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

In [4]:
```
data = pd.read_csv('acs_ny.csv')
```

In [5]:
```
data.head()
# x_col = ['Acres'[0], 'FamilyType'[2], 'NumUnits'[7],
#        'OwnRent'[10], 'YearBuilt'[11], 'FoodStamp'[14],
#        'HeatingFuel'[15], 'Language'[17]]
```

Out[5]:

	Acres	FamilyIncome	FamilyType	NumBedrooms	NumChildren	NumPeople	NumRo
0	10-Jan	150	Married	4	1	3	9
1	10-Jan	180	Female Head	3	2	4	6
2	10-Jan	280	Female Head	4	0	2	8
3	10-Jan	330	Female Head	2	1	2	4
4	10-Jan	330	Male Head	3	1	2	5

In [6]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22745 entries, 0 to 22744
Data columns (total 18 columns):
Acres            22745 non-null object
FamilyIncome     22745 non-null int64
FamilyType       22745 non-null object
NumBedrooms      22745 non-null int64
NumChildren      22745 non-null int64
NumPeople        22745 non-null int64
NumRooms         22745 non-null int64
NumUnits         22745 non-null object
NumVehicles      22745 non-null int64
NumWorkers       22745 non-null int64
OwnRent          22745 non-null object
YearBuilt        22745 non-null object
HouseCosts       22745 non-null int64
ElectricBill     22745 non-null int64
FoodStamp        22745 non-null object
HeatingFuel      22745 non-null object
Insurance        22745 non-null int64
Language         22745 non-null object
dtypes: int64(10), object(8)
memory usage: 3.1+ MB
```

In [7]:
```
data.describe()
```

Out[7]:

	FamilyIncome	NumBedrooms	NumChildren	NumPeople	NumRooms	NumV
count	2.274500e+04	22745.000000	22745.000000	22745.000000	22745.000000	22745.
mean	1.102814e+05	3.385315	0.901165	3.390459	7.174764	2.1126
std	1.004539e+05	1.092477	1.159535	1.407659	2.345623	0.9691
min	5.000000e+01	0.000000	0.000000	2.000000	1.000000	0.0000
25%	5.254000e+04	3.000000	0.000000	2.000000	6.000000	2.0000
50%	8.700000e+04	3.000000	0.000000	3.000000	7.000000	2.0000
75%	1.338000e+05	4.000000	2.000000	4.000000	8.000000	3.0000
max	1.605000e+06	8.000000	12.000000	18.000000	21.000000	6.0000

The data is very clean with no missing values.

In [8]:
```
data.isnull().sum()
```

Out[8]:
```
Acres           0
FamilyIncome    0
FamilyType      0
NumBedrooms     0
NumChildren     0
NumPeople       0
NumRooms        0
NumUnits        0
NumVehicles     0
NumWorkers      0
OwnRent         0
YearBuilt       0
HouseCosts      0
ElectricBill    0
FoodStamp       0
HeatingFuel     0
Insurance       0
Language        0
dtype: int64
```

In [9]:
```
x = data.iloc[:, 2:]
y = data.iloc[:, 1:2]
```

We will make dummy variables for categorical variables: The categorical variables here are: Acres, FamilyType, NumUnits, OwnRent, YearBuilt, FoodStamp, HeatingFuel, Language

In [10]:
```
encoded_x = pd.get_dummies(x,drop_first = True)
```

In [11]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(encoded_x, y, test_size =
0.2, random_state = 0)
```

In [12]:
```
from sklearn.linear_model import LinearRegression
regression = LinearRegression()
lm = regression.fit(X_train,Y_train)
```

In [13]:
```
y_pred = regression.predict(X_test)
```

In [14]:
```
print(lm.intercept_)
print(lm.coef_)
```

```
[-80800.16615241]
[[  2.48021866e+03   4.67784045e+03  -8.10461495e+03   4.73533561e+03
    6.71082735e+03   1.95796521e+04   2.70678110e+01   5.15552947e+01
    1.82433181e+01   1.02494604e+04   3.03508401e+04   1.13475607e+04
    9.60800367e+03   4.86757724e+04   8.16761059e+03   2.80985179e+03
    8.25876691e+03   6.44835907e+03   6.45777141e+03   1.31873527e+04
    1.68446923e+04   1.12023329e+04   1.93418763e+04   7.97085844e+03
    1.03771375e+04   1.45504044e+04   7.20327516e+03   4.56537299e+04
    6.35914343e+03  -1.21848069e+04   1.52077060e+04   1.62297445e+04
    7.94061885e+03   1.42126280e+04   1.17086936e+04   3.21869565e+03
    8.14541435e+02   6.85317121e+02   1.81225543e+03  -3.44355992e+03
   -1.23035662e+04]]
```

In [15]:
```
import sklearn
```

In [16]:
```
#we will take out the R^2 value
sklearn.metrics.r2_score(Y_test, y_pred)
```

Out[16]:
```
0.35256801039469621
```

In [17]:
```
import statsmodels.api as sm
```

```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib\site-p
ackages\statsmodels\compat\pandas.py:56: FutureWarning: The pandas.core.datet
ools module is deprecated and will be removed in a future version. Please use
the pandas.tseries module instead.
  from pandas.core import datetools
```

In [18]:
```
X = encoded_x
Y = data.iloc[:, 1:2]
## fit a OLS model with intercept on TV and Radio
X = sm.add_constant(X)
est = sm.OLS(Y, X).fit()

est.summary()
```

Out[18]:

OLS Regression Results

Dep. Variable:	FamilyIncome	**R-squared:**	0.344
Model:	OLS	**Adj. R-squared:**	0.343
Method:	Least Squares	**F-statistic:**	291.0
Date:	Sat, 14 Apr 2018	**Prob (F-statistic):**	0.00
Time:	10:15:16	**Log-Likelihood:**	-2.8944e+05
No. Observations:	22745	**AIC:**	5.790e+05
Df Residuals:	22703	**BIC:**	5.793e+05
Df Model:	41		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	-7.314e+04	2.26e+04	-3.237	0.001	-1.17e+05	-2.89e+04
NumBedrooms	2610.5627	673.575	3.876	0.000	1290.309	3930.817
NumChildren	4137.2606	703.547	5.881	0.000	2758.261	5516.260
NumPeople	-7555.3019	650.785	-11.610	0.000	-8830.884	-6279.720
NumRooms	4643.8267	302.838	15.334	0.000	4050.244	5237.410
NumVehicles	6938.4598	678.174	10.231	0.000	5609.192	8267.727
NumWorkers	1.901e+04	784.745	24.228	0.000	1.75e+04	2.06e+04
HouseCosts	27.0085	0.592	45.643	0.000	25.849	28.168
ElectricBill	46.7250	5.670	8.241	0.000	35.612	57.838
Insurance	18.6187	0.680	27.384	0.000	17.286	19.951
FamilyType_Male Head	8882.8225	2804.646	3.167	0.002	3385.524	1.44e+04
FamilyType_Married	3.015e+04	1661.750	18.143	0.000	2.69e+04	3.34e+04
NumUnits_Single attached	1.093e+04	3669.308	2.979	0.003	3740.098	1.81e+04
NumUnits_Single detached	9592.0328	3278.715	2.926	0.003	3165.527	1.6e+04
OwnRent_Outright	5.398e+04	6790.797	7.948	0.000	4.07e+04	6.73e+04
OwnRent_Rented	7495.5002	2011.641	3.726	0.000	3552.545	1.14e+04
YearBuilt_1940-1949	-3670.2014	2.12e+04	-0.173	0.862	-4.52e+04	3.78e+04
YearBuilt_1950-1959	618.4751	2.11e+04	0.029	0.977	-4.08e+04	4.2e+04
YearBuilt_1960-1969	-124.0640	2.11e+04	-0.006	0.995	-4.16e+04	4.13e+04
YearBuilt_1970-1979	-306.6532	2.11e+04	-0.015	0.988	-4.18e+04	4.11e+04
YearBuilt_1980-1989	6833.9315	2.12e+04	0.323	0.747	-3.46e+04	4.83e+04

YearBuilt_1990-1999	8202.1665	2.12e+04	0.388	0.698	-3.33e+04	4.97e+04
YearBuilt_2000-2004	6246.4785	2.12e+04	0.294	0.769	-3.54e+04	4.79e+04
YearBuilt_2005	8598.6384	2.18e+04	0.395	0.693	-3.41e+04	5.13e+04
YearBuilt_2006	7281.1955	2.19e+04	0.332	0.740	-3.57e+04	5.03e+04
YearBuilt_2007	8208.4163	2.21e+04	0.372	0.710	-3.5e+04	5.14e+04
YearBuilt_2008	6459.8614	2.25e+04	0.288	0.774	-3.76e+04	5.05e+04
YearBuilt_2009	-1837.0954	2.29e+04	-0.080	0.936	-4.67e+04	4.3e+04
YearBuilt_2010	3.312e+04	2.41e+04	1.373	0.170	-1.41e+04	8.04e+04
YearBuilt_Before 1939	-551.3734	2.11e+04	-0.026	0.979	-4.19e+04	4.08e+04
FoodStamp_Yes	-1.185e+04	2266.129	-5.228	0.000	-1.63e+04	-7406.278
HeatingFuel_Electricity	1.016e+04	7037.505	1.444	0.149	-3632.872	2.4e+04
HeatingFuel_Gas	1.34e+04	6583.622	2.036	0.042	496.946	2.63e+04
HeatingFuel_None	3925.2730	1.82e+04	0.215	0.830	-3.18e+04	3.97e+04
HeatingFuel_Oil	9834.1494	6625.616	1.484	0.138	-3152.511	2.28e+04
HeatingFuel_Other	8165.9121	8826.806	0.925	0.355	-9135.231	2.55e+04
HeatingFuel_Solar	2076.5958	3.73e+04	0.056	0.956	-7.1e+04	7.52e+04
HeatingFuel_Wood	-2028.0935	6933.553	-0.293	0.770	-1.56e+04	1.16e+04
Language_English	3761.2304	3028.383	1.242	0.214	-2174.608	9697.069
Language_Other	817.7825	5642.628	0.145	0.885	-1.02e+04	1.19e+04
Language_Other European	-2152.2006	3422.223	-0.629	0.529	-8859.993	4555.592
Language_Spanish	-9620.3926	3490.135	-2.756	0.006	-1.65e+04	-2779.489

Omnibus:	16166.760	**Durbin-Watson:**	0.565
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	427635.583
Skew:	3.107	**Prob(JB):**	0.00
Kurtosis:	23.313	**Cond. No.**	3.35e+05

In Model 3, we are going to use backward elimination to delete p-values less than 0.05.