000
total_bedrooms       0.069260 -0.066658          -0.318998     0.927253
population           0.099773 -0.108785          -0.296244     0.857126
households           0.055310 -0.071035          -0.302916     0.918484
median_income       -0.015176 -0.079809          -0.119034     0.198050
median_house_value  -0.045967 -0.144160           0.105623     0.134153
proximity           -0.289779  0.200974           0.112468    -0.015693

                    total_bedrooms  population  households  median_income  \
longitude                 0.069260    0.099773    0.055310      -0.015176
latitude                 -0.066658   -0.108785   -0.071035      -0.079809
housing_median_age       -0.318998   -0.296244   -0.302916      -0.119034
total_rooms               0.927253    0.857126    0.918484       0.198050
total_bedrooms            1.000000    0.873910    0.974725      -0.007682
population                0.873910    1.000000    0.907222       0.004834
households                0.974725    0.907222    1.000000       0.013033
median_income            -0.007682    0.004834    0.013033       1.000000
median_house_value        0.049454   -0.024650    0.065843       0.688075
proximity                -0.014688   -0.070282   -0.018186      -0.014957

                    median_house_value  proximity
longitude                    -0.045967  -0.289779
latitude                     -0.144160   0.200974
housing_median_age            0.105623   0.112468
total_rooms                   0.134153  -0.015693
total_bedrooms                0.049454  -0.014688
population                   -0.024650  -0.070282
households                    0.065843  -0.018186
median_income                 0.688075  -0.014957
median_house_value            1.000000   0.081750
proximity                     0.081750   1.000000
```

In [74]:
```
sns.heatmap( House.corr(), annot=True );
```




```
In [75]: House["proximity"].value_counts()

Out[75]: 0    9136
         1    6551
         4    2658
         3    2290
         2       5
         Name: proximity, dtype: int64

In [76]: House.columns

Out[76]: Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
                'total_bedrooms', 'population', 'households', 'median_income',
                'ocean_proximity', 'median_house_value', 'proximity'],
               dtype='object')

In [77]: House.dtypes

Out[77]: longitude             float64
         latitude              float64
         housing_median_age      int64
```

```
total_rooms             int64
total_bedrooms        float64
population              int64
households              int64
median_income         float64
ocean_proximity        object
median_house_value      int64
proximity               int64
dtype: object
```

```
In [78]: House.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 11 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        20640 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
proximity             20640 non-null int64
dtypes: float64(4), int64(6), object(1)
memory usage: 1.7+ MB
```

```
In [79]: House.describe()
```

Out[79]:

	longitude	latitude	housing_median_age	total_rooms	\
count	20640.000000	20640.000000	20640.000000	20640.000000	
mean	-119.569704	35.631861	28.639486	2635.763081	
std	2.003532	2.135952	12.585558	2181.615252	
min	-124.350000	32.540000	1.000000	2.000000	
25%	-121.800000	33.930000	18.000000	1447.750000	
50%	-118.490000	34.260000	29.000000	2127.000000	
75%	-118.010000	37.710000	37.000000	3148.000000	
max	-114.310000	41.950000	52.000000	39320.000000	

	total_bedrooms	population	households	median_income	\
count	20640.000000	20640.000000	20640.000000	20640.000000	
mean	537.870553	1425.476744	499.539680	3.870671	
std	419.266592	1132.462122	382.329753	1.899822	
min	1.000000	3.000000	1.000000	0.499900	
25%	297.000000	787.000000	280.000000	2.563400	
50%	438.000000	1166.000000	409.000000	3.534800	

```
75%        643.250000   1725.000000    605.000000       4.743250
max       6445.000000  35682.000000   6082.000000      15.000100

       median_house_value     proximity
count        20640.000000  20640.000000
mean        206855.816909      1.165843
std         115395.615874      1.420662
min          14999.000000      0.000000
25%         119600.000000      0.000000
50%         179700.000000      1.000000
75%         264725.000000      1.000000
max         500001.000000      4.000000
```

```
In [80]: print('Number of rows House set :',House.shape)
```

```
Number of rows House set : (20640, 11)
```

```
In [81]: House.head()
```

Out[81]:

	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	

	proximity
0	3
1	3
2	3
3	3
4	3

```
In [82]: House_sample =House.drop('ocean_proximity', axis = 1, inplace = True)
```

```
In [83]: House_sample = House.sample(n=500, random_state=0)
```

```
In [84]: House_sample.head()
```

Out[84]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	\
14740	-117.05	32.58	22	2101	399.0	

```
10101    -117.97     33.92                 32         2620           398.0
20566    -121.84     38.65                 29         3167           548.0
2670     -115.60     33.20                 37          709           187.0
15709    -122.43     37.79                 25         1637           394.0

       population  households  median_income  median_house_value  proximity
14740        1551         371         4.1518              136900          4
10101        1296         429         5.7796              241300          0
20566        1554         534         4.3487              200700          1
2670          390         142         2.4511               72500          1
15709         649         379         5.0049              460000          3
```

In [85]:
```
House_sample.isnull().sum()
```

Out[85]:
```
longitude             0
latitude              0
housing_median_age    0
total_rooms           0
total_bedrooms        0
population            0
households            0
median_income         0
median_house_value    0
proximity             0
dtype: int64
```

In [86]:
```
df = pd.DataFrame(House_sample)
df.head()
```

Out[86]:
```
       longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
14740    -117.05     32.58                  22         2101           399.0
10101    -117.97     33.92                  32         2620           398.0
20566    -121.84     38.65                  29         3167           548.0
2670     -115.60     33.20                  37          709           187.0
15709    -122.43     37.79                  25         1637           394.0

       population  households  median_income  median_house_value  proximity
14740        1551         371         4.1518              136900          4
10101        1296         429         5.7796              241300          0
20566        1554         534         4.3487              200700          1
2670          390         142         2.4511               72500          1
15709         649         379         5.0049              460000          3
```

In [87]:
```
House_sample = House.sample(n=500, random_state=0)
```

In [88]:
```
House_sample.columns
```

Out[88]:
```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'median_house_value', 'proximity'],
      dtype='object')
```

```
In [89]: np.random.seed(42)

In [90]: columnsTitles = ['longitude', 'latitude', 'housing_median_age', 'total_rooms',
                'total_bedrooms', 'population', 'households', 'median_income','proximity',
                'median_house_value', ]

In [91]: df1 = df.reindex(columns=columnsTitles)

In [92]: from sklearn.model_selection import train_test_split

         X = df1.iloc[:,0:9]
         y = df1.iloc[:,9:]

In [93]: df1.columns

Out[93]: Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
                'total_bedrooms', 'population', 'households', 'median_income',
                'proximity', 'median_house_value'],
               dtype='object')

In [94]: y.head()

Out[94]:        median_house_value
         14740              136900
         10101              241300
         20566              200700
         2670                72500
         15709              460000

In [99]: y.describe()

Out[99]:        median_house_value
         count          500.000000
         mean        206933.650000
         std         116784.588272
         min          43800.000000
         25%         119775.000000
         50%         180650.000000
         75%         265650.000000
         max         500001.000000

In [100]: X.head()

Out[100]:        longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
          14740    -117.05     32.58                  22         2101           399.0
          10101    -117.97     33.92                  32         2620           398.0
          20566    -121.84     38.65                  29         3167           548.0
          2670     -115.60     33.20                  37          709           187.0
          15709    -122.43     37.79                  25         1637           394.0
```

```
       population  households  median_income  proximity
14740        1551         371         4.1518          4
10101        1296         429         5.7796          0
20566        1554         534         4.3487          1
2670          390         142         2.4511          1
15709         649         379         5.0049          3
```

```
In [96]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_sta
```

```
In [33]: y_test.head()
```

```
Out[33]:        median_house_value
         11873               88900
         9219                57000
         3230                46300
         2371                58900
         4585               275000
```

```
In [101]: len(X) == len(y)
```

```
Out[101]: True
```

```
In [102]: X_test.head()
```

```
Out[102]:        longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
         11873    -117.37     34.00                  36          730           155.0
         9219     -120.27     37.12                  36         1219           258.0
         3230     -119.61     36.31                  25         1847           371.0
         2371     -119.56     36.70                  40         1195           326.0
         4585     -118.27     34.06                  26          513           338.0

                population  households  median_income  proximity
         11873         476         142         2.4306          1
         9219          639         245         1.9464          1
         3230         1460         353         1.8839          1
         2371         1135         315         2.1182          1
         4585         1204         321         1.4904          0
```

```
In [103]: print(X_train.shape)
          print(y_train.shape)
          print(X_test.shape)
          print(y_test.shape)
```

```
(400, 9)
(400, 1)
(100, 9)
(100, 1)
```

```
In [104]: lm = LinearRegression()
```

```
In [105]: lm = lm.fit(X_train, y_train)

In [106]: lm.coef_

Out[106]: array([[-4.25768829e+04, -4.46482921e+04,  1.56749894e+03,
                  -1.00484726e+01,  1.40694416e+02, -5.35149513e+01,
                   7.89817131e+01,  4.20994337e+04, -3.88394628e+03]])

In [107]: coefficients = pd.concat([pd.DataFrame(X_train.columns),pd.DataFrame(np.transpose(lm
          coefficients

Out[107]:                    0             0
          0           longitude -42576.882876
          1            latitude -44648.292112
          2  housing_median_age   1567.498944
          3         total_rooms    -10.048473
          4      total_bedrooms    140.694416
          5          population    -53.514951
          6          households     78.981713
          7       median_income  42099.433732
          8           proximity  -3883.946276

In [108]: lm.intercept_

Out[108]: array([-3503927.15494399])

In [109]: y_pred = lm.predict(X_test)

In [110]: r2_score(y_test, y_pred)

Out[110]: 0.5240794903823125

In [111]: X_train = sma.add_constant(X_train)
          X_test  = sma.add_constant(X_test)

In [112]: lm2 = sm.OLS(y_train, X_train).fit()

In [113]: lm2 = sm.OLS(y_train, X_train).fit()

In [114]: lm2.summary()

Out[114]: <class 'statsmodels.iolib.summary.Summary'>
          """
                                      OLS Regression Results
          ==============================================================================
          Dep. Variable:     median_house_value   R-squared:                       0.610
          Model:                            OLS   Adj. R-squared:                  0.601
          Method:                 Least Squares   F-statistic:                     67.84
          Date:                Thu, 31 Jan 2019   Prob (F-statistic):           2.70e-74
          Time:                        13:27:41   Log-Likelihood:                -5033.3
```

```
No. Observations:                 400   AIC:                          1.009e+04
Df Residuals:                     390   BIC:                          1.013e+04
Df Model:                           9
Covariance Type:            nonrobust
================================================================================
                          coef    std err          t      P>|t|      [0.025      0.975
--------------------------------------------------------------------------------
const               -3.504e+06   4.62e+05     -7.587      0.000   -4.41e+06    -2.6e+0
longitude           -4.258e+04   5234.790     -8.133      0.000   -5.29e+04   -3.23e+0
latitude            -4.465e+04   4922.780     -9.070      0.000   -5.43e+04    -3.5e+0
housing_median_age   1567.4989    306.908      5.107      0.000     964.098    2170.90
total_rooms           -10.0485      4.859     -2.068      0.039     -19.602      -0.49
total_bedrooms        140.6944     43.264      3.252      0.001      55.635     225.75
population            -53.5150      9.176     -5.832      0.000     -71.557     -35.47
households             78.9817     51.396      1.537      0.125     -22.067     180.03
median_income         4.21e+04   2674.819     15.739      0.000    3.68e+04    4.74e+0
proximity          -3883.9463   2769.837     -1.402      0.162   -9329.627    1561.73
==============================================================================
Omnibus:                       79.077   Durbin-Watson:                   1.934
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              155.447
Skew:                           1.069   Prob(JB):                     1.76e-34
Kurtosis:                       5.181   Cond. No.                     5.11e+05
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
[2] The condition number is large, 5.11e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

In [115]:
```
from scipy import stats
stats.shapiro(lm2.resid)
```

Out[115]:
```
(0.9403040409088135, 1.3602392649747497e-11)
```

In [116]:
```
y_pred2 = lm2.predict(X_test)
```

In [117]:
```
y_pred2.head()
```

Out[117]:
```
11873    130367.091677
9219     103142.835940
3230      65515.785062
2371      93961.897954
4585     117740.896701
dtype: float64
```

In [119]:
```
y_pred2.describe()
```

Out[119]:
```
count       100.000000
mean     196620.953533
```

```
std       89256.995441
min        7443.386625
25%      123432.093103
50%      189573.451314
75%      268940.652029
max      394340.104753
dtype: float64
```

In [120]:
```
coefficients = pd.concat([pd.DataFrame(X_train.columns),pd.DataFrame(np.transpose(lm
coefficients
```

Out[120]:

	0	0
0	const	-42576.882876
1	longitude	-44648.292112
2	latitude	1567.498944
3	housing_median_age	-10.048473
4	total_rooms	140.694416
5	total_bedrooms	-53.514951
6	population	78.981713
7	households	42099.433732
8	median_income	-3883.946276
9	proximity	NaN

In [121]:
```
print(np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
89773.80168965246
```

In [122]:
```
#Random Forest
clf = RandomForestClassifier(n_estimators = 100)
```

In [123]:
```
clf.fit(X_train, y_train)
```

```
/Users/vaseekaranrajagopal/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1: Data
  """Entry point for launching an IPython kernel.
```

Out[123]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       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=100, n_jobs=1,
                       oob_score=False, random_state=None, verbose=0,
                       warm_start=False)
```

In [124]:
```
y_pred = clf.predict(X_test)
```

In [125]:
```
cm = confusion_matrix(y_test,y_pred)
```

```
In [57]: print(metrics.accuracy_score(y_test, y_pred))

         print(1-metrics.accuracy_score(y_test, y_pred))
```

```
0.03
0.97
```

```
In [130]: #Decision Tree

          dtclf = DecisionTreeClassifier()

In [131]: dtclf.fit(X_train, y_train)

Out[131]: DecisionTreeClassifier(class_weight=None, criterion='gini', 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=None,
                      splitter='best')

In [132]: y_pred = dtclf.predict(X_test)
          y_pred

Out[132]: array([350000, 352800,  96000, 151800,  65000, 170800, 169200,  43800,
                 192000, 206200, 376000,  59100,  75000, 154200, 141100, 220100,
                  73200, 170800, 238000, 222600, 460000, 152800, 220700, 148000,
                 193800, 420000, 153900,  51300, 142300, 319700, 155300,  89400,
                 352800, 248600, 350000, 161000, 141600, 136900, 173300, 241900,
                 500001, 113500, 171300, 231800, 247800, 500001, 420000, 121200,
                 500001, 500001, 128600, 395300, 226900, 116200,  68300, 142100,
                 170800, 170800, 366700, 142300, 269900, 117000, 142300, 192000,
                 180400, 154200, 154200, 201300, 425000, 208300, 162000, 500001,
                 375700, 277600, 234500, 173300, 204300, 376000,  99400, 500001,
                  98300, 157500,  93800, 244900, 128700, 221400,  98300, 136400,
                  99400, 334100, 500001, 137500, 121600, 193800, 236100, 351900,
                  61800,  91100, 137500, 176100])

In [133]: dtclf.score(X_train, y_train)

Out[133]: 1.0

In [134]: accuracy_score(y_test, y_pred)

Out[134]: 0.04

In [ ]:
```