In [1]:
```python
#Melbourne House Price Data from https://www.kaggle.com/anthonypino/melbourne-housing-market
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as stats
import seaborn as sns
from matplotlib import rcParams
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeRegressor

%matplotlib inline
%pylab inline
houses = pd.read_csv('/Users/colinbrant/Downloads/MELBOURNE_HOUSE_PRICES_LESS.csv')
```

```
Populating the interactive namespace from numpy and matplotlib
```

In [2]:
```python
#First get an idea for what the dataset looks like
houses.head()
```

Out[2]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Postcode	Regionname	Propertycount	Distai
0	Abbotsford	49 Lithgow St	3	h	1490000.0	S	Jellis	1/04/2017	3067	Northern Metropolitan	4019	
1	Abbotsford	59A Turner St	3	h	1220000.0	S	Marshall	1/04/2017	3067	Northern Metropolitan	4019	
2	Abbotsford	119B Yarra St	3	h	1420000.0	S	Nelson	1/04/2017	3067	Northern Metropolitan	4019	
3	Aberfeldie	68 Vida St	3	h	1515000.0	S	Barry	1/04/2017	3040	Western Metropolitan	1543	
4	Airport West	92 Clydesdale Rd	2	h	670000.0	S	Nelson	1/04/2017	3042	Western Metropolitan	3464	1

In [3]:
```python
#Use columns to find all the column names
houses.columns
```

Out[3]:
```
Index(['Suburb', 'Address', 'Rooms', 'Type', 'Price', 'Method', 'SellerG',
       'Date', 'Postcode', 'Regionname', 'Propertycount', 'Distance',
       'CouncilArea'],
      dtype='object')
```

In [4]:
```python
#Next get the types values of each column
houses.dtypes
```

Out[4]:
```
Suburb            object
Address           object
Rooms              int64
Type              object
Price            float64
Method            object
SellerG           object
Date              object
Postcode           int64
Regionname        object
Propertycount      int64
Distance         float64
CouncilArea       object
dtype: object
```

In [5]:
```python
#Next get some info on the dataset
houses.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 63023 entries, 0 to 63022
Data columns (total 13 columns):
Suburb           63023 non-null object
Address          63023 non-null object
Rooms            63023 non-null int64
Type             63023 non-null object
Price            48433 non-null float64
Method           63023 non-null object
SellerG          63023 non-null object
Date             63023 non-null object
Postcode         63023 non-null int64
Regionname       63023 non-null object
Propertycount    63023 non-null int64
Distance         63023 non-null float64
CouncilArea      63023 non-null object
dtypes: float64(2), int64(3), object(8)
memory usage: 6.3+ MB
```

In [6]:
```python
#There are some columns that won't be helpful for this analysis so we can drop them
houses = houses.drop(columns='Distance')
houses = houses.drop(columns = 'CouncilArea')
houses.head()
```

Out[6]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Postcode	Regionname	Propertycount
0	Abbotsford	49 Lithgow St	3	h	1490000.0	S	Jellis	1/04/2017	3067	Northern Metropolitan	4019
1	Abbotsford	59A Turner St	3	h	1220000.0	S	Marshall	1/04/2017	3067	Northern Metropolitan	4019
2	Abbotsford	119B Yarra St	3	h	1420000.0	S	Nelson	1/04/2017	3067	Northern Metropolitan	4019
3	Aberfeldie	68 Vida St	3	h	1515000.0	S	Barry	1/04/2017	3040	Western Metropolitan	1543
4	Airport West	92 Clydesdale Rd	2	h	670000.0	S	Nelson	1/04/2017	3042	Western Metropolitan	3464

In [7]:
```python
#No we can check for null and duplicate values in the data
houses = houses.dropna()
houses = houses.drop_duplicates()
```

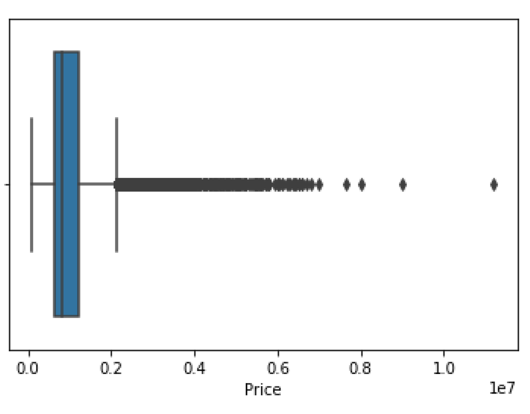
In [8]:
```python
#To get an idea of the spread of values use the describe method
houses.describe()
#Mean room number is 3.07
#Mean house price is 997,000 with a min of 85,000 and a max of 11,200,000
```

Out[8]:

	Rooms	Price	Postcode	Propertycount
count	48432.000000	4.843200e+04	48432.000000	48432.000000
mean	3.071688	9.978980e+05	3123.211472	7566.427218
std	0.944705	5.935050e+05	125.535986	4457.447851
min	1.000000	8.500000e+04	3000.000000	39.000000
25%	2.000000	6.200000e+05	3051.000000	4280.000000
50%	3.000000	8.300000e+05	3103.000000	6567.000000
75%	4.000000	1.220000e+06	3163.000000	10412.000000
max	31.000000	1.120000e+07	3980.000000	21650.000000

In [9]:
```python
#Next step is to eliminate price outliers
#First visualize using a boxplot
sns.boxplot(houses['Price'])
```

Out[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe039be9550>
```



In [10]:
```python
#Next use z score for a more precise calculation
z_score = np.abs(stats.zscore(houses['Price']))
houses = houses[(z_score < 3)]
houses.head()
#Now Price outliers with a z score greater than 3 have been removed
```

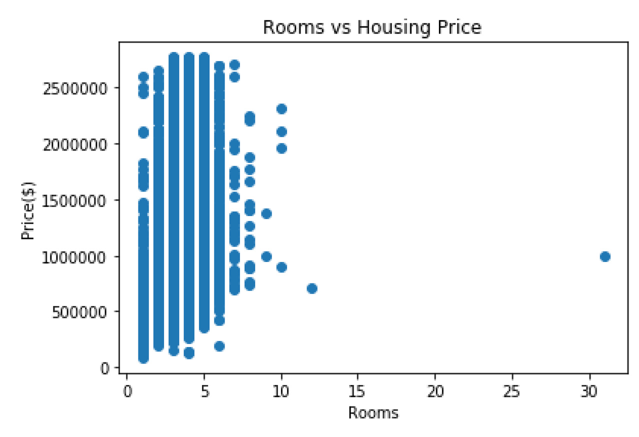
Out[10]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Postcode	Regionname	Propertycount
0	Abbotsford	49 Lithgow St	3	h	1490000.0	S	Jellis	1/04/2017	3067	Northern Metropolitan	4019
1	Abbotsford	59A Turner St	3	h	1220000.0	S	Marshall	1/04/2017	3067	Northern Metropolitan	4019
2	Abbotsford	119B Yarra St	3	h	1420000.0	S	Nelson	1/04/2017	3067	Northern Metropolitan	4019
3	Aberfeldie	68 Vida St	3	h	1515000.0	S	Barry	1/04/2017	3040	Western Metropolitan	1543
4	Airport West	92 Clydesdale Rd	2	h	670000.0	S	Nelson	1/04/2017	3042	Western Metropolitan	3464

In [11]:
```python
#Next we will want to build a cluster model based on rooms
plt.scatter(houses.Rooms,houses.Price)
plt.title('Rooms vs Housing Price')
plt.xlabel('Rooms')
plt.ylabel('Price($)')

#From the scatterplot we can see there is liekly a correlation however there are too many da
ta points
#to be analyzed so the scatterplot isn't very helpful
```

Out[11]:
```
Text(0, 0.5, 'Price($)')
```



In [12]:
```python
#So instead we will analyze the data using linear regression
#We will find out how the room variable affects price

X = houses.Rooms.values.reshape(-1,1)
y = houses.Price.values.reshape(-1,1)

train_X, test_X, train_y, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

#Now that we have split up the data we need to train the algorithm
regress_model = LinearRegression()
regress_model.fit(train_X, train_y)
#Print the intercept and slope calculated from the model
print(regress_model.intercept_)

print(regress_model.coef_)
```

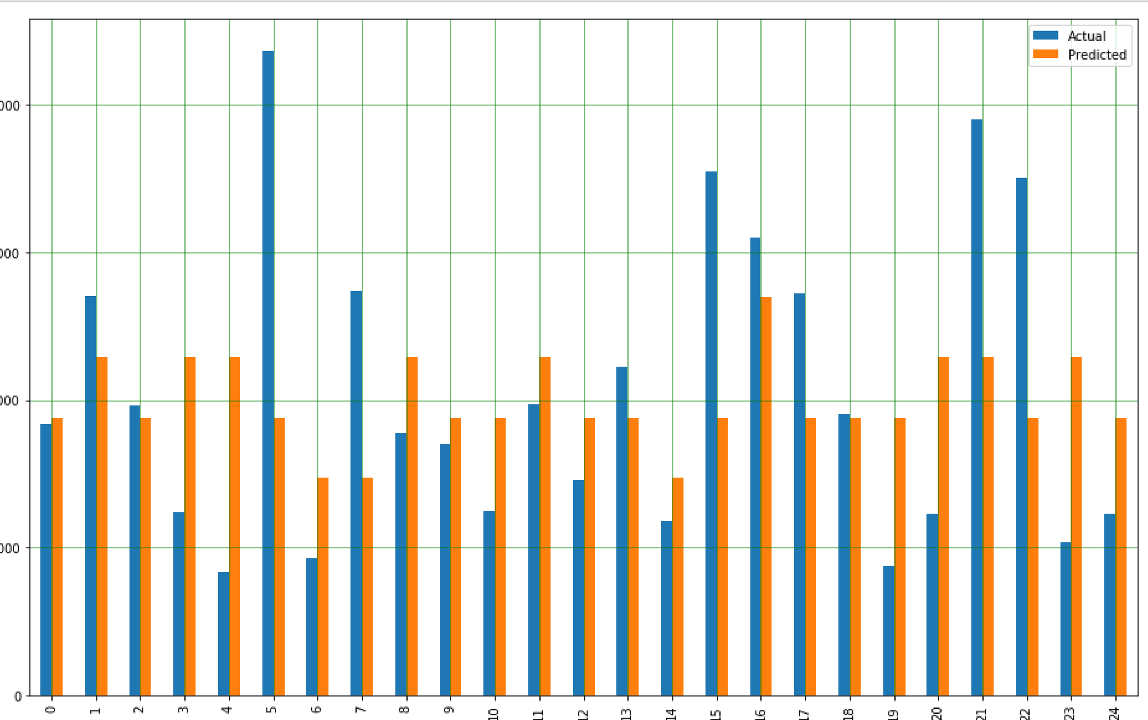
```
[326327.47136028]
[[204498.99604801]]
```

In [13]:
```python
#Runs the prediction in the model and makes a comparison between predicted and actual values
predict = regress_model.predict(test_X)
results = pd.DataFrame({'Actual': y_test.flatten(), 'Predicted': predict.flatten()})
print(results)
```

```
         Actual     Predicted
0      920000.0  9.398245e+05
1     1350000.0  1.144323e+06
2      980000.0  9.398245e+05
3      620000.0  1.144323e+06
4      417000.0  1.144323e+06
5     2180000.0  9.398245e+05
6      465000.0  7.353255e+05
7     1370000.0  7.353255e+05
8      890000.0  1.144323e+06
9      850000.0  9.398245e+05
10     625000.0  9.398245e+05
11     985000.0  1.144323e+06
12     730000.0  9.398245e+05
13    1110000.0  9.398245e+05
14     591000.0  7.353255e+05
15    1775000.0  9.398245e+05
16    1550000.0  1.348822e+06
17    1360000.0  9.398245e+05
18     950000.0  9.398245e+05
19     436500.0  9.398245e+05
20     615000.0  1.144323e+06
21    1950000.0  1.144323e+06
22    1753000.0  9.398245e+05
23     519000.0  1.144323e+06
24     615000.0  9.398245e+05
25     700000.0  9.398245e+05
26     376000.0  5.308265e+05
27    1210000.0  9.398245e+05
28     755000.0  9.398245e+05
29    1165000.0  9.398245e+05
...         ...           ...
9483  1250000.0  7.353255e+05
9484   517500.0  1.144323e+06
9485  1550000.0  9.398245e+05
9486   815000.0  1.144323e+06
9487   850000.0  9.398245e+05
9488  1100000.0  1.348822e+06
9489   900000.0  9.398245e+05
9490  1275000.0  9.398245e+05
9491   505000.0  9.398245e+05
9492  2560000.0  1.144323e+06
9493   870000.0  9.398245e+05
9494   649000.0  7.353255e+05
9495   610000.0  1.144323e+06
9496   410000.0  9.398245e+05
9497   770000.0  9.398245e+05
9498   630000.0  9.398245e+05
9499   525000.0  1.144323e+06
9500   797000.0  9.398245e+05
9501   520000.0  9.398245e+05
9502   572000.0  7.353255e+05
9503   550000.0  7.353255e+05
9504  1010000.0  1.144323e+06
9505   748000.0  7.353255e+05
9506  1155000.0  9.398245e+05
9507   430000.0  7.353255e+05
9508   630000.0  9.398245e+05
9509   360000.0  5.308265e+05
9510   500000.0  5.308265e+05
9511   545000.0  9.398245e+05
9512   380000.0  7.353255e+05

[9513 rows x 2 columns]
```

In [14]:
```python
#Graphs the first 25 predicted and actual values as bars
compare = results.head(25)
compare.plot(kind='bar',figsize=(16,10))
plt.grid(which='major', linestyle='-', linewidth='0.5', color='green')
plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')
plt.show()
```



In [15]:
```python
#Now evaluate the the performance of the algorithm
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, predict))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, predict))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, predict)))
#In this case our model was not too accurate so lets try a different model
```

```
Mean Absolute Error: 325993.31878437113
Mean Squared Error: 183231565774.59683
Root Mean Squared Error: 428055.5638869758
```

In [16]:

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-16-295dcb61c7e9> in <module>
      1 #Next instead of a sklearn we will use Prophet to make future predictions
----> 2 pro = Prophet()
      3 pro.fit(houses)

NameError: name 'Prophet' is not defined
```