CAPSTONE PROJECT 1: FINAL REPORT

HOUSE PRICE PREDICTIONS

10/09/2019

Mehmet KETENCI
KANSAS CITY, MO

INTRODUCTION:

In this project, our aim is to predict house prices in certain places of Washington area to give very clear perspective to both sellers and buyers.Prediction of property prices is becoming increasingly important and beneficial. Property prices are a good indicator of both the overall market conditions and the economic health of a country.

STAKEHOLDERS: It is important for real estate markets can categorize house depend on the prices of customer budget to match right customers to required homes while buyers to find home price range that they are looking for.

DATA ACQUISITION:

I acquired the data from Kaggle.(https://www.kaggle.com/shree1992/housedata)

STEPS FOR SOLUTION

1) **Data Wrangling:**
   - Gaining insight about data and taking action according to data itself with Python and Pandas.
   - Checking the data to clean with preprocessing and converting raw data to desired format to be used for future processes.
   - Trying to see any outliers that don't affect data pretty much.

2) **Exploratory Data Analysis(EDA):**
   - Focusing on especially remarkable data that will be detected by Pandas.
   - Deeping into behind to reasons by visualizations.
   - Getting benefit of visualizations to reach the goal using Seaborn and Matplotlib libraries.

3) **Statistical Approach:**
   - Featuring especially things will be searched in previous steps (Data Wrangling and EDA) using Z-T table with OLS chart
   - Working on p-values of estimations in main and support branch of this capstone.

4) **Machine Learning:**
   - Finalize 'House Price Prediction' with Machine Learning algorithms by supporting the codes that will be produced in statistical part.
   - Using Supervised Learning Models to predict house prices in this chapter since the target data is a numerical column**.**

If we summarize, actions will be taken as follows:
examining data, cleaning if necessary,making visualizations to gain insight the data,applying some statistical tests,building model to predict prices and developing our model to get better results.

DATA WRANGLING:

DATA: House data has 4600 rows with 18 different columns.Set_index,groupby, concatenate, pivot_table are some of pandas techniques I applied in Data Wrangling parts.

LOADING DATA: Lets load the data and try to see general view of it

```
house = pd.read_csv('data.csv')
house.head()
```

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above
0	2014-05-02 00:00:00	313000.0	3.0	1.50	1340	7912	1.5	0	0	3	1340
1	2014-05-02 00:00:00	2384000.0	5.0	2.50	3650	9050	2.0	0	4	5	3370
2	2014-05-02 00:00:00	342000.0	3.0	2.00	1930	11947	1.0	0	0	4	1930
3	2014-05-02 00:00:00	420000.0	3.0	2.25	2000	8030	1.0	0	0	4	1000
4	2014-05-02 00:00:00	550000.0	4.0	2.50	1940	10500	1.0	0	0	4	1140

sqft_basement	yr_built	yr_renovated	street	city	statezip	country
0	1955	2005	18810 Densmore Ave N	Shoreline	WA 98133	USA
280	1921	0	709 W Blaine St	Seattle	WA 98119	USA
0	1966	0	26206-26214 143rd Ave SE	Kent	WA 98042	USA
1000	1963	0	857 170th Pl NE	Bellevue	WA 98008	USA
800	1976	1992	9105 170th Ave NE	Redmond	WA 98052	USA

GAINING INSIGHT THE DATA:

We can determine data types to decide which features we can be using for the next processes.Data doesn't have any missing value according to given and it is not required to be cleaned.

```
house.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4600 entries, 0 to 4599
Data columns (total 18 columns):
date             4600 non-null object
price            4600 non-null float64
bedrooms         4600 non-null float64
bathrooms        4600 non-null float64
sqft_living      4600 non-null int64
sqft_lot         4600 non-null int64
floors           4600 non-null float64
waterfront       4600 non-null int64
view             4600 non-null int64
condition        4600 non-null int64
sqft_above       4600 non-null int64
sqft_basement    4600 non-null int64
yr_built         4600 non-null int64
yr_renovated     4600 non-null int64
street           4600 non-null object
city             4600 non-null object
statezip         4600 non-null object
country          4600 non-null object
dtypes: float64(4), int64(9), object(5)
memory usage: 647.0+ KB
```

Now let's try to see how other columns affect price. Sqft living, sqft above, bathroom columns are prominent features that highly correlation with price.

```
house.corr().price
```

```
price            1.000000
bedrooms         0.200336
bathrooms        0.327110
sqft_living      0.430410
sqft_lot         0.050451
floors           0.151461
waterfront       0.135648
view             0.228504
condition        0.034915
sqft_above       0.367570
sqft_basement    0.210427
yr_built         0.021857
yr_renovated    -0.028774
Name: price, dtype: float64
```

OUTLIER:

Are there homes with $0 prices?

```
len(house[house.price==0])
```

```
49
```

There are 49 homes with $0 that we can drop these columns that doesn't affect our dataset with dropna method.Our data has 4600-49=4551 features.
Can we see the impact of number of total room on price?

house['total_room'] = house['bedrooms']+house['bathrooms']

house.head()

price	bedrooms	bathrooms	total_room	sqft_per_room
313000.0	3.0	1.50	4.50	297.777778
2384000.0	5.0	2.50	7.50	486.666667
342000.0	3.0	2.00	5.00	386.000000
420000.0	3.0	2.25	5.25	380.952381
550000.0	4.0	2.50	6.50	298.461538

```
h.corr().price
```

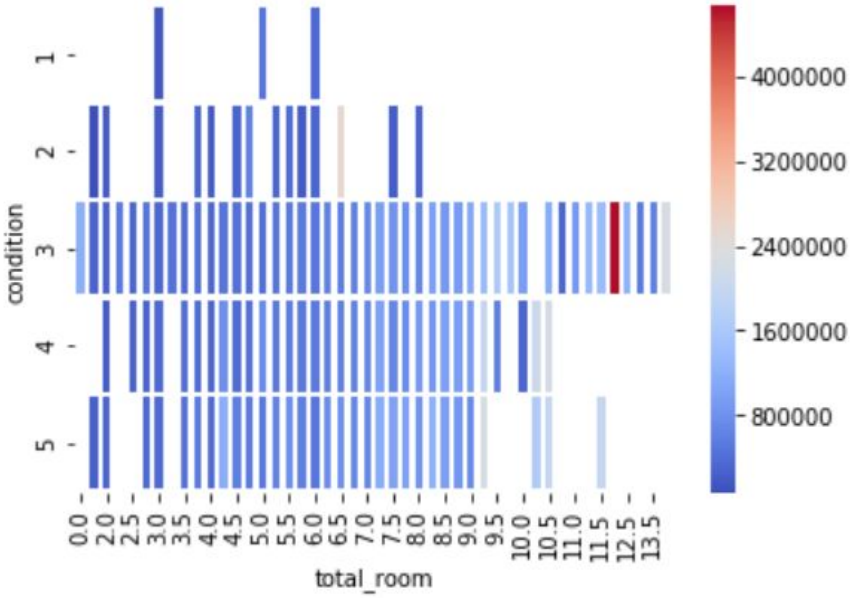
```
price            1.000000
bedrooms         0.210228
bathrooms        0.341126
sqft_living      0.445494
sqft_lot         0.051347
floors           0.152758
waterfront       0.150083
view             0.242587
condition        0.038892
sqft_above       0.380661
sqft_basement    0.217782
yr_built         0.021757
yr_renovated    -0.029034
total_room       0.307452
Name: price, dtype: float64
```

In this graph below, we can see the triple relationships among price-#total room-condition.There is a home with dark orange color which condition is 3 and its price is pretty bigger than average.This

home price needs to be reviewed in detail. We see the same issue for the home with condition is 2.It is also way above the average and required what the reason behind is.

```
df = h.pivot_table(index='condition',columns='total_room',values='price')
sns.heatmap(df,cmap='coolwarm',linewidths=1.5)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x22f03764dd8>
```

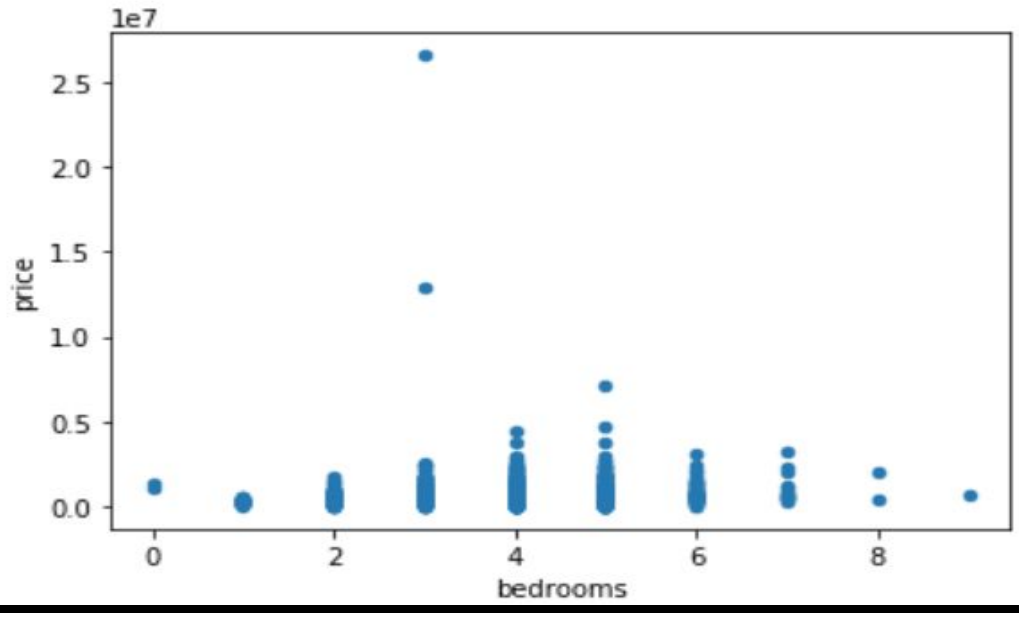


EXPLORATORY DATA ANALYSIS (EDA):

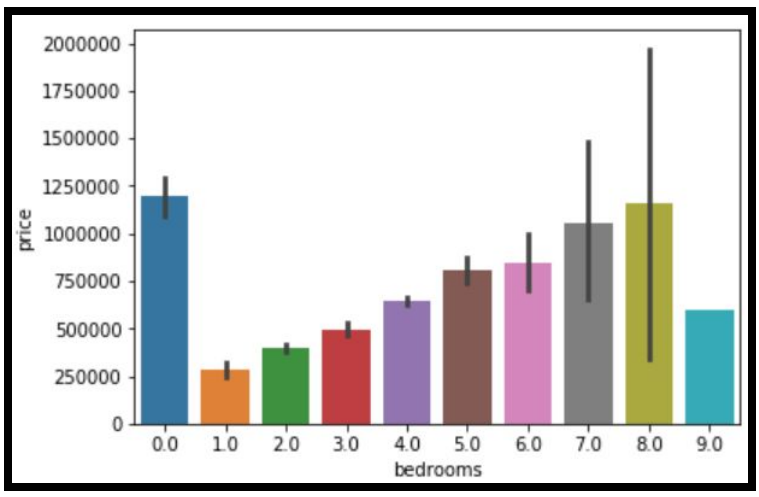
GRAPHS AND TYPES:(Histogram,Scatter,Bar,Heatmap)

We can easily have a chance to see home prices with outliers scattering depend on bedroom numbers.

```
house.plot(kind='scatter',x='bedrooms',y='price')
plt.show()
```

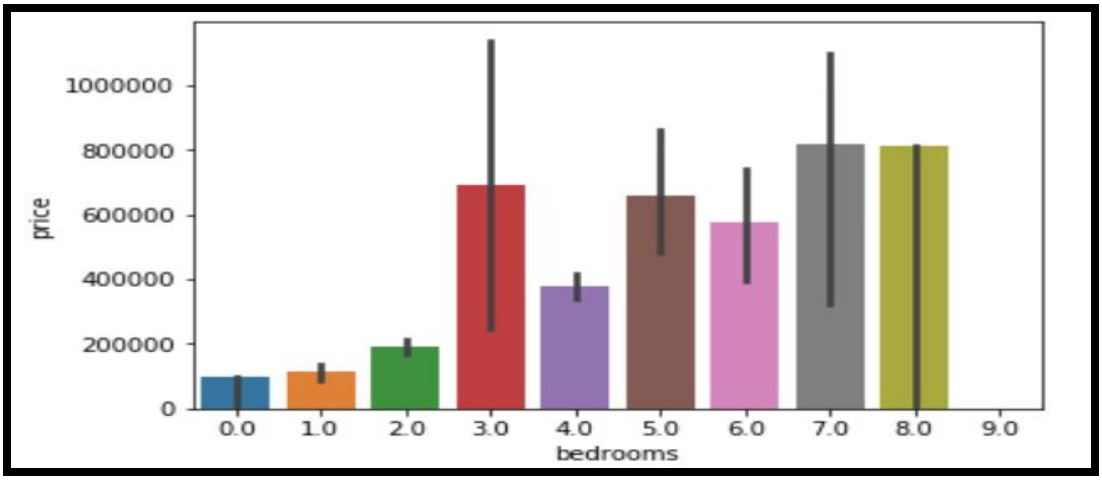


Graph below is another graph type,barplot shows us relationship between bedroom and price. It can be commented as price increases as the number of bedroom increases.



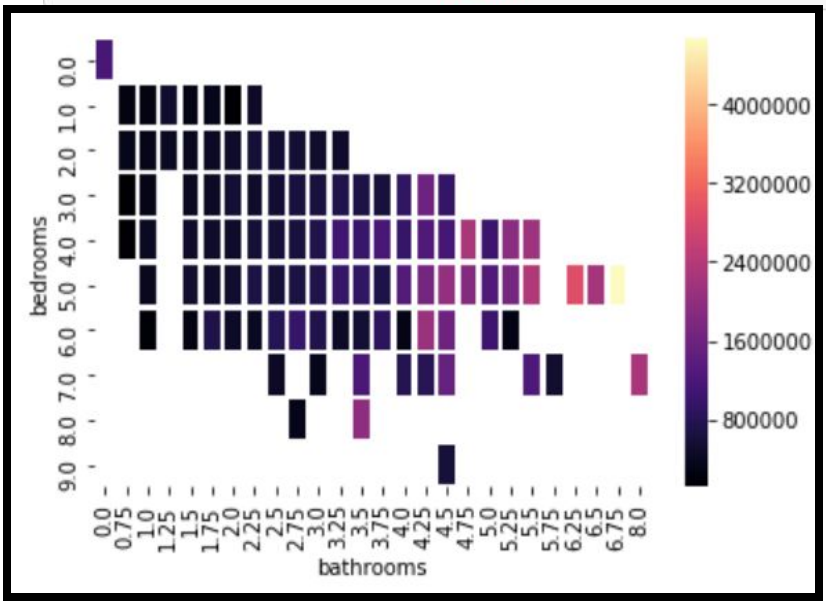
If we desire to see the same features on houses above standards.

```
h=new_house
h[h.price >= new_house['price'].mean()]
h
sns.barplot(x='bedrooms',y='price',data=h,estimator = np.std)
```



Let's use heatmap and pivot_table to get an idea of effecting price both bedroom and bathroom numbers.

```
df = h.pivot_table(index='bedrooms',columns='bathrooms',values='price')
sns.heatmap(df,cmap='magma',linewidths=1.5)
```



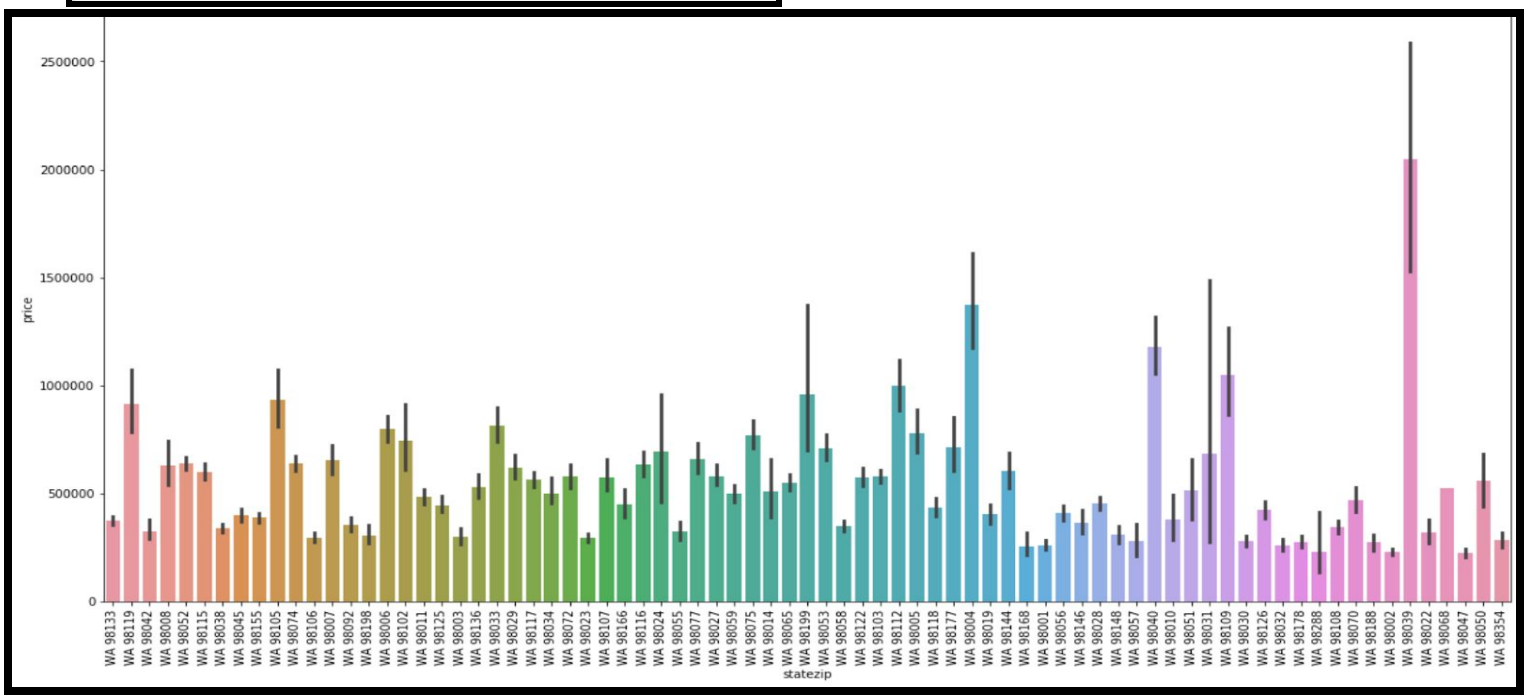
In this study, I looked for price range depend on the size of per room.

```
h['sqft_per_room'] = (h['sqft_above']+h['sqft_basement'])/(h['bedrooms']+h['bathrooms'])
h.head()
```

price	bedrooms	bathrooms	total_room	sqft_per_room
313000.0	3.0	1.50	4.50	297.777778
2384000.0	5.0	2.50	7.50	486.666667
342000.0	3.0	2.00	5.00	386.000000
420000.0	3.0	2.25	5.25	380.952381
550000.0	4.0	2.50	6.50	298.461538

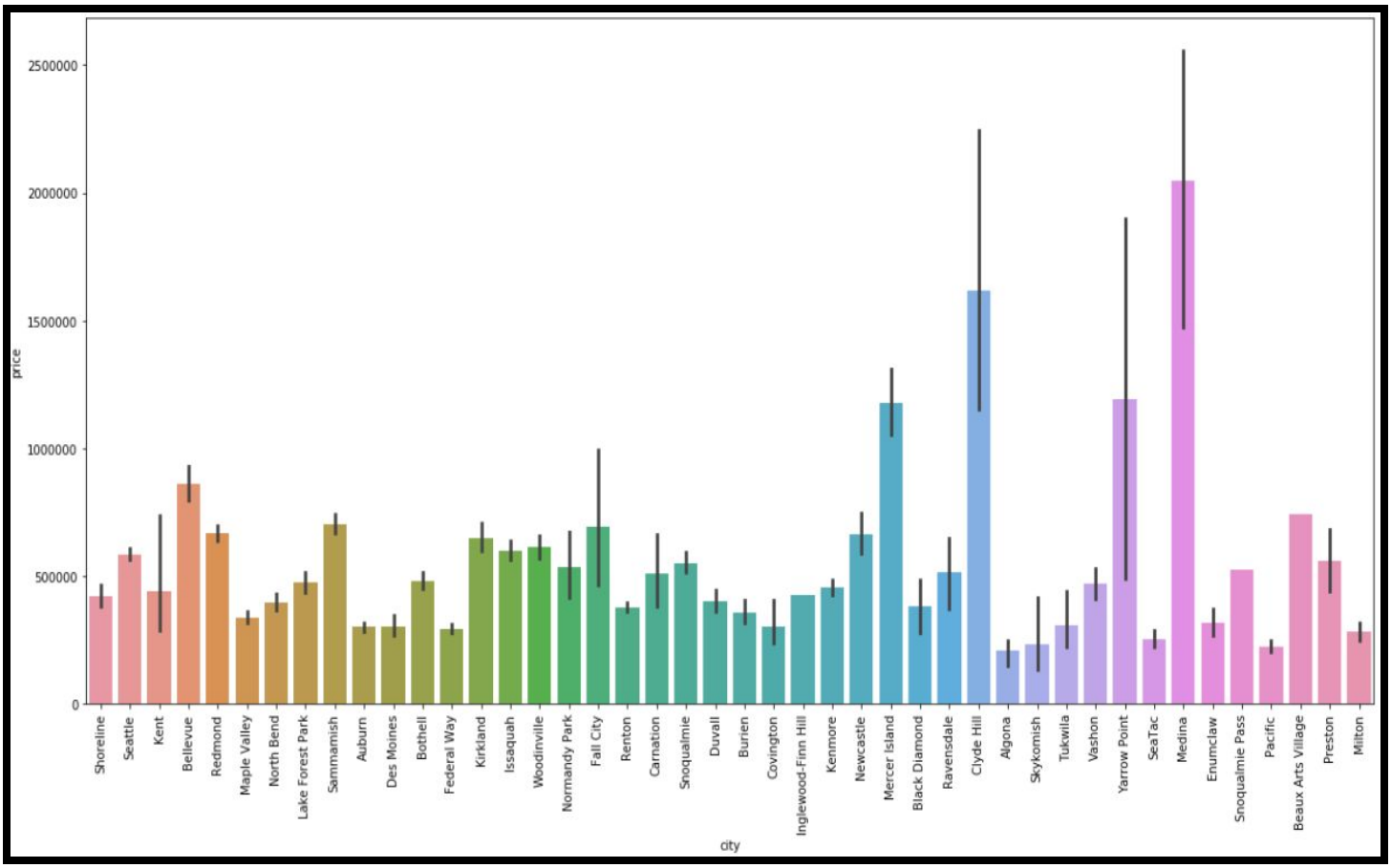
Here are prices with certain cities and zip codes. This might give an idea about the price distribution.This a core resources especially for real estate agents.

```
plt.figure(figsize=(20,10))
sns.barplot(x='city',y='price',data = h)
plt.xticks(rotation = 'vertical')
plt.show()
```



EDA RESULTS:

Some homes with 0 rooms might be a studio or office in the center of the city. There are some distinct zip codes that have higher prices, as well as the opposite situation exists may depend on safety,crime rates or closing to important places. Good conditions,perfect views also contribute positive side to the price. Number of bedroom and bathroom will huge impact on prices. People in interestingly consider bathroom number. Prices with under marketing should be either foreclosure homes,short sales,fixer uppers, homes in bad locations,homes priced too low by mistakes.



STATISTICAL APPROACH:

Statistical approach is a powerful way at catching significant data and helping machine learning algorithms to build an effective model.

In this chapter, I will continue my capstone project about House predictions approaching statistically way.Here are my three different observations:

1) Comparing average price of homes with 2 and 3 bedrooms.
2) Determining average price in Issaquah greater than average price in Kirkland or not
3) Probability of a home hired by a family just moves into this area.

Inquiry:

A mortgage company wants to test if there is a significant difference between the mean price of homes with 3 bedrooms and the mean price of homes with 2 bedrooms in given dataset.

(significance level=0.05)

Let M2 and M3 be mean of homes that have 3 bedrooms and 2 bedrooms respectively.
H0: M2 = M3 (M2 - M3 = 0) and H1: M2 ≠ M3 (M2 - M3 ≠ 0)

EXAMINING HOUSES WITH 3 AND 2 BEDROOMS:

```
len(house[house['bedrooms']==3])
```

```
2032
```

```
h3 = house[house['bedrooms']==3]
house_3 = h3['price']
```

```
round(np.mean(house_3),2)
```

```
488613.02
```

```
round(np.std(house_3),2)
```

```
690338.86
```

```
len(house[house['bedrooms']==2])
```

```
566
```

```
h2 = house[house['bedrooms']==2]
house_2 = h2['price']
```

```
round(np.mean(house_2),2)
```

```
391621.92
```

```
round(np.std(house_2),2)
```

```
194947.17
```

let's take a sample from both house type that contains 50 houses from M2 and M3 respectively.(50<(566/10)),(50<(2032/10)) For the sake of 10% rule being independent, 50 is a plausible value which is less than 10% of the number of houses that have 2 and 3 bedrooms.

In this frequentis statistic,z value is way beyond z_critical value so we have to reject the null hypothesis(M3-M2=0) which we accept the alternative hypothesis. It indicates that there is a significant difference between mean price of homes with 2 bedrooms and mean price of homes with 3 bedrooms.

```
# I am finding mean and standard deviation of the sample respectively for houses with 2 bedrooms.
```

```
seed(35)
sample_2 = np.random.choice(house_2,50)
sample_mean_2 = np.mean(sample_2)
round(sample_mean_2,2)
```

```
425348.56
```

```
seed(35)
sample_std_2 = np.std(house_2)/(50**0.5)
round(sample_std_2,2)
```

```
27569.69
```

```
#I will find mean and standard deviation of the sample belongs to houses with 3 bedrooms.
# For the sake of 10% independent rule, I will choose 50 houses.(50<(2032/10))
```

```
seed(35)
sample_3 = np.random.choice(house_3,50)
sample_mean_3 = np.mean(sample_3)
round(sample_mean_3,2)
```

```
539394.89
```

```
seed(35)
sample_std_3 = np.std(house_3)/(50**0.5)
round(sample_std_3,2)
```

```
97628.66
```

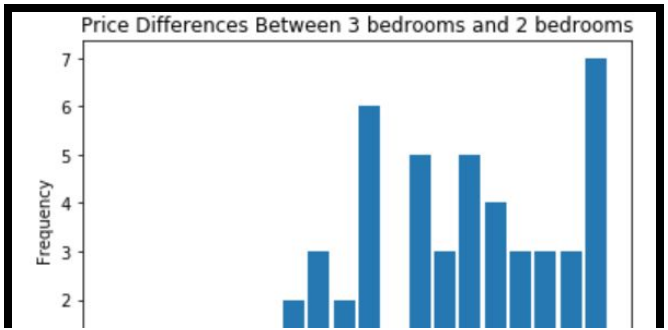
Now let`s apply z test

```
z = (sample_mean_3-sample_mean_2)/np.sqrt(np.sum((sample_std_2**2/50**0.5)+(sample_std_3**2/50**0
.5)))
round(z,2)
```

```
2.99
```

```
#significance level = 0.05
z_critical = norm.ppf(0.975)
round(z_critical,2)
```

```
1.96
```



2) Is average price in Issaquah greater than average price in Kirkland?

```
#Here is the list of cities in Washington
city = house['city']
city_num = city.value_counts()
city_num.head(7)
```

```
Seattle     1573
Renton       293
Bellevue     286
Redmond      235
Kirkland     187
Issaquah     187
Kent         185
Name: city, dtype: int64
```

We will inquiry house prices in two same size cities, Kirkland and Issaquah(both city has 187 homes) if the average price of homes in Issaquah greater than average price of homes in Kirkland. If I investigate having normal distribution conditions:

1) 187 homes were randomly chosen homes from both cities separately.

2) 187 homes for Kirkland is below 10% of total homes.(Independent) (Apparently number of homes in Kirkland is near 22000) same thing valid for Issaquah.187 homes is quite under the total number of homes(10000) in it. 3) Our sample is reasonably large 187>30(for both samples)

I ``d like to share steps to be taken for reaching a correct estimation. First, I will set up the Null and Alternative hypotheses, Let Misq,Mki be mean of sample means

H0: Misq≤Mki ↔ Misq-Mki≤0 H1: Misq>Mki ↔ Misq-Mki>0 Significance level: α=0.05

And then I will apply z formula: z = (Misq-Mki)/(np.sqrt(np.sum((Sisq**2)/(nisq**2)+(Ski**2)/(nki**2)))

```
Kirkland = house[house.city == 'Kirkland']
round(np.mean(Kirkland['price']),2)
```

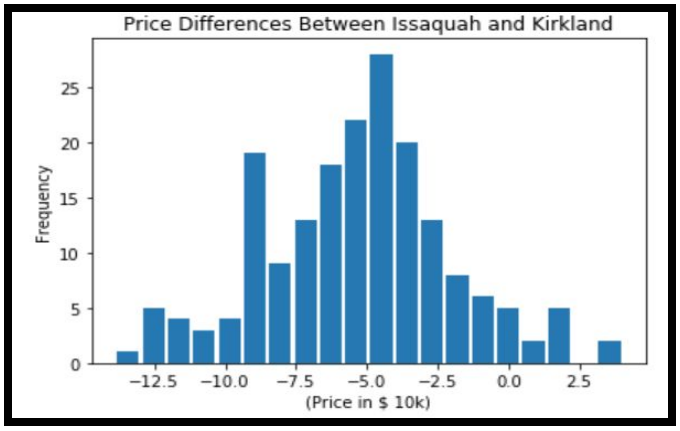
```
round(np.std(Kirkland['price']),2)
```

```
Issaquah = house[house.city == 'Issaquah']
round(np.mean(Issaquah['price']),2)
```

```
round(np.std(Issaquah['price']),2)
```

```
#Let`s start finding with z in order to see whether it falls into area within 95% interval or not.
z = (596163.75-651583.59)/np.sqrt(np.sum((245008.93**2/187)+(368857.02**2/187)))
round(z,2)
```

Now it is time to check falling this z value into within 95% confidence interval or not. For this reason, we need to figure our p value out.(significance level = 0.05) P value of -1.71 on z table is 0.0436 which is smaller than the significance level(0.05). We will reject the Null Hypothesis in favor of alternative hypothesis.It means average price of Homes in Issaquah is not expensive than Kirkland.



In other words, home prices in Kirkland is more expensive than home prices in Issaquah.

As you can see on the graph,major data gathered below 0 which mean of home prices in Issaquah is lower than mean of home prices in Kirkland.

3) BAYES THEOREM

Wederson Family moves from California to Washington. They search for a home that has 2 bathrooms and 3 or 4 bedrooms in new area.

What is the probability of they rent a home with condition is not lower than 4 (with Ms. Wederson`s demand) of given options above?(Solve by using Bayes Theorem)

X: probability of homes with 3 or 4 bedrooms and 2 bathrooms Y: probability of homes with not 3 or 4 bedrooms and not 2 bathrooms A: probability of condition greater than or equal to 4 B: probability of condition less than 4 and Mr. Wederson is searching among the homes that have 3 or 4

bedrooms and 2 bathrooms.(X) Of those homes,with his wife `s request, he will try to find out a home condition is greater than or equal to 4.(A)

Statistically,he is seeking P(A/X)

P(A|X)=(P(X|A)*P(A))/P(X) (BAYES T.)

Question: How many homes with 3 or 4 bedrooms and 2 bathrooms are there?

```
X = house[((house.bedrooms==3)|(house.bedrooms==4))&(house.bathrooms==2)]

print('P(X):{}{}{}'.format(len(X),'/',4600))
```

```
P(X):355/4600
```

Question: How many home conditions does have greater than or equal to 4?

```
A = house[house.condition>=4]

print('P(A):{}{}{}'.format(len(A ),'/',4600))
```

```
P(A):1687/4600
```

```
#Let`s find out of up_4 how many homes are there with 3 or 4 bedrooms and 2 bathrooms.
#P(X|A)=?
X_A = A[((A.bedrooms==3)|(A.bedrooms==4))&(A.bathrooms==2)]
print('P(X|A):{}{}{}'.format(len(X_A),'/',1687))
```

```
P(X|A):177/1687
```

Now, I am plug in values into Bayes formula

P(A|X)=(P(X|A)*P(A))/P(X) (BAYES T.)

```
(((177/1687)*(1687/4600))/(355/4600))*100
```

```
49.859154929577464
```

RESULT:

Probability of Wederson Family choosing homes they wish is 49.9%.

It seems they are lucky 1 out of 2 homes is the home they are looking for and they can easily access or find those homes by filtering method in any home selling websites.

MACHINE LEARNING ALGORITHMS:

Data Science is the process of making some assumptions and hypotheses on the data, and testing them by performing some tasks. Initially we could make the following intuitive assumptions for each feature. We should start with creating target (price) and feature columns(others) and split data into training and testing data.

```
# Creating features and target
X = house.iloc[:,2:14]
y = house.iloc[:,1].values
```

```
#Splitting dataset as test and train to build a model
from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=38)
```

We then set a baseline model which is linear regression. It is an algorithm that will be compared to other machine learning algorithms to obtain better outcomes.

We started with very low R2 score 0.24 then decide to try again linear regression with scaled data.

Here is the improved result below:

LINEAR REGRESSION

```
#Normalizing features

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
X_scaled = sc.fit_transform(X_train)
y_scaled = y_train
X_tested = sc.transform(X_test)
y_tested = y_test
```

```
#Building a baseline model - Linear Regression

from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_scaled,y_scaled)
y_pred = lr.predict(X_tested)
```

```
r2_score(y_tested,y_pred)
```

```
0.5473440758579227
```

We will continue to improve our models with different algorithms such as Support Vector Machine.

SUPPORT VECTOR MACHINE (SVM)

```
from sklearn.preprocessing import StandardScaler

sc1 = StandardScaler()
X_scaled = sc1.fit_transform(X_train)
y_scaled = y_train
X_tested = sc1.transform(X_test)
y_tested = y_test
```

```
#I tried kernels and choosed best was 'linear'

from sklearn.svm import SVR
svr_reg = SVR(kernel = 'linear')
svr_reg.fit(X_scaled,y_scaled)
```

```
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1,
    gamma='auto_deprecated', kernel='linear', max_iter=-1, shrinking=True,
    tol=0.001, verbose=False)
```

```
y_pred = svr_reg.predict(X_tested)
```

```
r2_score(y_tested,y_pred)
```

```
-0.013970282219468277
```

Support Vector Machine is an algorithm that we can skip at this point.

Random Forest is another supervised learning algorithm that we can perform to improve our model.

RANDOM FOREST

```
#Question: What is the optimum estimator and how do we decide n estimator?

from sklearn.ensemble import RandomForestRegressor

rf = RandomForestRegressor(n_estimators=50,random_state=38)
rf.fit(X_train,y_train)
```

```
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=50,
                      n_jobs=None, oob_score=False, random_state=38, verbose=0,
                      warm_start=False)
```

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

```
r2_score(y_test,y_pred)
```

```
0.4559014952594673
```

```
# Let`s change n_estimators as 500
rf1 = RandomForestRegressor(n_estimators=500,random_state=38)
rf1.fit(X_train,y_train)
```

```
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=500,
                      n_jobs=None, oob_score=False, random_state=38, verbose=0,
                      warm_start=False)
```

```
y_pred = rf1.predict(X_test)
```

```
r2_score(y_test,y_pred)
```

```
0.4665887869826857
```

Random forest gave better results with 500 estimators but it still did not satisfy the need.

There is another ML algorithm that gives much better scores in general Kaggle competitions which is Gradient Boosting.

GRADIENT BOOSTING

```
from sklearn.ensemble import GradientBoostingRegressor
gbr = GradientBoostingRegressor(random_state=38)
gbr.fit(X_train,y_train)
y_pred = gbr.predict(X_test)
```

```
r2_score(y_test,y_pred)
```

```
0.5019632251969264
```

From what we applied with ML so far, we decided to make feature selection with hyperparameter tuning with GridSearch.

UNIVARIATE SELECTION

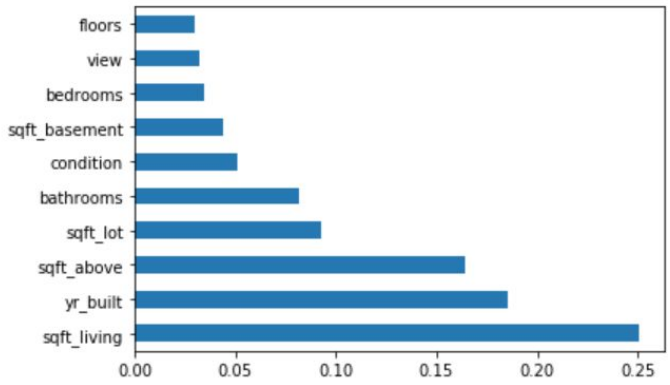
```
from sklearn.ensemble import ExtraTreesRegressor
model = ExtraTreesRegressor(n_estimators=100)
model.fit(X,y)
```

```
ExtraTreesRegressor(bootstrap=False, 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=100, n_jobs=None,
                    oob_score=False, random_state=None, verbose=0,
                    warm_start=False)
```

```
model.feature_importances_
```

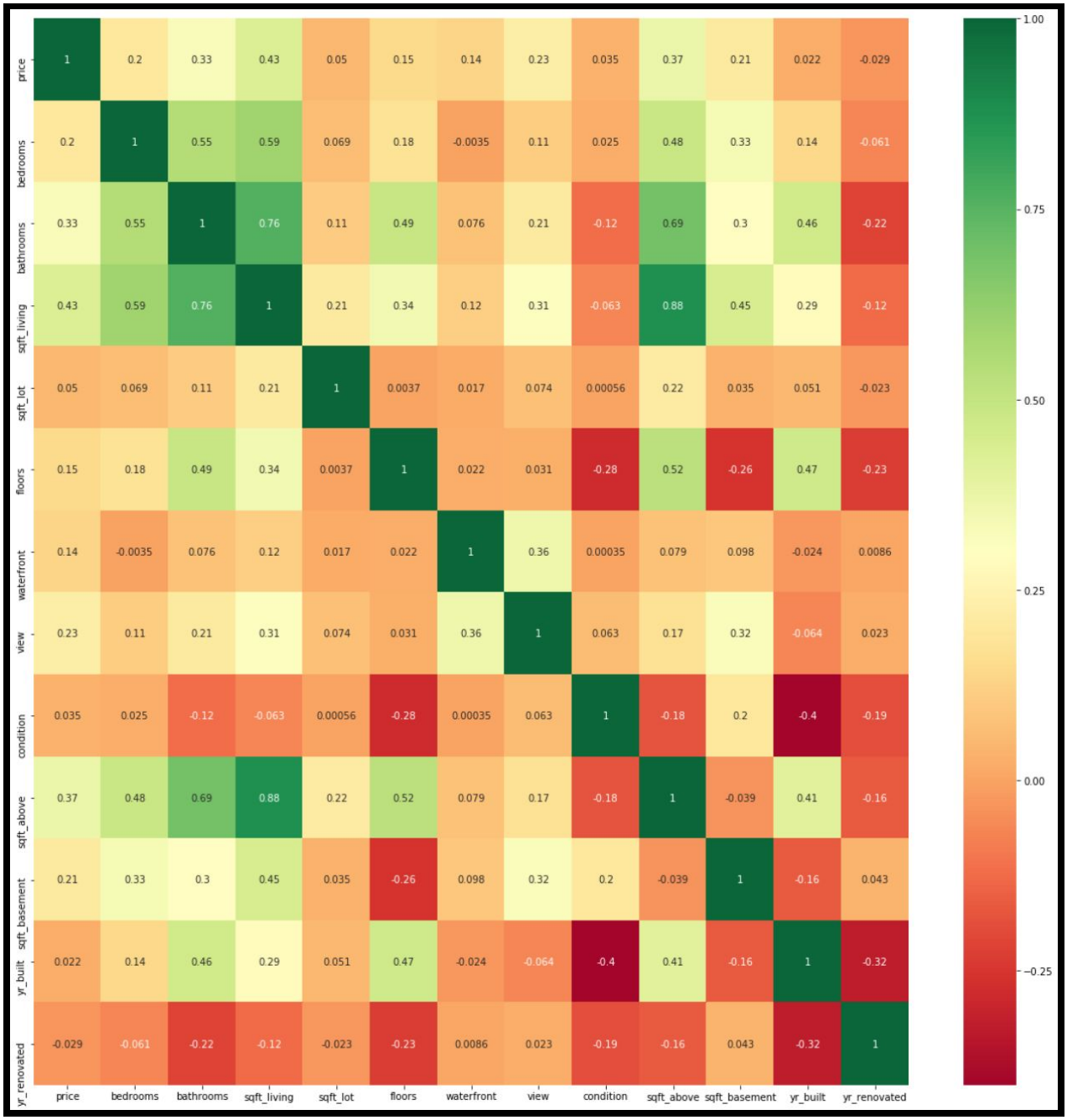
```
array([0.03421333, 0.08164022, 0.25061447, 0.09248207, 0.0296628 ,
       0.01333475, 0.03190069, 0.0510895 , 0.16417404, 0.04413165,
       0.18532957, 0.02142691])
```

```
feat_importances = pd.Series(model.feature_importances_,index = X.columns)
feat_importances.nlargest(10).plot(kind='barh')
plt.show()
```



We searched for effect of other columns on price by feature selection.

CORRELATION MATRIX WITH HEATMAP



Heatmap is another way of determining feature selection.

In this moment, we decide to create Polynomial Features.

PREPROCESSING POLYNOMIAL FEATURES

```
from sklearn.preprocessing import PolynomialFeatures
```

```
poly_features = PolynomialFeatures(interaction_only=True)
X_train_poly = poly_features.fit_transform(X_train)
X_test_poly = poly_features.fit_transform(X_test)
poly_model = LinearRegression()
poly_model.fit(X_train_poly,y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
y_test_predict = poly_model.predict(poly_features.fit_transform(X_test))
r2_score(y_test,y_test_predict)
```

```
0.5835449417335691
```

```
sc2 = StandardScaler()
X_poly_scaled = sc2.fit_transform(X_train_poly)
y_scaled = y_train
X_poly_tested = sc2.transform(X_test_poly)
y_tested = y_test
```

```
lin_reg = LinearRegression()
lin_reg.fit(X_poly_scaled,y_scaled)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
y_poly_pred = lin_reg.predict(X_poly_tested)
r2_score(y_tested,y_poly_pred)
```

```
0.5921726562728207
```

Model was pretty improved by preprocessing with polynomial features compared to started.

We then enrich the model with Ridge Regression.

RIDGE REGRESSION

```
from sklearn.linear_model import Ridge
ridge = Ridge(alpha=3.7,random_state=38)
ridge.fit(X_poly_scaled,y_train)
```

```
Ridge(alpha=3.7, copy_X=True, fit_intercept=True, max_iter=None,
      normalize=False, random_state=38, solver='auto', tol=0.001)
```

```
y_ridge_pred = ridge.predict(X_poly_tested)
r2_score(y_test,y_ridge_pred)
```

```
0.599859165492652
```

In general, with more columns for each observation, we'll get more information and the model will be able to learn better from the dataset and therefore, make better predictions. We have gotten much better results with this small dataset which includes 4600 rows. We actually started with 0.25 R2 score and pulled up to 0.60 with ML algorithms.Ridge regression is able to explain 60% of the variability in predicting house prices based on the input features.