King County House Price Prediction

In [1]:

```
import numpy as np
import pandas as pd

import seaborn as sns
import matplotlib.pyplot as plt

from scipy import stats
from scipy.stats import boxcox

from sklearn import metrics
from sklearn.metrics import mean_squared_error, mean_absolute_error,r2_score


from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR

from sklearn import ensemble
import xgboost as xgb
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import AdaBoostRegressor
import catboost as cb
```

In [2]:

```
data = pd.read_csv('kc_house_data.csv')
df = pd.DataFrame(data)
df.head()
```

Out[2]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0

5 rows × 21 columns

In [3]:

```
df.shape
```

Out[3]:

```
(21613, 21)
```

In [4]:

```
df.describe()
```

Out[4]:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	
count	2.161300e+04	2.161300e+04	21613.000000	21613.000000	21613.000000	2.161300e+04	2
mean	4.580302e+09	5.400881e+05	3.370842	2.114757	2079.899736	1.510697e+04	
std	2.876566e+09	3.671272e+05	0.930062	0.770163	918.440897	4.142051e+04	
min	1.000102e+06	7.500000e+04	0.000000	0.000000	290.000000	5.200000e+02	
25%	2.123049e+09	3.219500e+05	3.000000	1.750000	1427.000000	5.040000e+03	
50%	3.904930e+09	4.500000e+05	3.000000	2.250000	1910.000000	7.618000e+03	
75%	7.308900e+09	6.450000e+05	4.000000	2.500000	2550.000000	1.068800e+04	
max	9.900000e+09	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21613 entries, 0 to 21612
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   id             21613 non-null  int64  
 1   date           21613 non-null  object 
 2   price          21613 non-null  float64
 3   bedrooms       21613 non-null  int64  
 4   bathrooms      21613 non-null  float64
 5   sqft_living    21613 non-null  int64  
 6   sqft_lot       21613 non-null  int64  
 7   floors         21613 non-null  float64
 8   waterfront     21613 non-null  int64  
 9   view           21613 non-null  int64  
 10  condition      21613 non-null  int64  
 11  grade          21613 non-null  int64  
 12  sqft_above     21613 non-null  int64  
 13  sqft_basement  21613 non-null  int64  
 14  yr_built       21613 non-null  int64  
 15  yr_renovated   21613 non-null  int64  
 16  zipcode        21613 non-null  int64  
 17  lat            21613 non-null  float64
 18  long           21613 non-null  float64
 19  sqft_living15  21613 non-null  int64  
 20  sqft_lot15     21613 non-null  int64  
dtypes: float64(5), int64(15), object(1)
memory usage: 3.5+ MB
```

NULL Values

In [6]:

```
df.isnull().sum()
```

Out[6]:

```
id               0
date             0
price            0
bedrooms         0
bathrooms        0
sqft_living      0
sqft_lot         0
floors           0
waterfront       0
view             0
condition        0
grade            0
sqft_above       0
sqft_basement    0
yr_built         0
yr_renovated     0
zipcode          0
lat              0
long             0
sqft_living15    0
sqft_lot15       0
dtype: int64
```

Correlation

In [7]:

```
df.corr()
```

Out[7]:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	w
id	1.000000	-0.016762	0.001286	0.005160	-0.012258	-0.132109	0.018525	-
price	-0.016762	1.000000	0.308350	0.525138	0.702035	0.089661	0.256794	
bedrooms	0.001286	0.308350	1.000000	0.515884	0.576671	0.031703	0.175429	-
bathrooms	0.005160	0.525138	0.515884	1.000000	0.754665	0.087740	0.500653	
sqft_living	-0.012258	0.702035	0.576671	0.754665	1.000000	0.172826	0.353949	
sqft_lot	-0.132109	0.089661	0.031703	0.087740	0.172826	1.000000	-0.005201	
floors	0.018525	0.256794	0.175429	0.500653	0.353949	-0.005201	1.000000	
waterfront	-0.002721	0.266369	-0.006582	0.063744	0.103818	0.021604	0.023698	
view	0.011592	0.397293	0.079532	0.187737	0.284611	0.074710	0.029444	
condition	-0.023783	0.036362	0.028472	-0.124982	-0.058753	-0.008958	-0.263768	
grade	0.008130	0.667434	0.356967	0.664983	0.762704	0.113621	0.458183	
sqft_above	-0.010842	0.605567	0.477600	0.685342	0.876597	0.183512	0.523885	
sqft_basement	-0.005151	0.323816	0.303093	0.283770	0.435043	0.015286	-0.245705	
yr_built	0.021380	0.054012	0.154178	0.506019	0.318049	0.053080	0.489319	-
yr_renovated	-0.016907	0.126434	0.018841	0.050739	0.055363	0.007644	0.006338	
zipcode	-0.008224	-0.053203	-0.152668	-0.203866	-0.199430	-0.129574	-0.059121	
lat	-0.001891	0.307003	-0.008931	0.024573	0.052529	-0.085683	0.049614	-
long	0.020799	0.021626	0.129473	0.223042	0.240223	0.229521	0.125419	-
sqft_living15	-0.002901	0.585379	0.391638	0.568634	0.756420	0.144608	0.279885	
sqft_lot15	-0.138798	0.082447	0.029244	0.087175	0.183286	0.718557	-0.011269	

In [8]:

```
df.corr()["price"].sort_values(ascending = True)
```

Out[8]:

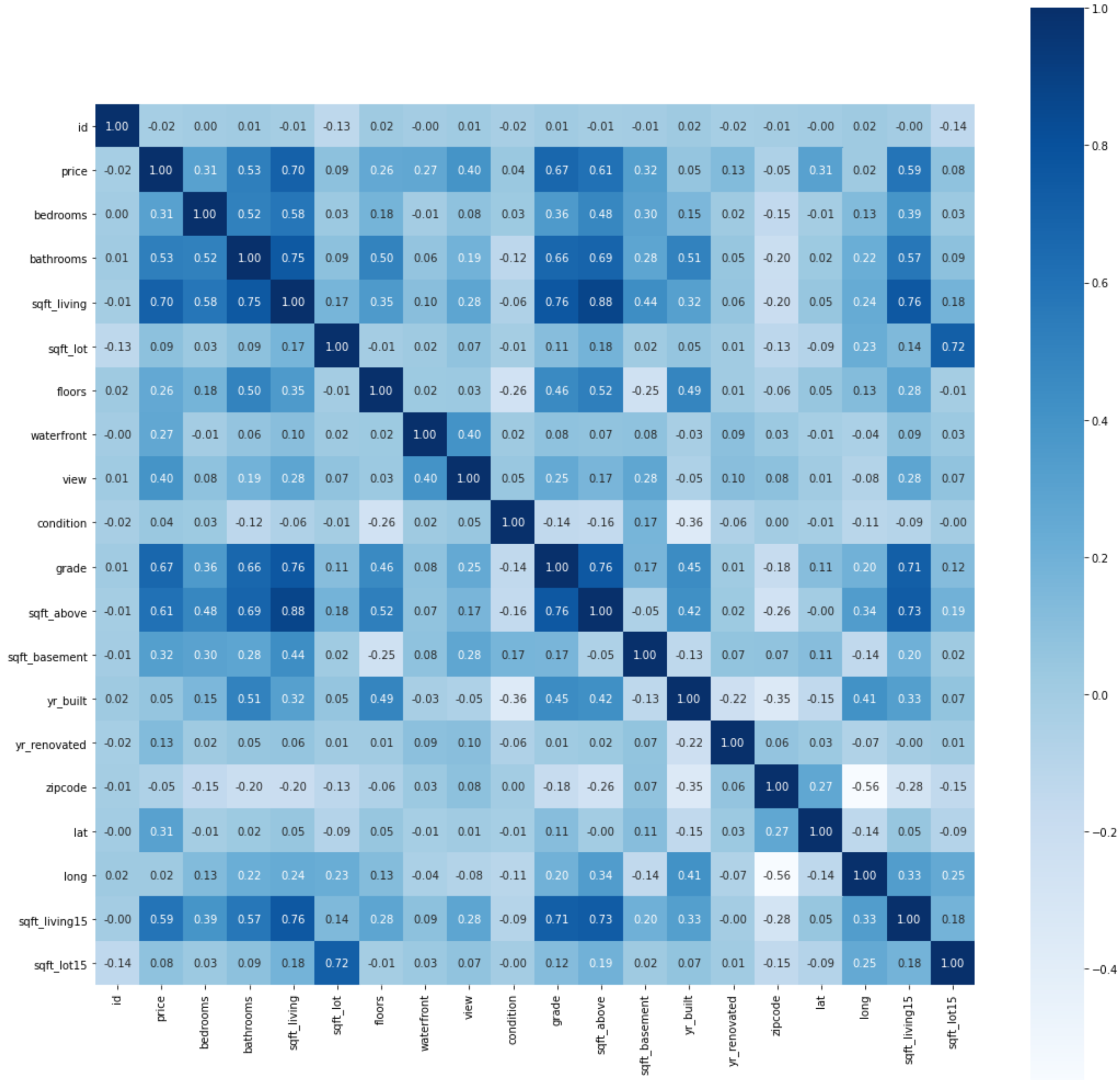
```
zipcode         -0.053203
id              -0.016762
long             0.021626
condition        0.036362
yr_built         0.054012
sqft_lot15       0.082447
sqft_lot         0.089661
yr_renovated     0.126434
floors           0.256794
waterfront       0.266369
lat              0.307003
bedrooms         0.308350
sqft_basement    0.323816
view             0.397293
bathrooms        0.525138
sqft_living15    0.585379
sqft_above       0.605567
grade            0.667434
sqft_living      0.702035
price            1.000000
Name: price, dtype: float64
```

In [9]:

```
plt.figure(figsize=(18,18))
sns.heatmap(df.corr(),annot=True,cmap="Blues",fmt='.2f',square=True)
```

Out[9]:

<AxesSubplot:>



Feature Engineering

In [10]:

```
df =df.drop(["id"],axis = 1)

df.head()
```

Out[10]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	\
0	20141013T000000	221900.0	3	1.00	1180	5650	1.0	0	
1	20141209T000000	538000.0	3	2.25	2570	7242	2.0	0	
2	20150225T000000	180000.0	2	1.00	770	10000	1.0	0	
3	20141209T000000	604000.0	4	3.00	1960	5000	1.0	0	
4	20150218T000000	510000.0	3	2.00	1680	8080	1.0	0	

In [11]:

```
df['date'] = pd.to_datetime(df['date'])

df['year'] = df['date'].dt.year

df['month'] = df['date'].dt.month

df['day'] = df['date'].dt.day

df.head()
```

Out[11]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condit
0	2014-10-13	221900.0	3	1.00	1180	5650	1.0	0	0	
1	2014-12-09	538000.0	3	2.25	2570	7242	2.0	0	0	
2	2015-02-25	180000.0	2	1.00	770	10000	1.0	0	0	
3	2014-12-09	604000.0	4	3.00	1960	5000	1.0	0	0	
4	2015-02-18	510000.0	3	2.00	1680	8080	1.0	0	0	

5 rows × 23 columns

In [12]:

```
df = df.drop(['date'],axis = 1)
```

In [13]:

```
df.head()
```

Out[13]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	gr
0	221900.0	3	1.00	1180	5650	1.0	0	0	3	
1	538000.0	3	2.25	2570	7242	2.0	0	0	3	
2	180000.0	2	1.00	770	10000	1.0	0	0	3	
3	604000.0	4	3.00	1960	5000	1.0	0	0	5	
4	510000.0	3	2.00	1680	8080	1.0	0	0	3	

5 rows × 22 columns

In [14]:

```
df['yr_built'].values[1763],df['yr_renovated'].values[1763],df['year'].values[1763]
```

Out[14]:

```
(2015, 0, 2014)
```

In [15]:

```
df.loc[df["yr_renovated"] == 0, "yr_renovated"] = df['yr_built']
```

In [16]:

```
df['age'] = (df['year'] - df['yr_renovated'])

df.head()
```

Out[16]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	gr
0	221900.0	3	1.00	1180	5650	1.0	0	0	3	
1	538000.0	3	2.25	2570	7242	2.0	0	0	3	
2	180000.0	2	1.00	770	10000	1.0	0	0	3	
3	604000.0	4	3.00	1960	5000	1.0	0	0	5	
4	510000.0	3	2.00	1680	8080	1.0	0	0	3	

5 rows × 23 columns

In [17]:

```
df.corr()["price"].sort_values(ascending = False)
```

Out[17]:

```
price            1.000000
sqft_living      0.702035
grade            0.667434
sqft_above       0.605567
sqft_living15    0.585379
bathrooms        0.525138
view             0.397293
sqft_basement    0.323816
bedrooms         0.308350
lat              0.307003
waterfront       0.266369
floors           0.256794
yr_renovated     0.105755
sqft_lot         0.089661
sqft_lot15       0.082447
yr_built         0.054012
condition        0.036362
long             0.021626
year             0.003576
month           -0.010081
day             -0.014670
zipcode         -0.053203
age             -0.105672
Name: price, dtype: float64
```

In [18]:

```
df = df.drop(columns = ['month','day','year','yr_built','yr_renovated'],axis = 1)
```

In [19]:

```
index=[]
x=0
for i in df['age'].values:
    if i<0:
        print(x,i)
        index.append(x)
    x+=1
```

```
1763 -1
2295 -1
2687 -1
7097 -1
7526 -1
8039 -1
11599 -1
14489 -1
14859 -1
15687 -1
17098 -1
18575 -1
19805 -1
20770 -1
20852 -1
20963 -1
21262 -1
21372 -1
```

In [20]:

```
# The age is in negative, because year_sold < year_built which is not possible. So dropping
```

In [21]:

```
df = df.drop(index,axis = 0)
df = df.drop(columns = ['zipcode'],axis = 1)
```

In [22]:

```
df.head()
```

Out[22]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	gr
0	221900.0	3	1.00	1180	5650	1.0	0	0	3	
1	538000.0	3	2.25	2570	7242	2.0	0	0	3	
2	180000.0	2	1.00	770	10000	1.0	0	0	3	
3	604000.0	4	3.00	1960	5000	1.0	0	0	5	
4	510000.0	3	2.00	1680	8080	1.0	0	0	3	

Cheking Skewness

In [23]:

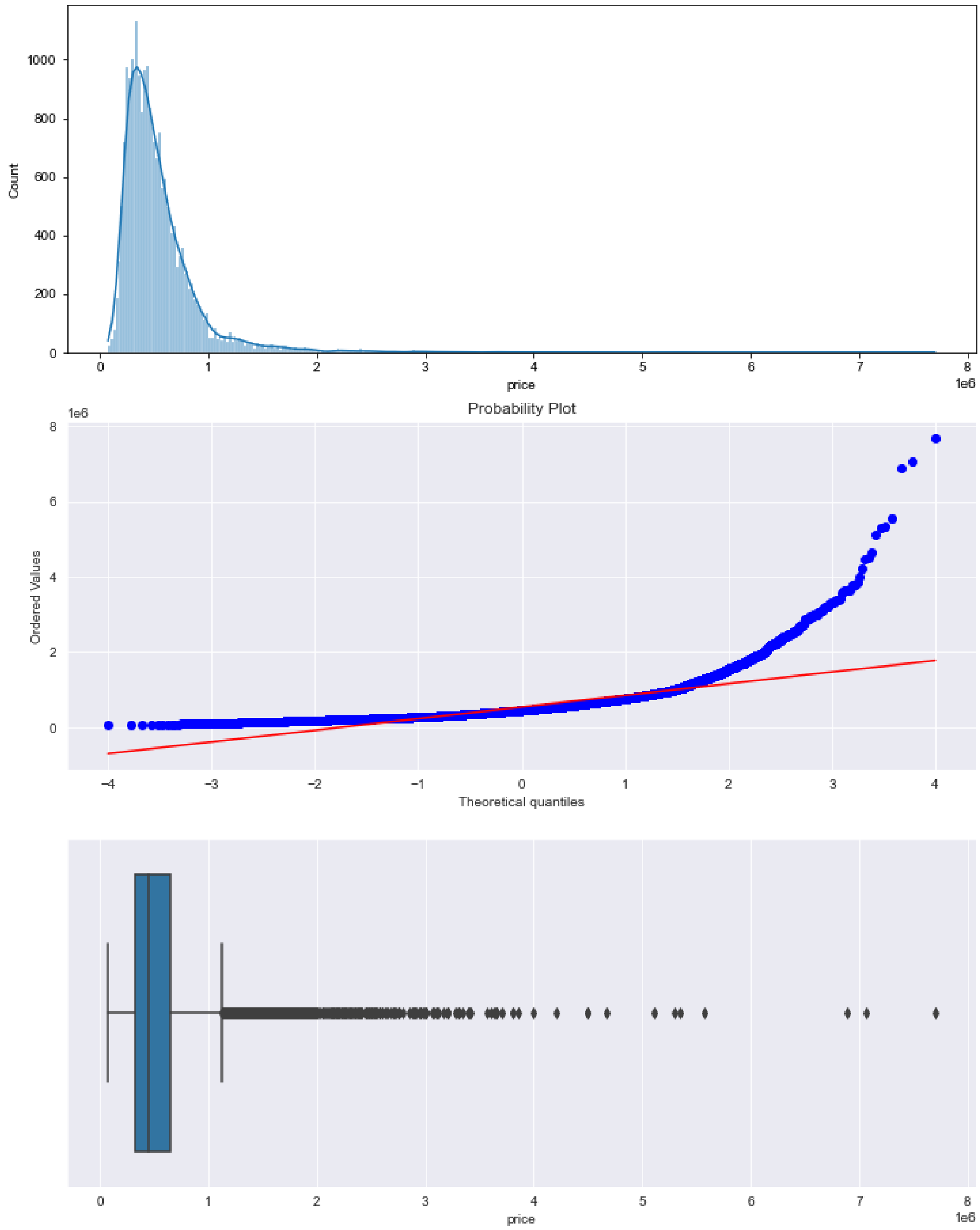
```python
#Checking Skewness
fig=plt.figure(figsize=(12,16))
fig.tight_layout()
ax1 = fig.add_subplot(3, 1, 1)
sns.set_style("darkgrid")
sns.histplot(df.loc[:, 'price'],kde=True,ax=ax1)

ax2 = fig.add_subplot(3, 1, 2)
stats.probplot(df.loc[:,'price'],plot=ax2)

ax3 = fig.add_subplot(3, 1, 3)
sns.boxplot(x=df.loc[:, 'price'],ax=ax3)
```

Out[23]:

```
<AxesSubplot:xlabel='price'>
```



In [24]:

```
df['price']=boxcox(df['price'])[0]
```

In [25]:

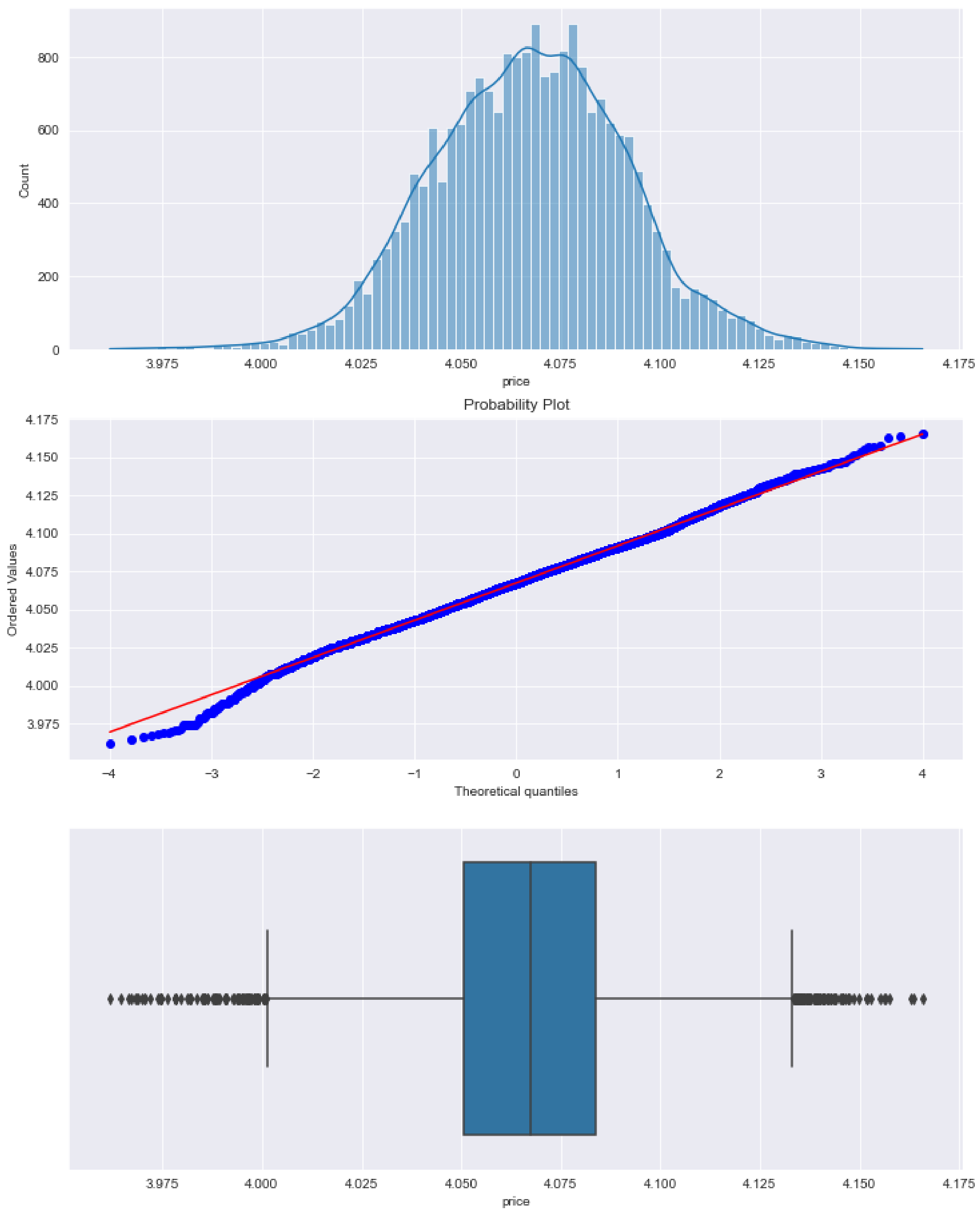
```
#Checking Skewness
fig=plt.figure(figsize=(12,16))
fig.tight_layout()
ax1 = fig.add_subplot(3, 1, 1)
sns.set_style("darkgrid")
sns.histplot(df.loc[:, 'price'],kde=True,ax=ax1)

ax2 = fig.add_subplot(3, 1, 2)
stats.probplot(df.loc[:,'price'],plot=ax2)

ax3 = fig.add_subplot(3, 1, 3)
sns.boxplot(x=df.loc[:, 'price'],ax=ax3)
```

Out[25]:

<AxesSubplot:xlabel='price'>



In [26]:

```
df.shape
```

Out[26]:

```
(21595, 17)
```

In [27]:

```
i=0
while i<df.shape[1]:
    x=df.columns[i]
    if df[x].dtypes!=object and abs(df[x]).skew() >0.5 and i!=0:
        df[x]=stats.boxcox(df[x]+1)[0]
    i+=1
```

In [28]:

```
df
```

Out[28]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	cond
0	4.030919	1.845296	0.790440	7.814065	4.195162	0.433862	0.0	0.0	0.80
1	4.075683	1.845296	1.478450	8.770486	4.240651	0.543474	0.0	0.0	0.80
2	4.018914	1.375287	0.790440	7.298470	4.296609	0.433862	0.0	0.0	0.80
3	4.080876	2.248442	1.812704	8.435165	4.171951	0.433862	0.0	0.0	0.91
4	4.073237	1.845296	1.356321	8.245539	4.260032	0.433862	0.0	0.0	0.80
...	...	...	...	...	...	...	...	...	
21608	4.056522	1.845296	1.594803	8.130891	3.841087	0.589749	0.0	0.0	0.80
21609	4.061723	2.248442	1.594803	8.638189	4.200485	0.543474	0.0	0.0	0.80
21610	4.061978	1.375287	0.621954	7.637361	3.885643	0.543474	0.0	0.0	0.80
21611	4.061723	1.845296	1.594803	8.185693	4.019296	0.543474	0.0	0.0	0.80
21612	4.051349	1.375287	0.621954	7.637361	3.828261	0.543474	0.0	0.0	0.80

21595 rows × 17 columns

Scaling

In [29]:

```
df_scaled=df
from sklearn.preprocessing import StandardScaler
#scaled=RobustScaler()
scaled = StandardScaler()

df_scaled = pd.DataFrame(scaled.fit_transform(df),columns=df.columns)

df_scaled.head()
```

Out[29]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	co
0	-1.487163	-0.352330	-1.571621	-1.121672	-0.332344	-0.975183	-0.087209	-0.329935	-0.
1	0.341151	-0.352330	0.251450	0.706994	-0.032751	1.030360	-0.087209	-0.329935	-0.
2	-1.977451	-1.606097	-1.571621	-2.107485	0.335789	-0.975183	-0.087209	-0.329935	-0.
3	0.553232	0.723079	1.137145	0.065865	-0.485213	-0.975183	-0.087209	-0.329935	2.
4	0.241236	-0.352330	-0.072163	-0.296698	0.094890	-0.975183	-0.087209	-0.329935	-0.

Dummies

In [30]:

```
features = pd.get_dummies(df_scaled).reset_index(drop=True)
features.shape
```

Out[30]:

```
(21595, 17)
```

In [31]:

```
X = df_scaled.drop(['price'],axis = 1)
X.head()
```

Out[31]:

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	
0	-0.352330	-1.571621	-1.121672	-0.332344	-0.975183	-0.087209	-0.329935	-0.598191	-0.
1	-0.352330	0.251450	0.706994	-0.032751	1.030360	-0.087209	-0.329935	-0.598191	-0.
2	-1.606097	-1.571621	-2.107485	0.335789	-0.975183	-0.087209	-0.329935	-0.598191	-1.
3	0.723079	1.137145	0.065865	-0.485213	-0.975183	-0.087209	-0.329935	2.080336	-0.
4	-0.352330	-0.072163	-0.296698	0.094890	-0.975183	-0.087209	-0.329935	-0.598191	0.

In [32]:

```
y = df_scaled['price']
y.head()
```

Out[32]:

```
0   -1.487163
1    0.341151
2   -1.977451
3    0.553232
4    0.241236
Name: price, dtype: float64
```

In [33]:

```
rms=[]
mse=[]
mae=[]
r2=[]

def evaluation(model, x_test, y_test):
        pred = model.predict(x_test)

        rms.append(np.sqrt(mean_squared_error(y_test, pred)))

        mse.append(mean_squared_error(y_test, pred))

        mae.append(mean_absolute_error(y_test, pred))

        r2.append(r2_score(y_test,pred))

        print("Result\n","---------------------------------------------")
        print("\nRoot Mean Squared Error: ",np.sqrt(mean_squared_error(y_test, pred)))
        print("\nMean Squared Error: \t",mean_squared_error(y_test, pred))
        print("\nMean Absolute Error: \t",mean_absolute_error(y_test, pred))
        print("\nR2 Score: \t\t",r2_score(y_test,pred))
```

Spliting data for training and testing

In [34]:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

Linear Regression

In [35]:

```
lg  = LinearRegression()

model=lg.fit(X_train,y_train)

y_pred_lg = lg.predict(X_test)

y_pred_lg
```

Out[35]:

```
array([-1.34381624, -0.15454372, -0.02232806, ..., -0.0058335 ,
       -0.83150375, -0.2049119 ])
```

In [36]:

```
evaluation(model, X_test, y_test)
```

```
Result
 ---------------------------------------------

Root Mean Squared Error:  0.5107123586919792

Mean Squared Error:      0.26082711332072483

Mean Absolute Error:     0.391958205061279

R2 Score:                0.7393462089171894
```

KNN

In [37]:

```

kr = KNeighborsRegressor(n_neighbors=10)
model=kr.fit(X_train,y_train)
```

In [38]:

```
evaluation(model, X_test, y_test)
```

```
Result
 ---------------------------------------------

Root Mean Squared Error:  0.42202253003026213

Mean Squared Error:      0.1781030158531435

Mean Absolute Error:     0.30242172317552296

R2 Score:                0.8220153353906897
```

SVR

In [39]:

```
sv = SVR(kernel = 'rbf')
model=sv.fit(X_train, y_train)
```

In [40]:

```
evaluation(model, X_test, y_test)
```

```
Result
 ---------------------------------------------

Root Mean Squared Error:  0.38780732856598055

Mean Squared Error:      0.1503945240894824

Mean Absolute Error:     0.2730300617046993

R2 Score:                0.8497054145831248
```

XGB

In [41]:

```
xgb_regress = xgb.XGBRegressor(n_estimators = 2000, learning_rate = 0.1)
l=xgb_regress.fit(X_train, y_train)

y_pred_xg = l.predict(X_test)
accuracy = round(r2_score(y_test,y_pred_xg),3)
accuracy
```

Out[41]:

```
0.879
```

In [42]:

```
evaluation(l, X_test, y_test)
```

```
Result
 ---------------------------------------------

Root Mean Squared Error:  0.3481417810946497

Mean Squared Error:      0.12120269974375499

Mean Absolute Error:     0.23691109537962213

R2 Score:                0.878877840668219
```

Gradient Boosting

In [43]:

```
params = {
    "n_estimators": 1000,
    "max_depth": 4,
    "min_samples_split": 5,
    "learning_rate": 0.1
}

reg = ensemble.GradientBoostingRegressor(**params)
l=reg.fit(X_train, y_train)
```

In [44]:

```
evaluation(l, X_test, y_test)
```

```
Result
 ---------------------------------------------

Root Mean Squared Error:  0.3397205518739977

Mean Squared Error:      0.11541005336557356

Mean Absolute Error:     0.23384886750326775

R2 Score:                0.8846666377746705
```

ADABOOST

In [45]:

```
regr = ensemble.AdaBoostRegressor(random_state=0, n_estimators=1000)
l=regr.fit(X_train, y_train)
```

In [46]:

```
evaluation(l, X_test, y_test)
```

```
Result
 ---------------------------------------------

Root Mean Squared Error:  0.4985404861954686

Mean Squared Error:      0.24854261637601419

Mean Absolute Error:     0.38537733668965807

R2 Score:                0.7516225426902305
```

Random Forest

In [47]:

```
regressor = RandomForestRegressor(n_estimators = 100, random_state = 0)

# fit the regressor with x and y data
l=regressor.fit(X_train, y_train)
```

In [48]:

```
evaluation(l, X_test, y_test)
```

```
Result
 ---------------------------------------------

Root Mean Squared Error:  0.36239858625509846

Mean Squared Error:      0.13133273531969406

Mean Absolute Error:     0.24957256832632102

R2 Score:                0.8687545366027192
```

CAT Boost

In [49]:

```
model = cb.CatBoostRegressor(iterations=1000,
                          learning_rate=0.1,
                          depth=8)
# Fit model
cat=model.fit(X_train, y_train)
```

```
0:      learn: 0.9314973        total: 156ms    remaining: 2m 35s
1:      learn: 0.8703926        total: 164ms    remaining: 1m 22s
2:      learn: 0.8132449        total: 173ms    remaining: 57.3s
3:      learn: 0.7617152        total: 181ms    remaining: 45.1s
4:      learn: 0.7153295        total: 189ms    remaining: 37.7s
5:      learn: 0.6777702        total: 197ms    remaining: 32.7s
6:      learn: 0.6426906        total: 205ms    remaining: 29.1s
7:      learn: 0.6101366        total: 213ms    remaining: 26.5s
8:      learn: 0.5829402        total: 221ms    remaining: 24.3s
9:      learn: 0.5584274        total: 229ms    remaining: 22.7s
10:     learn: 0.5363441        total: 237ms    remaining: 21.3s
11:     learn: 0.5171809        total: 246ms    remaining: 20.3s
12:     learn: 0.4997535        total: 255ms    remaining: 19.4s
13:     learn: 0.4853623        total: 264ms    remaining: 18.6s
14:     learn: 0.4692499        total: 272ms    remaining: 17.9s
15:     learn: 0.4555180        total: 280ms    remaining: 17.2s
16:     learn: 0.4451452        total: 288ms    remaining: 16.6s
17:     learn: 0.4350465        total: 296ms    remaining: 16.2s
18:     learn: 0.4267540        total: 304ms    remaining: 15.7s
```

In [50]:

```
evaluation(cat, X_test, y_test)
```

```
Result
 ---------------------------------------------

Root Mean Squared Error:  0.33141486335313103

Mean Squared Error:      0.1098358116513745

Mean Absolute Error:     0.22662380597553808

R2 Score:                0.8902371753492335
```

Results

In [51]:

```
res=pd.DataFrame()
res['Model']=["LR","KNN","SVM","XGB","GRA","ADA","RF","CAT"]
res['RMSE']=rms
res['MSE']=mse
res['MAE']=mae
res['R2']=r2
```

In [52]:

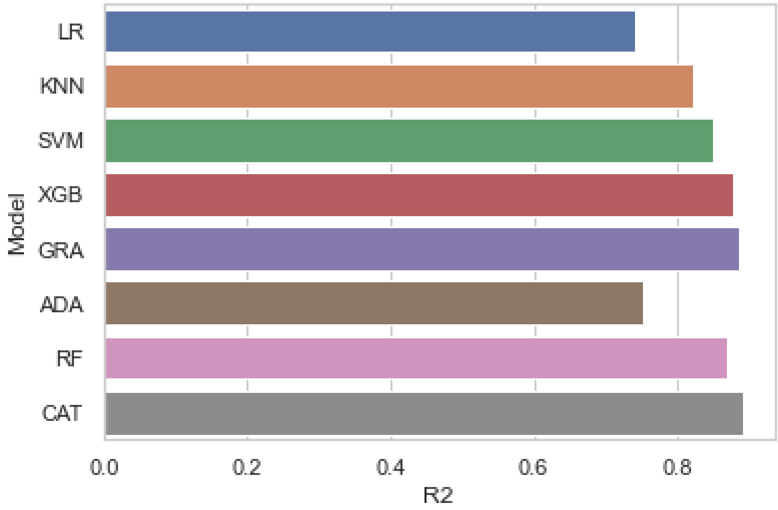
```
res
```

Out[52]:

	Model	RMSE	MSE	MAE	R2
0	LR	0.510712	0.260827	0.391958	0.739346
1	KNN	0.422023	0.178103	0.302422	0.822015
2	SVM	0.387807	0.150395	0.273030	0.849705
3	XGB	0.348142	0.121203	0.236911	0.878878
4	GRA	0.339721	0.115410	0.233849	0.884667
5	ADA	0.498540	0.248543	0.385377	0.751623
6	RF	0.362399	0.131333	0.249573	0.868755
7	CAT	0.331415	0.109836	0.226624	0.890237

In [53]:

```
sns.set(style="whitegrid")
ax=sns.barplot(y='Model',x='R2',data=res)
```



In [ ]:

```
```