project

June 28, 2020

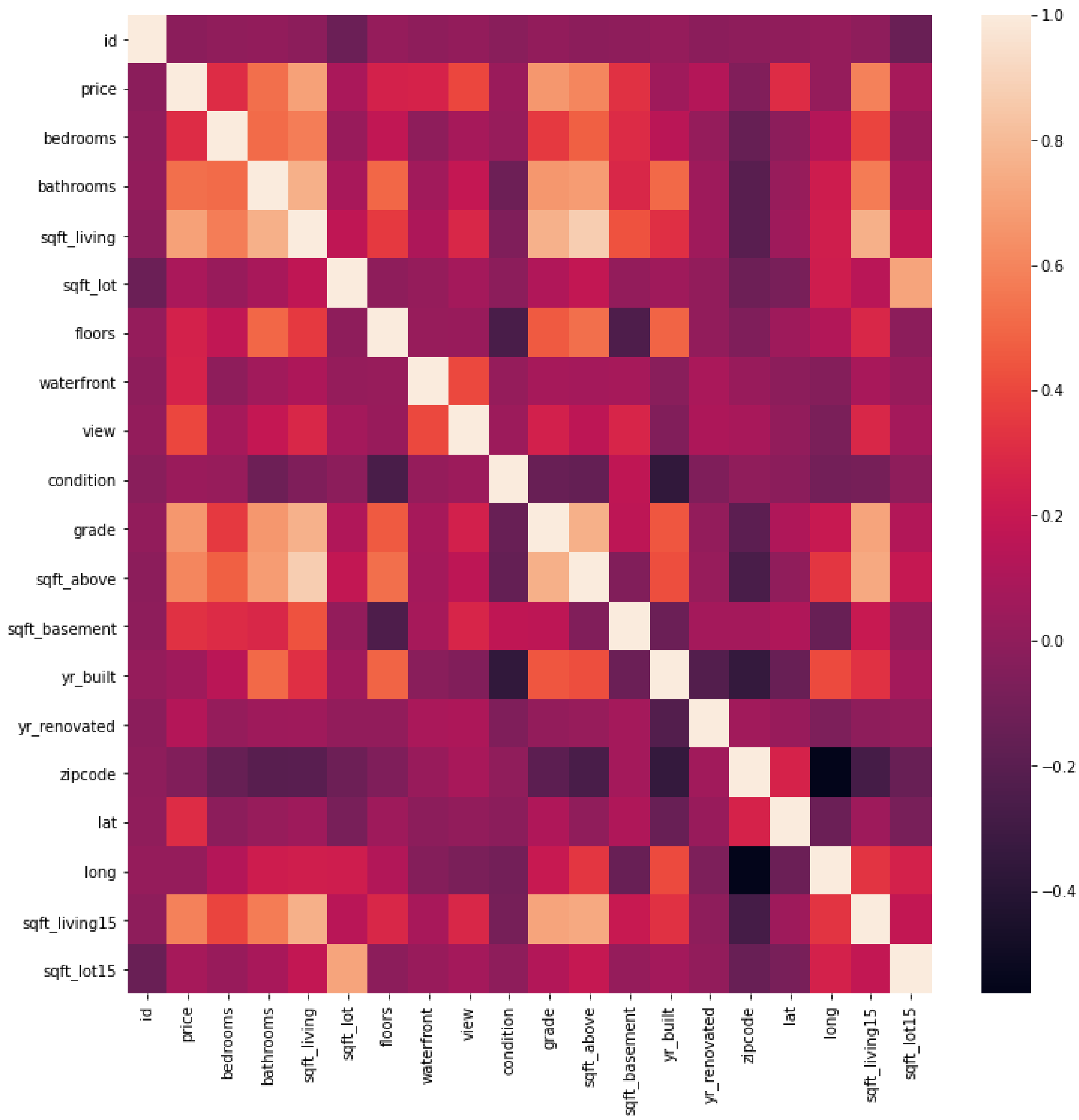
0.1 EDA

[25]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.linear_model import LinearRegression
#1
df = pd.read_csv('kc_house_data.csv')

# Correlation Plot Heatmap
plt.figure(figsize= (12, 12))
sns.heatmap(df.corr())
df.corr(method='pearson')
```

[25]:
```
                     id     price  bedrooms  bathrooms  sqft_living  sqft_lot  \
id             1.000000 -0.016772  0.001150   0.005162    -0.012241 -0.131911
price         -0.016772  1.000000  0.308787   0.525906     0.701917  0.089876
bedrooms       0.001150  0.308787  1.000000   0.514508     0.578212  0.032471
bathrooms      0.005162  0.525906  0.514508   1.000000     0.755758  0.088373
sqft_living   -0.012241  0.701917  0.578212   0.755758     1.000000  0.173453
sqft_lot      -0.131911  0.089876  0.032471   0.088373     0.173453  1.000000
floors         0.018608  0.256804  0.177944   0.502582     0.353953 -0.004814
waterfront    -0.002727  0.266398 -0.006834   0.063744     0.103854  0.021632
view           0.011536  0.397370  0.080008   0.188386     0.284709  0.074900
condition     -0.023803  0.036056  0.026496  -0.126479    -0.059445 -0.008830
grade          0.008188  0.667951  0.356563   0.665838     0.762779  0.114731
sqft_above    -0.010799  0.605368  0.479386   0.686668     0.876448  0.184139
sqft_basement -0.005193  0.323799  0.302808   0.283440     0.435130  0.015418
yr_built       0.021617  0.053953  0.155670   0.507173     0.318152  0.052946
yr_renovated  -0.016925  0.126424  0.018389   0.050544     0.055308  0.007686
zipcode       -0.008211 -0.053402 -0.154092  -0.204786    -0.199802 -0.129586
lat           -0.001798  0.306692 -0.009951   0.024280     0.052155 -0.085514
long           0.020672  0.022036  0.132054   0.224903     0.241214  0.230227
```

```
sqft_living15 -0.002701  0.585241  0.393406   0.569884     0.756402  0.144763
sqft_lot15    -0.138557  0.082845  0.030690   0.088303     0.184342  0.718204

                 floors  waterfront      view  condition     grade  \
id             0.018608   -0.002727  0.011536  -0.023803  0.008188
price          0.256804    0.266398  0.397370   0.036056  0.667951
bedrooms       0.177944   -0.006834  0.080008   0.026496  0.356563
bathrooms      0.502582    0.063744  0.188386  -0.126479  0.665838
sqft_living    0.353953    0.103854  0.284709  -0.059445  0.762779
sqft_lot      -0.004814    0.021632  0.074900  -0.008830  0.114731
floors         1.000000    0.023755  0.028814  -0.264075  0.458794
waterfront     0.023755    1.000000  0.401971   0.016611  0.082888
view           0.028814    0.401971  1.000000   0.045999  0.251728
condition     -0.264075    0.016611  0.045999   1.000000 -0.146896
grade          0.458794    0.082888  0.251728  -0.146896  1.000000
sqft_above     0.523989    0.072109  0.167609  -0.158904  0.756073
sqft_basement -0.245715    0.080559  0.277078   0.173849  0.168220
yr_built       0.489193   -0.026153 -0.053636  -0.361592  0.447865
yr_renovated   0.006427    0.092873  0.103951  -0.060788  0.014261
zipcode       -0.059541    0.030272  0.084622   0.002888 -0.185771
lat            0.049239   -0.014306  0.005871  -0.015102  0.113575
long           0.125943   -0.041904 -0.078107  -0.105877  0.200341
sqft_living15  0.280102    0.086507  0.280681  -0.093072  0.713867
sqft_lot15    -0.010722    0.030781  0.072904  -0.003126  0.120981

               sqft_above  sqft_basement  yr_built  yr_renovated   zipcode  \
id              -0.010799      -0.005193  0.021617     -0.016925 -0.008211
price            0.605368       0.323799  0.053953      0.126424 -0.053402
bedrooms         0.479386       0.302808  0.155670      0.018389 -0.154092
bathrooms        0.686668       0.283440  0.507173      0.050544 -0.204786
sqft_living      0.876448       0.435130  0.318152      0.055308 -0.199802
sqft_lot         0.184139       0.015418  0.052946      0.007686 -0.129586
floors           0.523989      -0.245715  0.489193      0.006427 -0.059541
waterfront       0.072109       0.080559 -0.026153      0.092873  0.030272
view             0.167609       0.277078 -0.053636      0.103951  0.084622
condition       -0.158904       0.173849 -0.361592     -0.060788  0.002888
grade            0.756073       0.168220  0.447865      0.014261 -0.185771
sqft_above       1.000000      -0.052156  0.424037      0.023251 -0.261570
sqft_basement   -0.052156       1.000000 -0.133064      0.071233  0.074725
yr_built         0.424037      -0.133064  1.000000     -0.224907 -0.347210
yr_renovated     0.023251       0.071233 -0.224907      1.000000  0.064325
zipcode         -0.261570       0.074725 -0.347210      0.064325  1.000000
lat             -0.001199       0.110414 -0.148370      0.029350  0.266742
long             0.344842      -0.144546  0.409993     -0.068321 -0.564259
sqft_living15    0.731767       0.200443  0.326377     -0.002695 -0.279299
sqft_lot15       0.195077       0.017550  0.070777      0.007944 -0.147294
```

	lat	long	sqft_living15	sqft_lot15
id	-0.001798	0.020672	-0.002701	-0.138557
price	0.306692	0.022036	0.585241	0.082845
bedrooms	-0.009951	0.132054	0.393406	0.030690
bathrooms	0.024280	0.224903	0.569884	0.088303
sqft_living	0.052155	0.241214	0.756402	0.184342
sqft_lot	-0.085514	0.230227	0.144763	0.718204
floors	0.049239	0.125943	0.280102	-0.010722
waterfront	-0.014306	-0.041904	0.086507	0.030781
view	0.005871	-0.078107	0.280681	0.072904
condition	-0.015102	-0.105877	-0.093072	-0.003126
grade	0.113575	0.200341	0.713867	0.120981
sqft_above	-0.001199	0.344842	0.731767	0.195077
sqft_basement	0.110414	-0.144546	0.200443	0.017550
yr_built	-0.148370	0.409993	0.326377	0.070777
yr_renovated	0.029350	-0.068321	-0.002695	0.007944
zipcode	0.266742	-0.564259	-0.279299	-0.147294
lat	1.000000	-0.135371	0.048679	-0.086139
long	-0.135371	1.000000	0.335626	0.255586
sqft_living15	0.048679	0.335626	1.000000	0.183515
sqft_lot15	-0.086139	0.255586	0.183515	1.000000

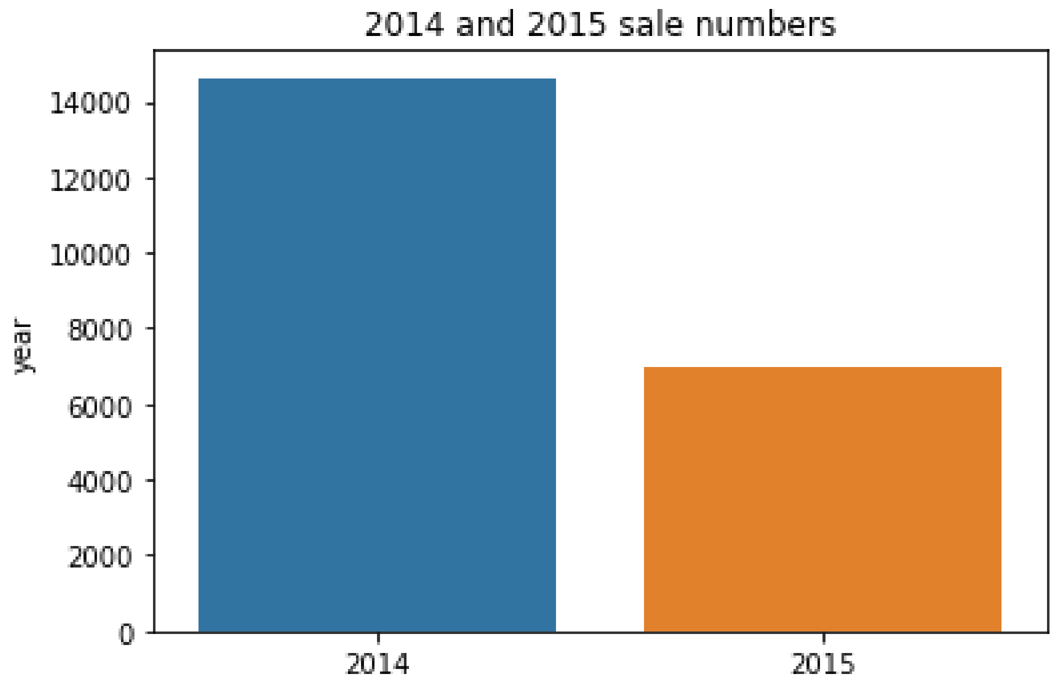


Most Positive: Sqft, Bedrooms, Bathrooms Most Negative: Zipcode, Lat, Long

(2). Sale numbers Vs. (years,months) and Sale prices correlation Vs. (years,months)

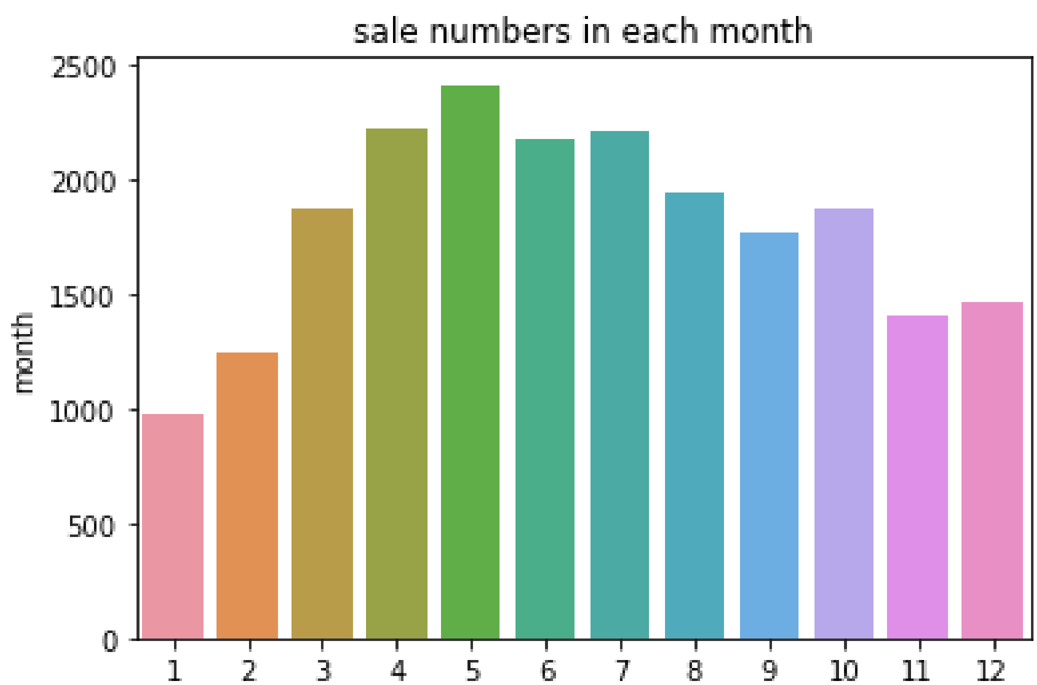
[26]:
```
df['month'] = pd.DatetimeIndex(df['date']).month
df['year'] = pd.DatetimeIndex(df['date']).year
month = df['month'].value_counts()
year = df['year'].value_counts()
sns.barplot(year.index.tolist(),year)
plt.title("2014 and 2015 sale numbers")
```

[26]:
```
Text(0.5, 1.0, '2014 and 2015 sale numbers')
```



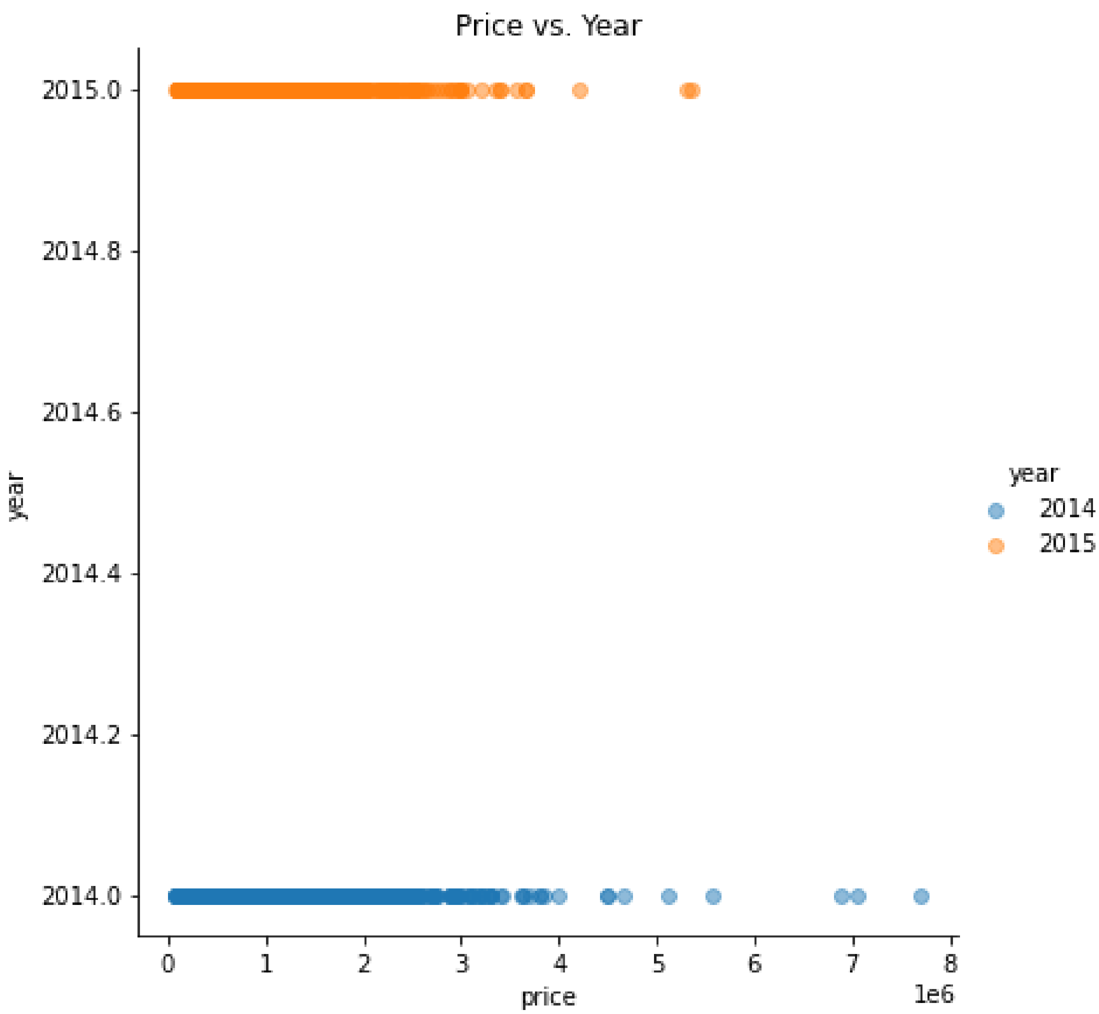
[27]:
```
sns.barplot(month.index.tolist(),month)
plt.title("sale numbers in each month")
```

[27]: Text(0.5, 1.0, 'sale numbers in each month')



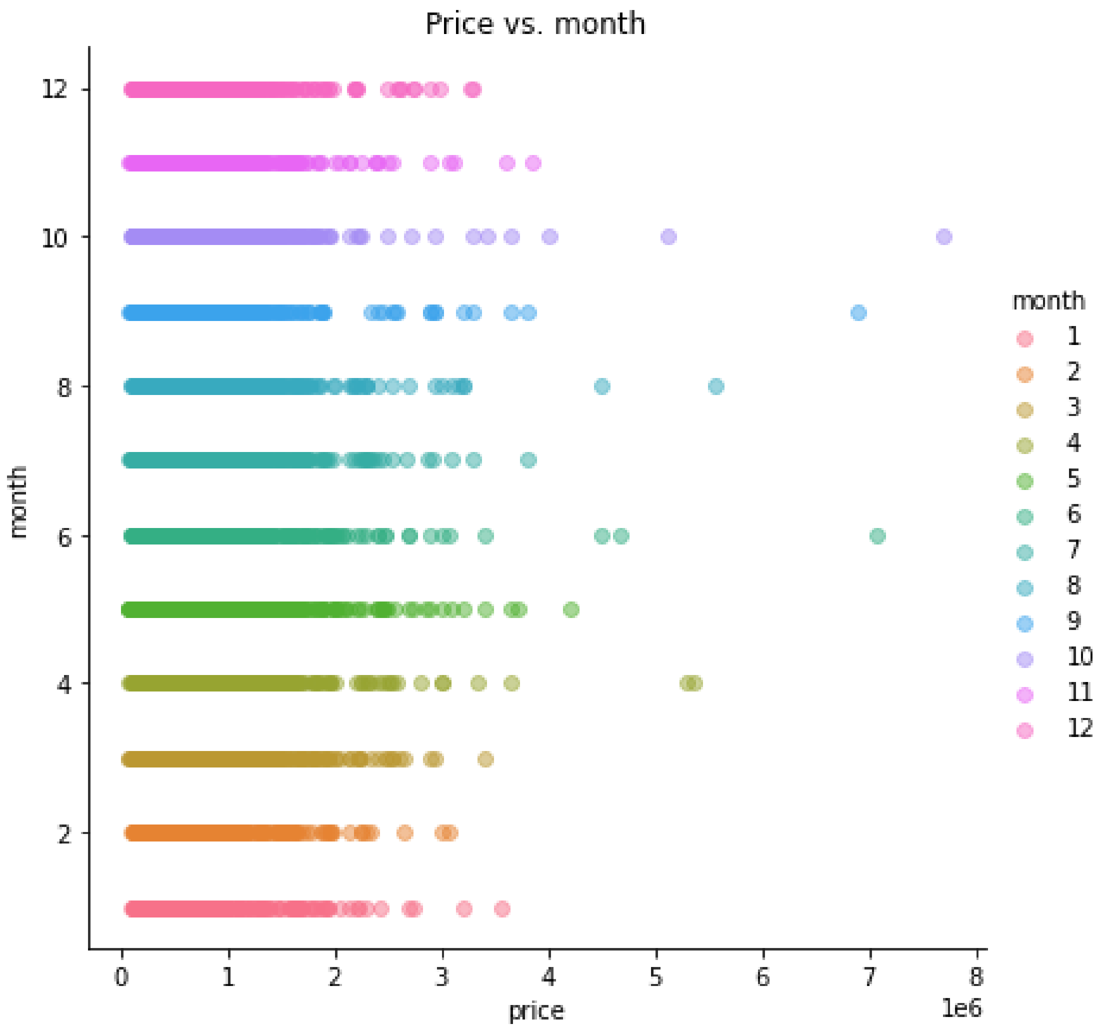
[28]:
```
g=sns.FacetGrid(df,hue='year',height=6)
g.map(plt.scatter,'price','year',alpha=0.5)
g.add_legend()
plt.title("Price vs. Year")
```

[28]: Text(0.5, 1.0, 'Price vs. Year')



[29]:
```
g=sns.FacetGrid(df,hue='month',height=6)
g.map(plt.scatter,'price','month',alpha=0.5)
g.add_legend()
plt.title("Price vs. month")
```

[29]: Text(0.5, 1.0, 'Price vs. month')



[30]:
```
print("Price correlation with year: ",df['price'].corr(df['year']))
print("Price correlation with month: ",df['price'].corr(df['month']))
```

```
Price correlation with year:  0.003727139624315499
Price correlation with month:  -0.009928289245273971
```

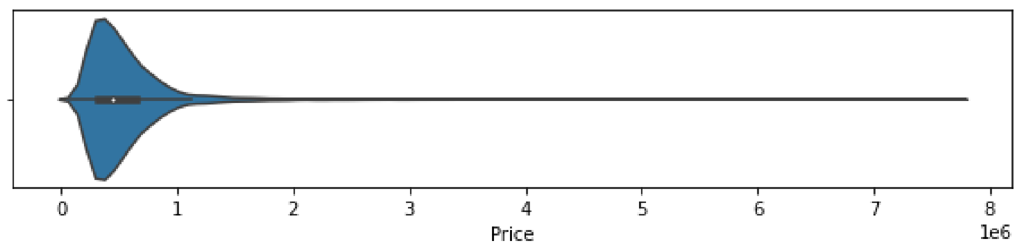
[31]:
```
#3 from sklearn.linear_model import LinearRegression
fig, (ax, box, sq, yr) = plt.subplots(4, figsize=(10,10))
plt.subplots_adjust(hspace = .5)
# Price
ax = sns.violinplot(ax = ax, x = df['price'])
print(np.percentile(df['price'], [25, 50, 75]))
ax.set(xlabel = 'Price')

#'bedrooms', 'bathrooms', 'floors', 'condition'
```

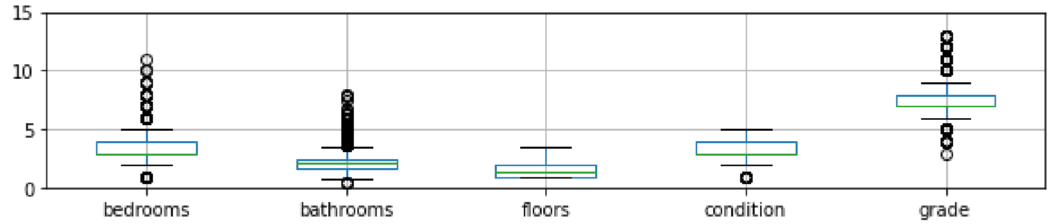
```
box = df.boxplot(ax = box, column = ['bedrooms', 'bathrooms', 'floors',␣
↪'condition', 'grade'])
box.set_ylim([0,15])

# Square feet living room
sq = sns.distplot(df.sqft_living, ax = sq)
sq.set(ylabel = 'density')

# Year built
yr = sns.distplot(df.yr_built, ax = yr)
yr.set(ylabel = 'density')
```

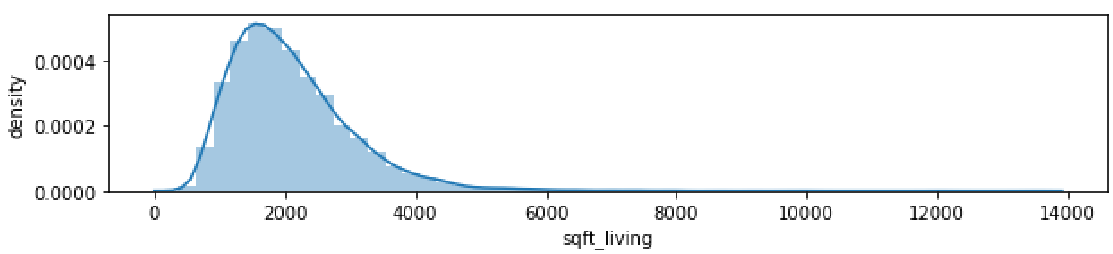
```
[322000. 450000. 645000.]
```

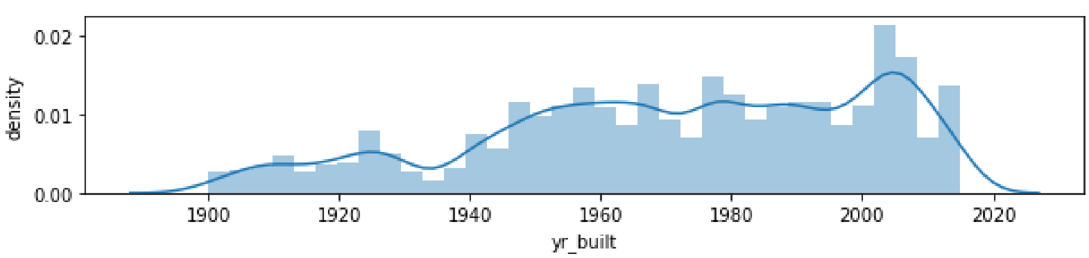
[31]:

```
[Text(0, 0.5, 'density')]
```









1 The first graph shows the distribution of prices in a violin plot. We can tell the 25-75% quartile is between $322,000 and $645,000 2 The second plot shows the box plots of bedrooms, bathrooms, floors, condition, grade. The medians are: Bedrooms ~ 3 Bathrooms ~ 2.5 Floors ~ 2 Condition ~ 3 Grade ~ 7 3 The third plot shows the distribution of square foot in living room. This plot is skewed with the most being ~1800 sqft 4 The last plot is the distribution of houses built over time. There has been a recent phase of construction in the 2000s, which means many houses are newly built and in decent condition.

(4). Create the scoring function for 'Grade' with accuracy:70%

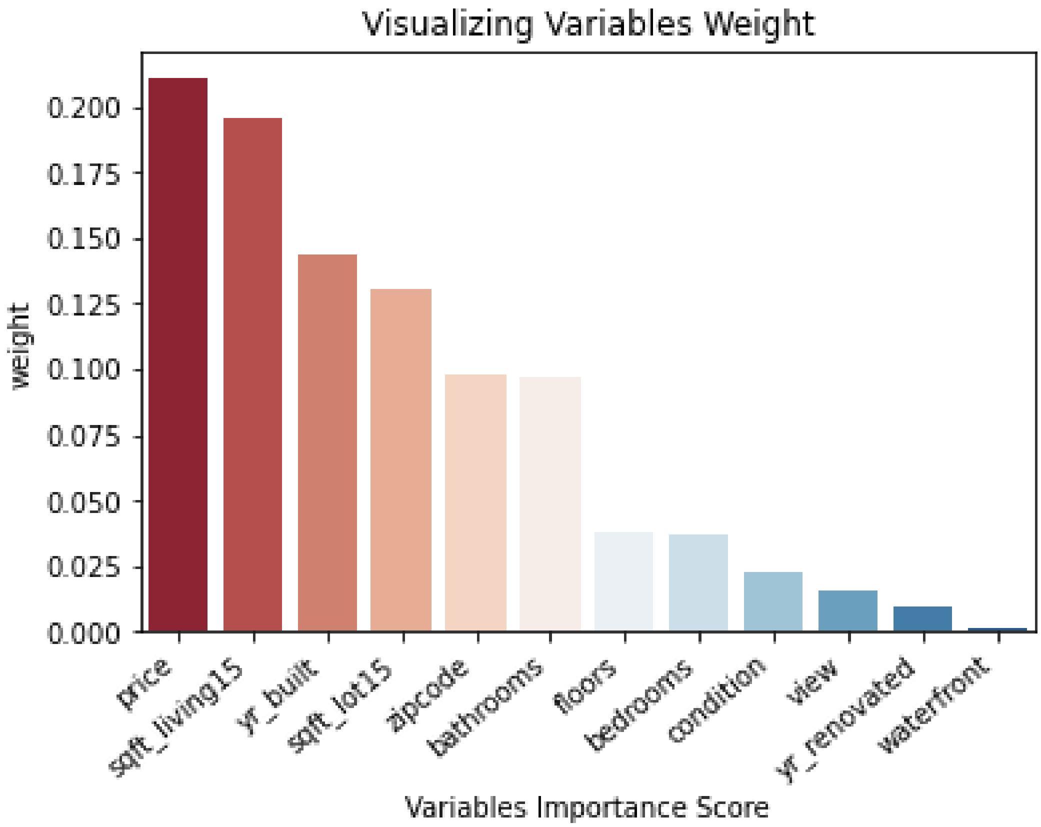
[32]:
```
X=df[['price','bedrooms','bathrooms','sqft_living15','sqft_lot15','floors','waterfront',"condi
      "yr_renovated"]]
y=df['grade']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

[33]:
```
clf = RandomForestClassifier(n_estimators=100)
clf.fit(X_train,y_train)
y_predit=clf.predict(X_test)
```

[34]:
```
variables = pd.Series(clf.feature_importances_,index=X.columns).
 ↪sort_values(ascending=False)
variables
```

[34]:
```
price            0.210859
sqft_living15    0.195388
yr_built         0.144138
sqft_lot15       0.130732
zipcode          0.097867
bathrooms        0.096445
floors           0.037984
bedrooms         0.037329
condition        0.022929
view             0.015756
yr_renovated     0.009387
waterfront       0.001187
dtype: float64
```

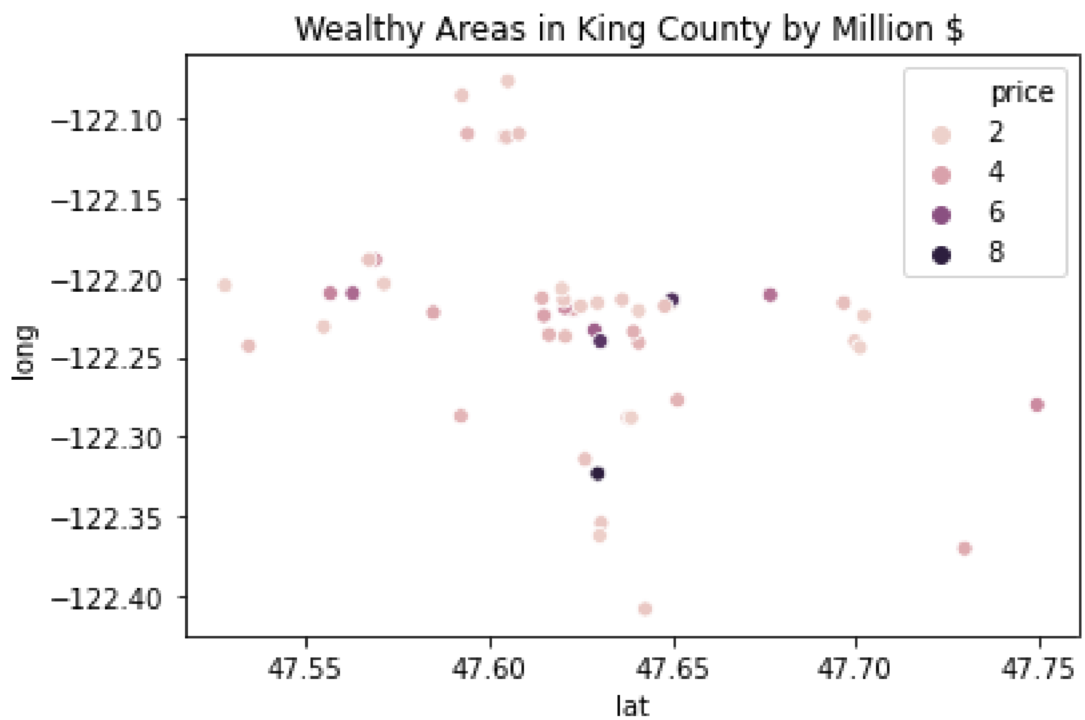
[35]:
```
ax=sns.barplot(x=variables.index, y=variables,palette=sns.color_palette("RdBu",␣
 ↪12))
ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="right")
# Add labels to your graph
plt.xlabel('Variables Importance Score')
plt.ylabel('weight')
plt.title("Visualizing Variables Weight")
plt.show()
```



[36]:
```
print("Scoring function accuracy:",metrics.accuracy_score(y_test, y_predit))
```

```
Scoring function accuracy: 0.7037037037037037
```

[37]:
```
#5
import seaborn as sns

wealthy = df.loc[df['price'] >= 3000000]

plt.title("Wealthy Areas in King County by Million $")
ax = sns.scatterplot(x=wealthy.lat, y=wealthy.long, hue=wealthy.price)
```



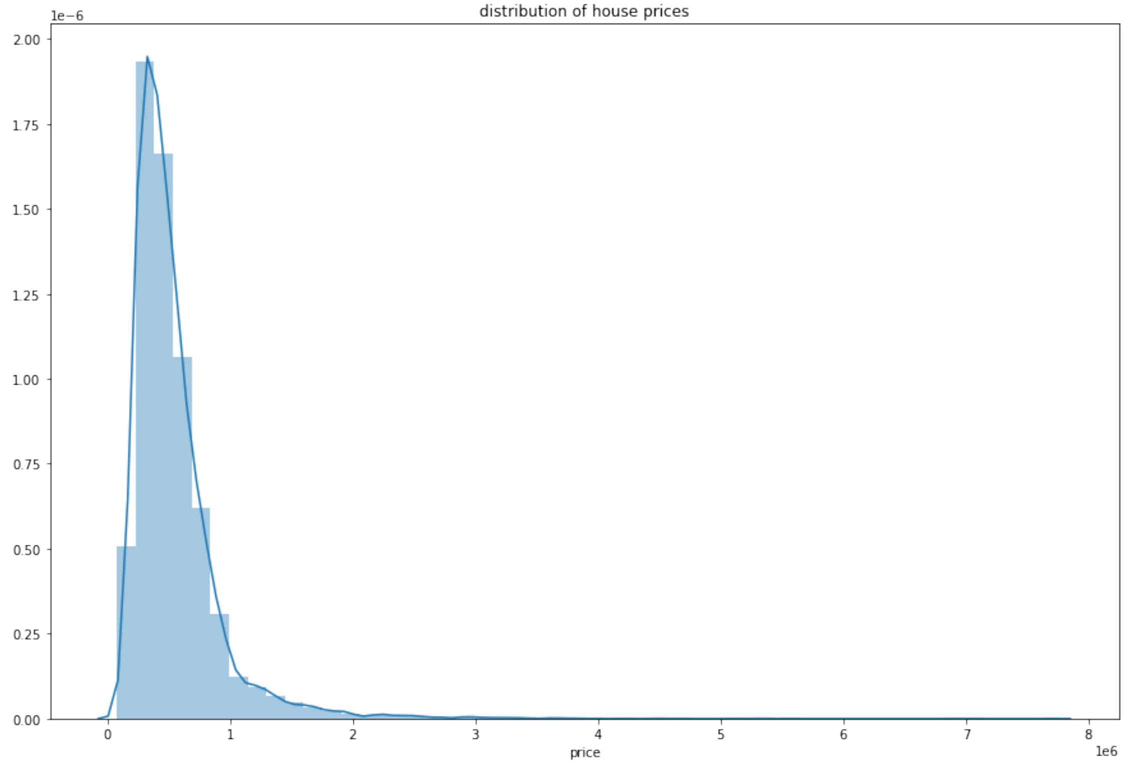
0.2 Modeling

0.2.1 Linear Regression

[38]:
```
X=df[['bedrooms','bathrooms','sqft_living15','grade','sqft_lot15','floors','waterfront',"condi
      "yr_renovated"]]
y=df['price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

[39]:
```
plt.figure(figsize=(15,10))
plt.tight_layout()
sns.distplot(df['price'])
plt.title("distribution of house prices")
```

[39]: Text(0.5, 1.0, 'distribution of house prices')



[40]:
```
reg = LinearRegression()
reg.fit(X_train,y_train)
coeff_df = pd.DataFrame(reg.coef_, X.columns, columns=['Coefficient'])
coeff_df
```

[40]:

	Coefficient
bedrooms	-8017.909375
bathrooms	110699.450542
sqft_living15	89.997854
grade	159993.490592
sqft_lot15	-0.237560
floors	18715.992830
waterfront	582458.379984
condition	24022.402672
yr_built	-4049.375666
zipcode	21.457573
view	48132.877243
yr_renovated	10.387863

[41]:
```
y_predit = reg.predict(X_test)
accurate_rate=1-np.mean(np.abs(y_predit-y_test)/y_test)
print("Accuracy: ",accurate_rate)
```

```
Accuracy:  0.6995068223297689
```

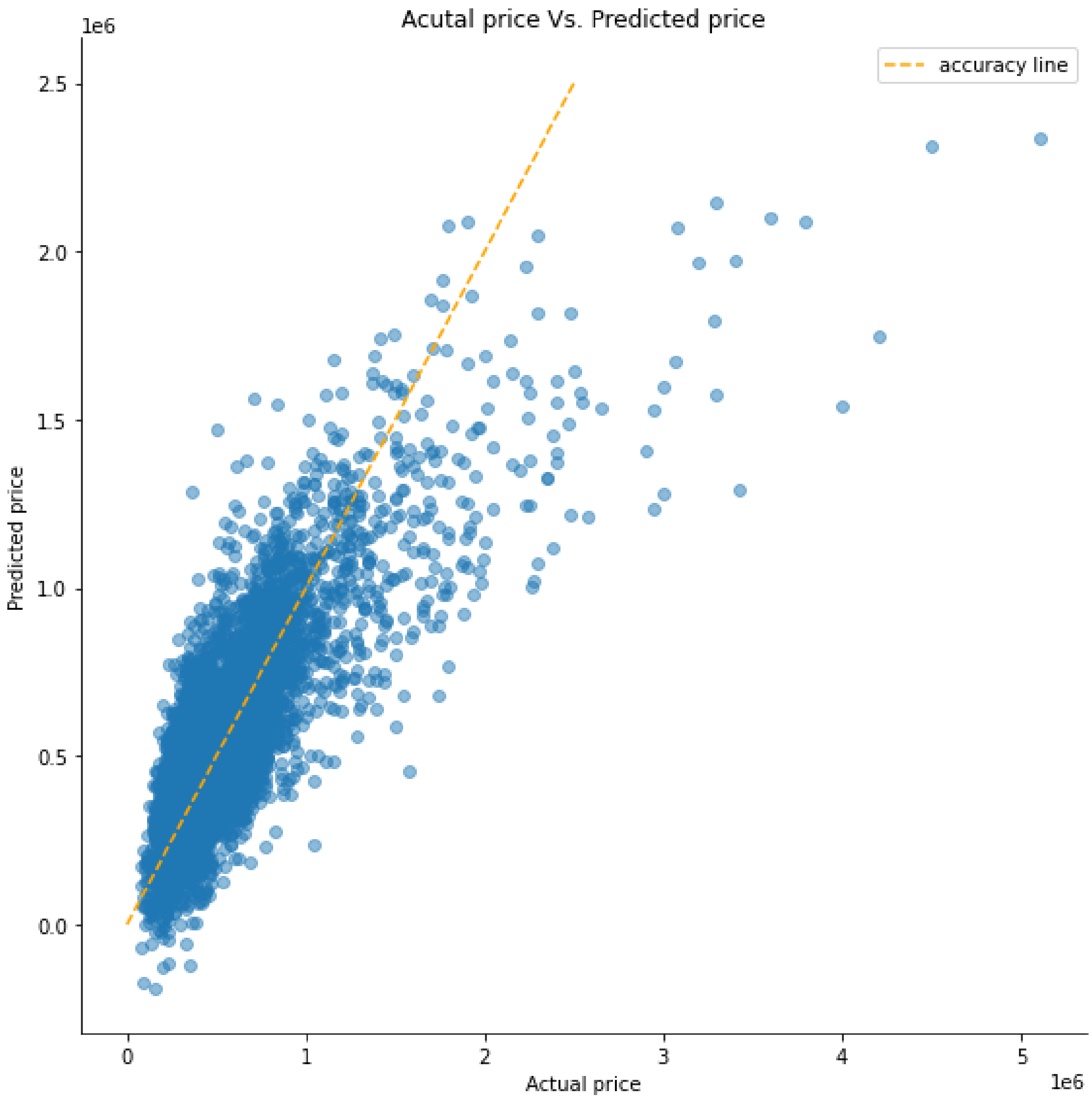
[42]:
```
result = pd.DataFrame({'Actual price': y_test, 'Predicted price': y_predit})
result.head(8)
```

[42]:

	Actual price	Predicted price
16644	570000.0	948422.541466
11196	427800.0	410592.227532
534	204000.0	106154.319955
16261	270000.0	503446.838729
17002	760369.0	672881.313048
625	289500.0	185084.386431
10450	330000.0	287006.767480
13494	570000.0	384449.655879

[43]:
```
g = sns.FacetGrid(result,height=8)
g.map(plt.scatter,'Actual price','Predicted price',alpha=0.5)
plt.plot([0,2500000],[0,2500000],ls='--',color='orange',label='accuracy line')
plt.title("Acutal price Vs. Predicted price")
plt.legend()
print("Model accuracy: ",accurate_rate)
```

```
Model accuracy:  0.6995068223297689
```



Linear regression is a model to find possible W, in "Y= XW+error" which has minimum Mean squared error(MSE). This linear regression model accuracy rate is around 66%.

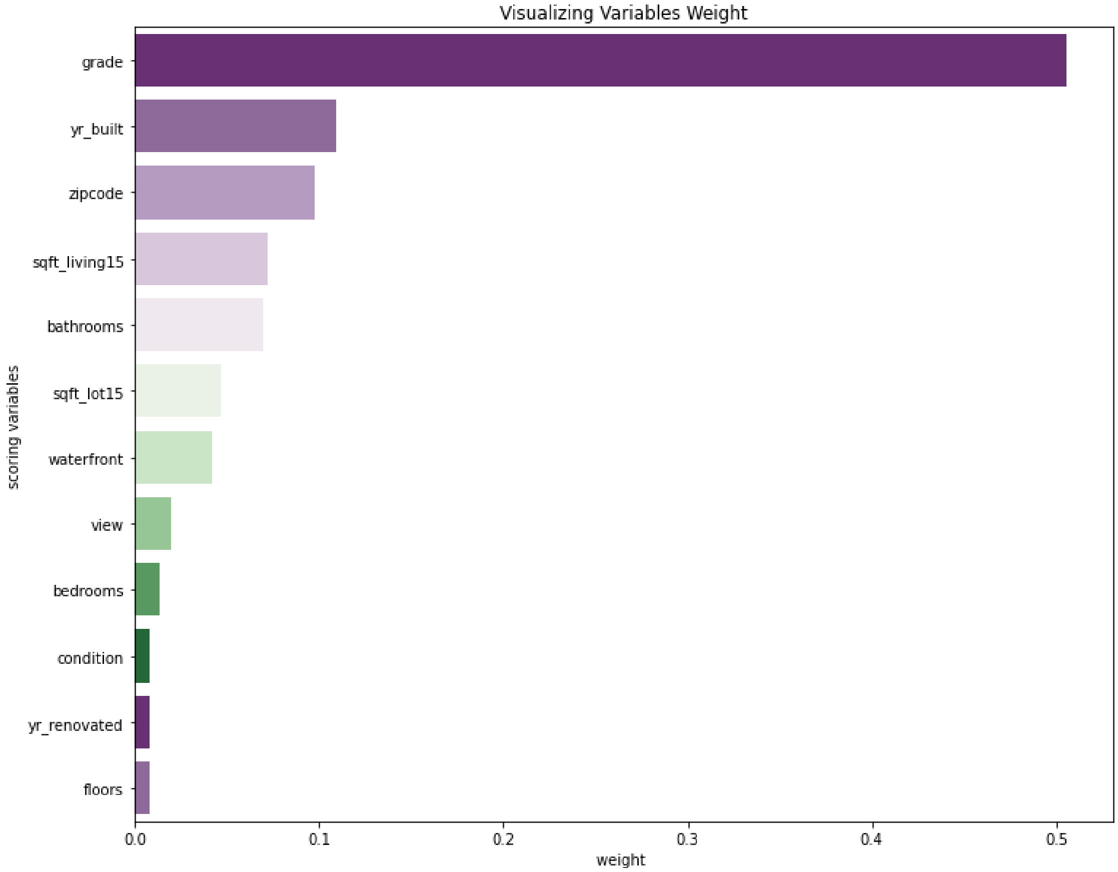
0.2.2 Random Forest Model

[44]:
```
X=df[['bedrooms','bathrooms','sqft_living15','grade','sqft_lot15','floors','waterfront',"condi
      "yr_renovated"]]
y=df['price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
clf = RandomForestRegressor(n_estimators=100)
clf.fit(X_train,y_train)
y_predit=clf.predict(X_test)
```

[45]:
```
variables = pd.Series(clf.feature_importances_,index=X.columns).
↪sort_values(ascending=False)
variables
```

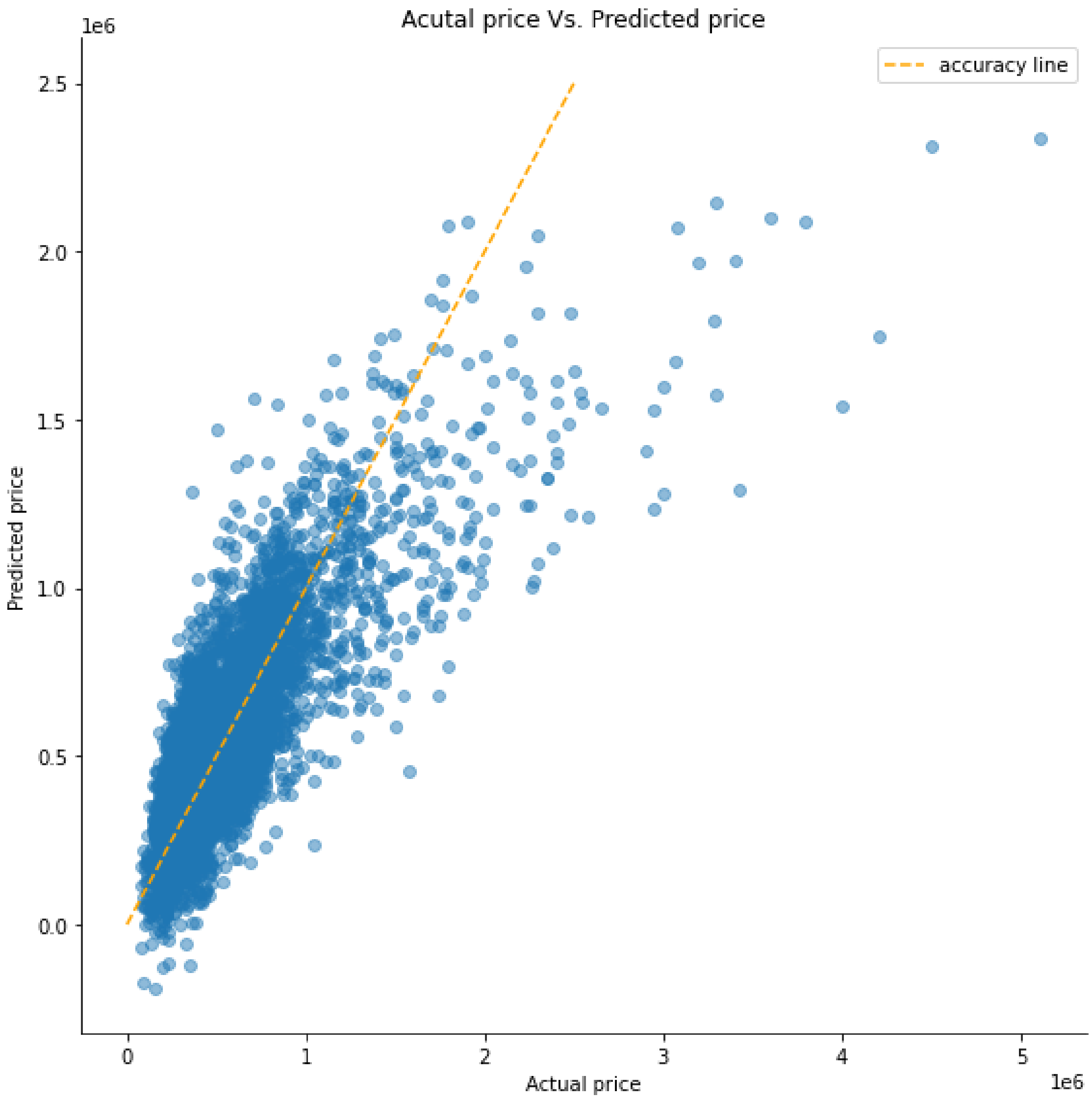
[45]:
```
grade            0.505252
yr_built         0.109656
zipcode          0.097298
sqft_living15    0.072033
bathrooms        0.069986
sqft_lot15       0.047110
waterfront       0.041807
view             0.019622
bedrooms         0.013280
condition        0.008312
yr_renovated     0.007956
floors           0.007688
dtype: float64
```

[46]:
```
ax=sns.barplot(x=variables, y=variables.index,palette=sns.color_palette("PRGn",␣
↪10))
ax.figure.set_size_inches(12,10)
# Add labels to your graph
plt.xlabel('weight ')
plt.ylabel('scoring variables')
plt.title("Visualizing Variables Weight")
plt.show()
```



[47]:
```
g = sns.FacetGrid(result,height=8)
g.map(plt.scatter,'Actual price','Predicted price',alpha=0.5)
plt.plot([0,2500000],[0,2500000],ls='--',color='orange',label='accuracy line')
plt.title("Acutal price Vs. Predicted price")
plt.legend()
print("Model accuracy: ",accurate_rate)
```

```
Model accuracy:  0.6995068223297689
```



[48]:
```
accurate_rate=1-np.mean(np.abs(y_predit-y_test)/y_test)
print("Random Forest accuracy:",accurate_rate)
```

```
Random Forest accuracy: 0.8274844176466372
```

Random forest regression is to select random samples and build decision trees for each sample. Then, Perform a vote for each predicted result and select the prediction result with the most votes as the final prediction. The Random forest model has accuracy rate around 81%.

0.2.3 K Nearest Neighbors

[49]:
```
from sklearn import neighbors
X=df[['bedrooms','bathrooms','floors','grade','sqft_living15','waterfront',"condition","yr_bui
      "yr_renovated"]]
y=df['price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

model = neighbors.KNeighborsRegressor(n_neighbors=10)
model.fit(X_train, y_train)
preds = model.predict(X_test)
```

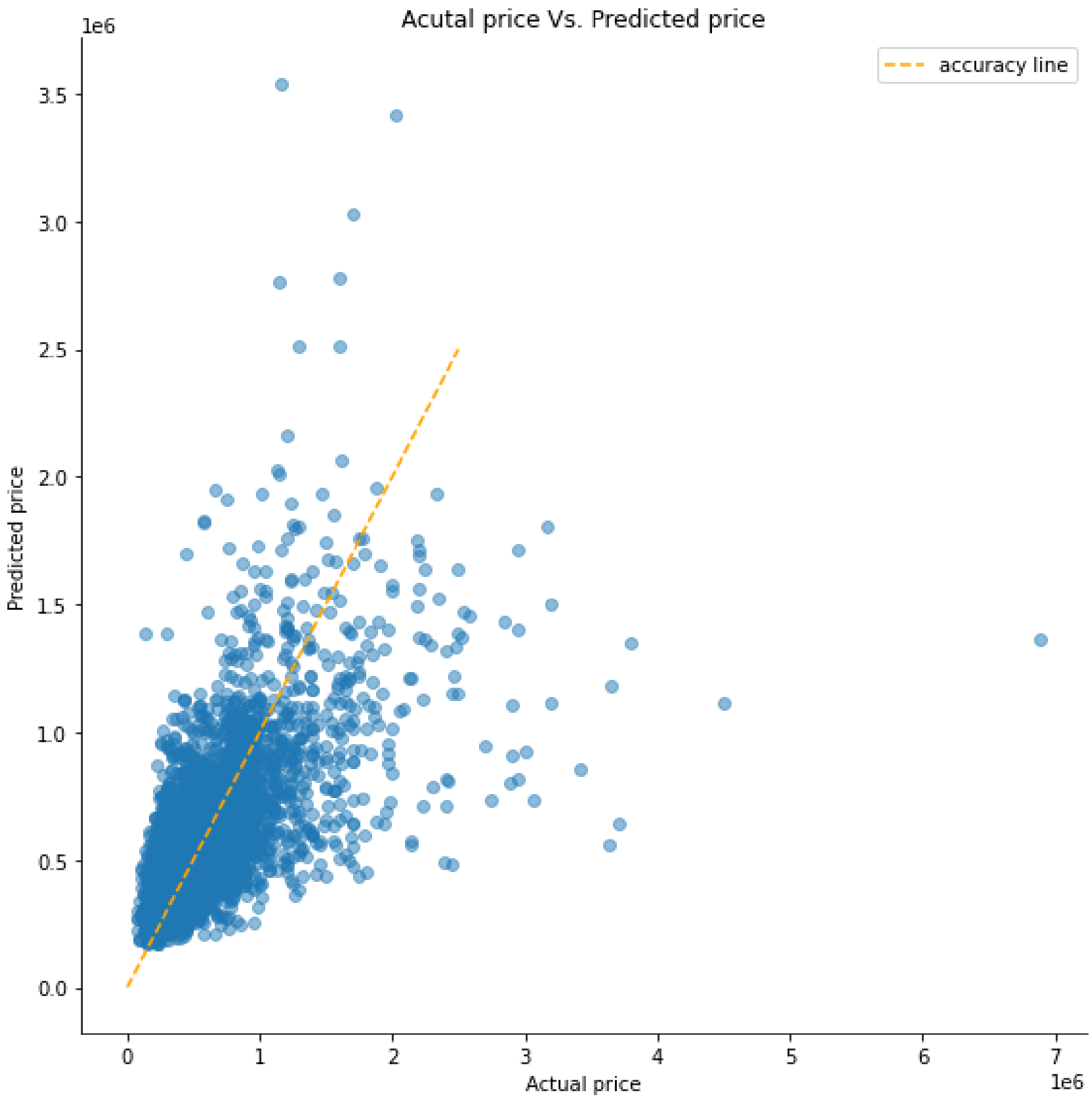
[50]:
```
result = pd.DataFrame({'Actual price': y_test, 'Predicted price': preds})
result.head(8)
```

[50]:

	Actual price	Predicted price
11743	250000.0	286345.0
3810	532000.0	422565.0
9408	375000.0	435894.0
10552	195000.0	203434.5
5363	1600000.0	657495.0
8963	394999.0	407325.0
18741	999000.0	433090.0
7915	305000.0	298410.0

[51]:
```
g = sns.FacetGrid(result,height=8)
g.map(plt.scatter,'Actual price','Predicted price',alpha=0.5)
plt.plot([0,2500000],[0,2500000],ls='--',color='orange',label='accuracy line')
plt.title("Acutal price Vs. Predicted price")
plt.legend()
```

[51]: <matplotlib.legend.Legend at 0x19dc50ec748>



[52]:
```
print('Nearest Neighbors Accuracy: ', model.score(X_test, y_test))
```

```
Nearest Neighbors Accuracy:  0.4211697320021205
```

KNN works by finding the distances between a query and all the examples in the data, selecting the specified number examples (K) closest to the query, then votes for the most frequent label (in the case of classification) or averages the labels (in the case of regression).

This method is not the best because it yielded a ~45% accuracy.