Final Project Submission

Authors

> Abdikarim Gedi
>
> Edwin Nderitu
>
> Eston Kamau
>
> Susan Warigia
>
> William Onsare
>
> Daniel Ekale

- Scheduled project review date: 17/04/2023 - 21/04/2023
- Instructor name: Antonny Muiko

TItle: Predicting Home Sale Prices in King County.

Explanation

> - This project aims to predict the sale price of homes in King County, based on various features such as the number of bedrooms, bathrooms, square footage, and location. The goal is to provide advice to homeowners about how home renovations might increase the estimated value of their homes, and by what amount.

Overview

This projects aims to analyze factors affecting house price in King County. Some of these factors are; location, year of construction, size, renovations done and many more. This would greatly help the agency get insights on how to cut down the costs and maximize the profit.

Business Problem

Our stakeholder is homeowners who are looking to renovate their homes and want to estimate the impact of these renovations on the value of their home. Our business problem is to identify which home features are most important in determining a home's sale price and estimate how much value can be added by improving these features.

> **Objectives**
>
> > - Identify the most significant home features that contribute to a home's sale price.
> > - Estimate the value added to a home by improving these significant features.
> > - Develop a methodology for accurately estimating the impact of home renovations on home value.

> - Provide homeowners with an easy-to-use tool for estimating the value added by renovating specific home features.
> - Provide actionable recommendations to homeowners looking to renovate their homes to maximize their home value.

In [ ]:
```
# Import the required libraries for analysis
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.formula.api as smf
import seaborn as sns
plt.style.use('ggplot')
```

Load the dataset into a data frame

In [ ]:
```
# load the data into a data frame
df = pd.read_csv('data/kc_house_data.csv')
df.head()
```

Out[ ]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_bas
0	7129300520	10/13/2014	221900.0	3	1.00	1180	5650	1.0	NaN	NONE	...	7 Average	1180	
1	6414100192	12/9/2014	538000.0	3	2.25	2570	7242	2.0	NO	NONE	...	7 Average	2170	
2	5631500400	2/25/2015	180000.0	2	1.00	770	10000	1.0	NO	NONE	...	6 Low Average	770	
3	2487200875	12/9/2014	604000.0	4	3.00	1960	5000	1.0	NO	NONE	...	7 Average	1050	
4	1954400510	2/18/2015	510000.0	3	2.00	1680	8080	1.0	NO	NONE	...	8 Good	1680	

5 rows × 21 columns

In [ ]:
```
rows, columns = df.shape
print(f'Nrows: {rows}, Ncolumns: {columns}')
```

```
Nrows: 21597, Ncolumns: 21
```

In [ ]:
```
cols = df.columns
cols
```

Out[ ]:
```
Index(['id', 'date', 'price', 'bedrooms', 'bathrooms', 'sqft_living',
       'sqft_lot', 'floors', 'waterfront', 'view', 'condition', 'grade',
       'sqft_above', 'sqft_basement', 'yr_built', 'yr_renovated', 'zipcode',
       'lat', 'long', 'sqft_living15', 'sqft_lot15'],
      dtype='object')
```

Data Understanding

The data used in this project is the King County House Sales dataset, which can be found in kc_house_data.csv in the data folder. The dataset contains 21,597 records and 21 columns. The description of the column names can be found in column_names.md in the same folder. It contains 21 columns with each being either numerical or categorical data.

Data processing

We will start by cleaning the data and handling missing values, if any. We will also convert some of the columns into the appropriate data type.

In [ ]:
```
def missing_values(data):

    # Columns with null values
    null_cols = data.columns[data.isna().any()]

    # Count null values in each column and sort in descending order
    count_null = data[null_cols].isna().sum().sort_values(ascending=False)

    # print the null column and the count
    for col, count in zip(null_cols, count_null):
        print(f'The {col} has {round(count/len(data)*100, 2)} % of the data missing')

missing_values(df)
```

```
The waterfront has 17.79 % of the data missing
The view has 11.0 % of the data missing
The yr_renovated has 0.29 % of the data missing
```

In [ ]:
```
def duplicates(data=df, unique_id = 'id'):
    # Check for duplicates
    duplicates = data[unique_id].duplicated().sum()
    # duplicates = df.id.duplicated().sum()
    print(f'The duplicated values are: {duplicates}')
    print(f'Which translates into {round(duplicates/len(data) * 100, 2)}% of the total data.')

    # Drop duplicated values:
    data.drop_duplicates(subset=unique_id, keep='last', inplace=True)

duplicates(df, 'id')
```

```
The duplicated values are: 177
Which translates into 0.82% of the total data.
```

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21420 entries, 0 to 21596
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   id             21420 non-null  int64  
 1   date           21420 non-null  object 
 2   price          21420 non-null  float64
 3   bedrooms       21420 non-null  int64  
 4   bathrooms      21420 non-null  float64
 5   sqft_living    21420 non-null  int64  
 6   sqft_lot       21420 non-null  int64  
 7   floors         21420 non-null  float64
 8   waterfront     19067 non-null  object 
 9   view           21357 non-null  object 
 10  condition      21420 non-null  object 
 11  grade          21420 non-null  object 
 12  sqft_above     21420 non-null  int64  
 13  sqft_basement  21420 non-null  object 
 14  yr_built       21420 non-null  int64  
 15  yr_renovated   17607 non-null  float64
 16  zipcode        21420 non-null  int64  
 17  lat            21420 non-null  float64
 18  long           21420 non-null  float64
 19  sqft_living15  21420 non-null  int64  
 20  sqft_lot15     21420 non-null  int64  
dtypes: float64(6), int64(9), object(6)
memory usage: 3.6+ MB
```

Dealing with numerical variables.

- We convert the date columns to dates
- we convert sqft_basement to float

Dealing with the date columns

- To be able to use the date columns we have to convert them into either and int or a datetime data type. The `date` column has been converted into `datetime` using the `pd.datetime()` function. The `yr_renovated` column which might be of help has been converted to datatype `int`

In [ ]:
```
df['date'] = pd.to_datetime(df['date']).dt.year
df['yr_renovated'] = df['yr_renovated'].fillna(0).astype(int)
```

Numerical Variables

In [ ]:
```
# The column square_basement is in object form. So let's conver it to int
df['sqft_basement'] = df['sqft_basement'].replace("?", '0').astype(float)
df_numeric = df.select_dtypes(include=np.number)
df_numeric.head(3)
```

Out[ ]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	sqft_above	sqft_basement	yr_built	yr_renovated	zipcode
0	7129300520	2014	221900.0	3	1.00	1180	5650	1.0	1180	0.0	1955	0	98178
1	6414100192	2014	538000.0	3	2.25	2570	7242	2.0	2170	400.0	1951	1991	98125
2	5631500400	2015	180000.0	2	1.00	770	10000	1.0	770	0.0	1933	0	98028

Categorical Variables

In [ ]:
```
df_categorical = df.select_dtypes(include=np.object_)
df_categorical
```

Out[ ]:

	waterfront	view	condition	grade
0	NaN	NONE	Average	7 Average
1	NO	NONE	Average	7 Average
2	NO	NONE	Average	6 Low Average
3	NO	NONE	Very Good	7 Average
4	NO	NONE	Average	8 Good
...	...	...	...	...
21592	NO	NONE	Average	8 Good
21593	NO	NONE	Average	8 Good
21594	NO	NONE	Average	7 Average
21595	NaN	NONE	Average	8 Good
21596	NO	NONE	Average	7 Average

21420 rows × 4 columns

Data Analysis

We can start by having a look at the summary statistics of all the numerical variables. The `describe()` function in pandas enables us implement this concent easily.

In [ ]:
```
df.drop('id', axis=1).describe()
```

Out[ ]:

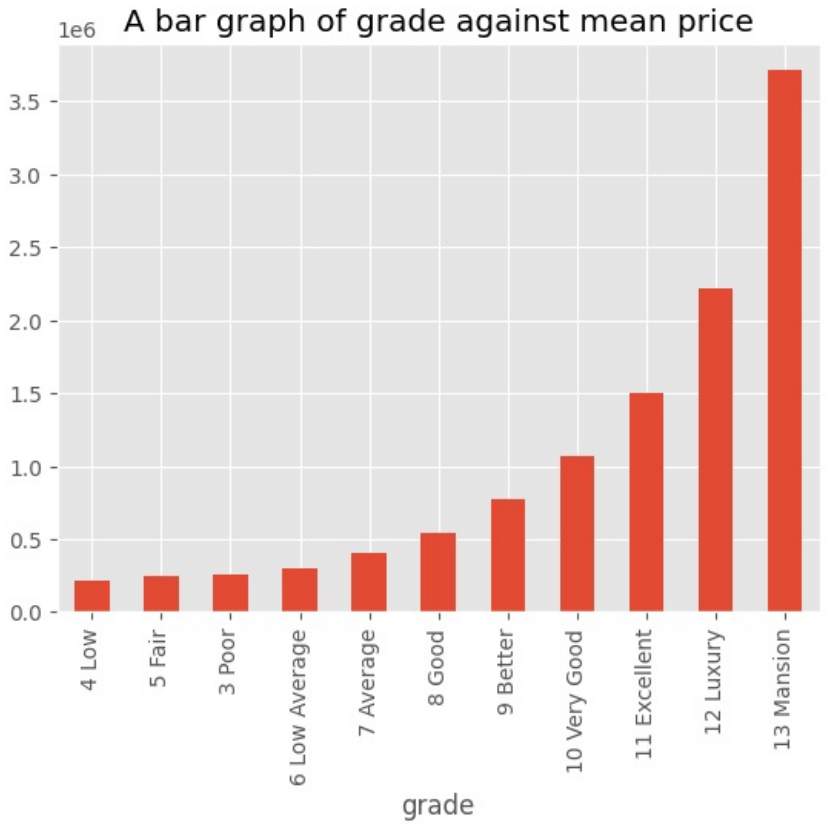
	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	sqft_above	sqft_basement	
count	21420.000000	2.142000e+04	21420.000000	21420.000000	21420.000000	2.142000e+04	21420.000000	21420.000000	21420.000000	214
mean	2014.325257	5.418614e+05	3.373950	2.118429	2083.132633	1.512804e+04	1.495985	1791.170215	285.937021	19
std	0.468482	3.675569e+05	0.925405	0.768720	918.808412	4.153080e+04	0.540081	828.692965	440.012962	
min	2014.000000	7.800000e+04	1.000000	0.500000	370.000000	5.200000e+02	1.000000	370.000000	0.000000	19
25%	2014.000000	3.249500e+05	3.000000	1.750000	1430.000000	5.040000e+03	1.000000	1200.000000	0.000000	19
50%	2014.000000	4.505500e+05	3.000000	2.250000	1920.000000	7.614000e+03	1.500000	1560.000000	0.000000	19
75%	2015.000000	6.450000e+05	4.000000	2.500000	2550.000000	1.069050e+04	2.000000	2220.000000	550.000000	19
max	2015.000000	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	3.500000	9410.000000	4820.000000	20

Interpretation

- From the table we can see the measures of central tendency like the mean, mode, mdeian and quartiles of various variables. We can also see the variability of the data by observing the standard deviations and the range.
- The mean for `price` variable is 5.418614e+05, the minimum being 7.8e+04 and the maximum price is 7.7e+06.

Mean price of houses in respect to grade

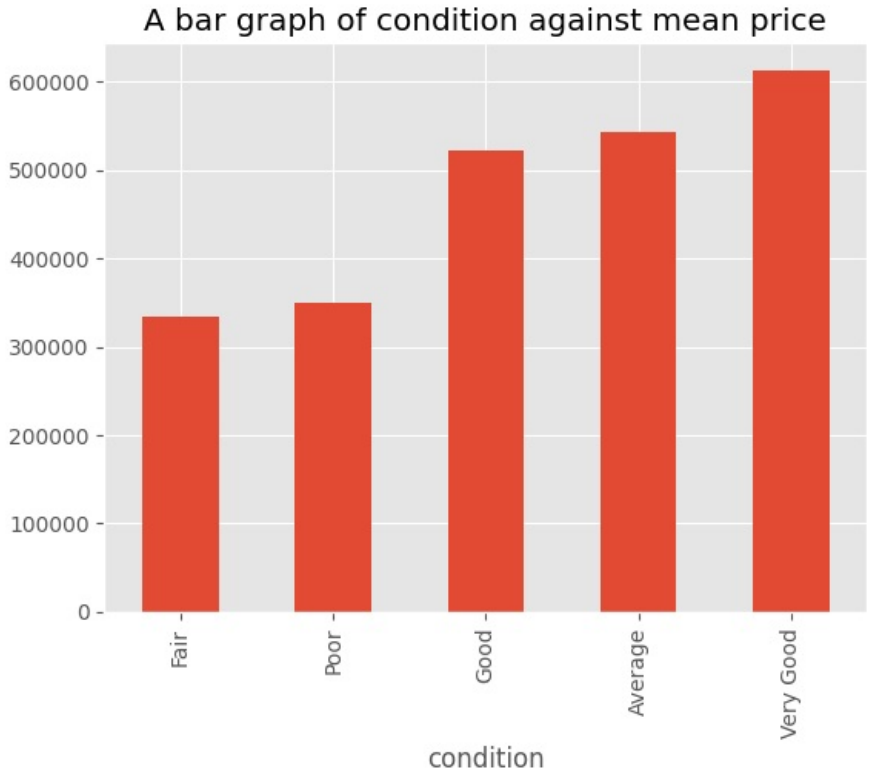
In [ ]:
```
def plot_variable(column):
    grp = df.groupby(column)['price'].mean().sort_values(ascending=True)
    grp.plot(kind='bar', title = 'A bar graph of ' + column + ' against mean price')

plot_variable('grade')
```



- We can see that average prices for the houses differ depending on grade. In the above plot we can see that houses that had the highest average prices are Average, Good, Better in that order. The least being the one with poor grade.

How does condition of a house affect house price?

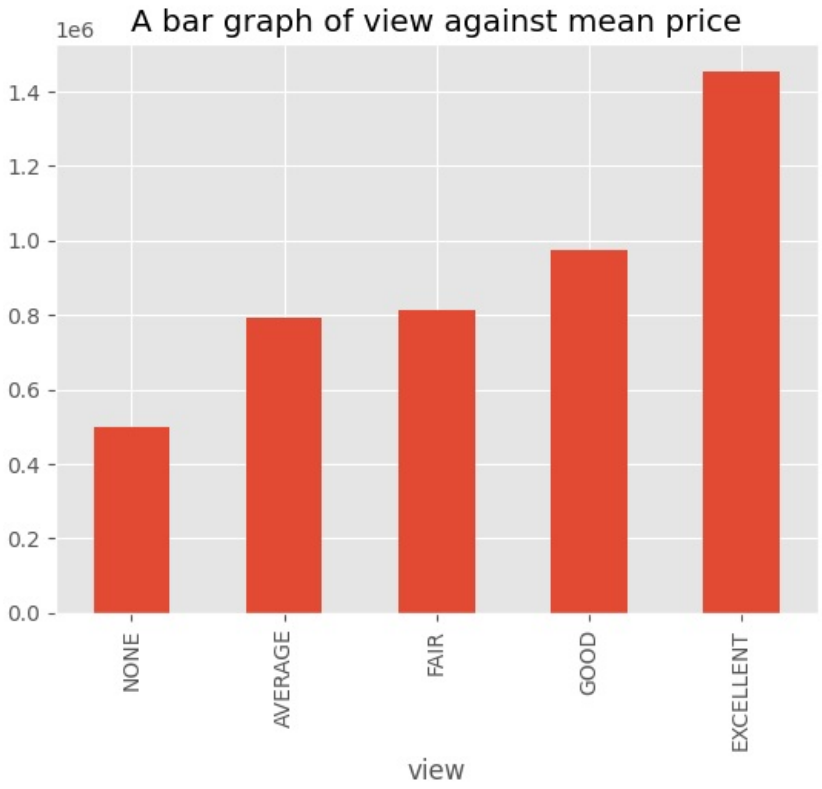
In [ ]:
```
plot_variable('condition')
```



- From the analysis, it is clear that the condition of the house affects the price. Improving the condition of the house can increase its value and attract potential buyers.

How view affects house price?

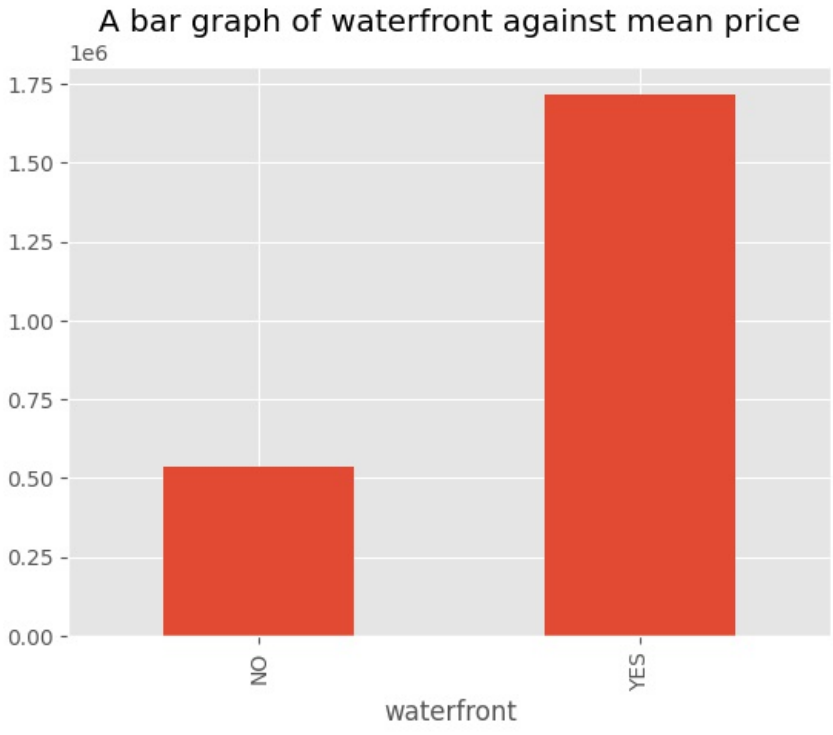
In [ ]:
```
plot_variable('view')
```



- The analysis shows that houses with excellent views had the highest average sale price. Enhancing the view by landscaping or adding features such as balcony can increase the value of the house.

How does waterfront affect house price?

In [ ]:
```
plot_variable('waterfront')
```



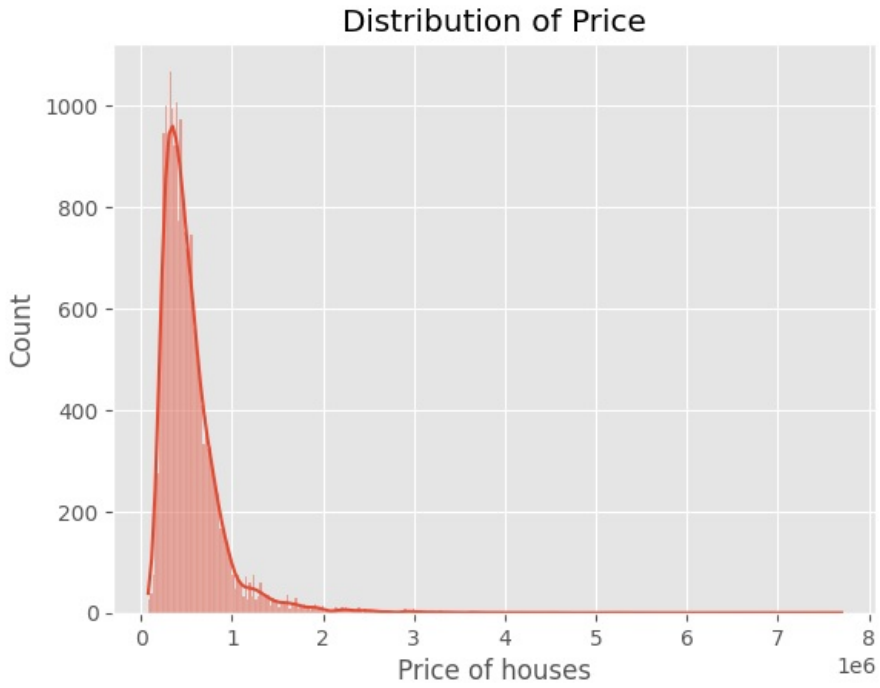
- We can clearly see that, houses with a waterfront have the highest mean prices. This means that the home owners should consider building their houses next to a waterfront to attract prospective buyers.

Distribution analysis and plots

To be able to understand our data well and build reliable models, we need to study the distribution patterns of different variables in our dataset. This will enable us identify the most appropriate tests to use to accurately make informed decsisions.

In [ ]:
```
#plot the distribution of the target variable
sns.histplot(df.price, kde=True)
```

```
plt.title('Distribution of Price')
plt.xlabel('Price of houses')
plt.ylabel('Count')
plt.show()
```



Interpretions

- The price distribution is positively skewed, which means there are more houses with lower price than higher price.

Skewness

In [ ]:
```
sk = df.select_dtypes(include=np.number).skew().sort_values(ascending=False)
sk_res = pd.DataFrame({'Columns':sk.index, 'skewness':sk.values})
for col, value in zip(sk_res['Columns'],sk_res['skewness']):
    if (value <= -1) or (value >= 1):
        print('The column below is highly skewed.')
        print(f'{col} : {value}')
        print()
    if (-0.5 > value > -1) or (0.5 < value < 1):
        print('The column below is moderately skewed.')
        print(f'{col} : {value}')
        print()
    if (0.5 > value > -0.5):
        print('The column below is approximately symmetric.')
        print(f'{col} : {value}')
        print()
```

```
The column below is highly skewed.
sqft_lot : 13.056251852883626

The column below is highly skewed.
sqft_lot15 : 9.513044564095368

The column below is highly skewed.
yr_renovated : 5.098099221916137

The column below is highly skewed.
price : 4.035378874466087

The column below is highly skewed.
bedrooms : 2.0399818162187944

The column below is highly skewed.
sqft_basement : 1.6041626954887744

The column below is highly skewed.
sqft_living : 1.4727019951483347

The column below is highly skewed.
sqft_above : 1.4450007970330156

The column below is highly skewed.
sqft_living15 : 1.1045308883004101

The column below is moderately skewed.
long : 0.8817548805770775

The column below is moderately skewed.
date : 0.7460678587456605

The column below is moderately skewed.
floors : 0.6087875250982765

The column below is moderately skewed.
bathrooms : 0.5188980554943093

The column below is approximately symmetric.
zipcode : 0.4077889666043932

The column below is approximately symmetric.
id : 0.24312735775575572

The column below is approximately symmetric.
yr_built : -0.47422199328416514

The column below is approximately symmetric.
lat : -0.4884209399750156
```

- columns like `sqft_lot, sqft_lot15, yr_renovated, price, and bedrooms` have high skewness. This indicates that they have a few extreme values that are far from the mean. On the other hand, some columns like `zipcode, id, yr_built, and lat` are approximately symmetric, indicating that they are normally distributed.

Dealing with outliers

In [ ]:
```python
def remove_outliers(df):
    """
    Removes outliers from a pandas DataFrame using the z-score method
    with a threshold of two standard deviations.

    """
    # Calculate the z-scores for all numerical columns
    z_scores = df.select_dtypes(include=[np.number]).apply(lambda x: np.abs((x - x.mean()) / x.std()))

    # Create a boolean mask for values that are within two standard deviations
    mask = (z_scores < 2).all(axis=1)

    # Return the DataFrame with outliers removed
    return df.loc[mask]
df = remove_outliers(df)
```

In [ ]:
```python
# Transform the price variable to make it normal and visualizing the results
transformed = np.log(df['price'])
sns.histplot(transformed, kde=True)
plt.title('Distribution of Price_Transf')
plt.xlabel('Price_Transf of houses')
plt.ylabel('Count')
```

```
plt.show()
```

- This transformation is commonly used to normalize data that has a skewed distribution, making it easier to analyze using statistical methods.

Convert Categorical variables to numerics

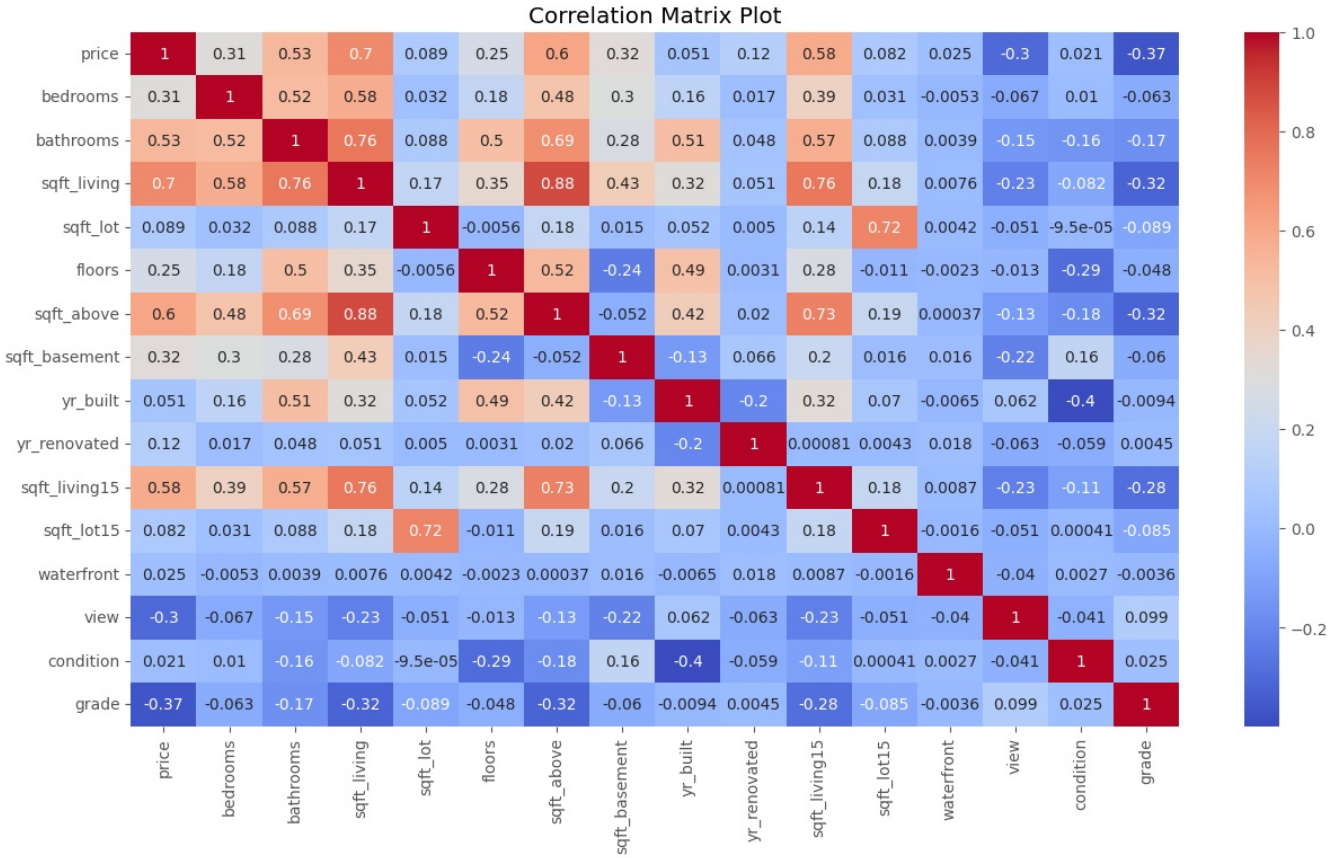
In order to incorporate the categorical variable in our analysis, we need to deal with them as numerica variables. This will enable us for example, visualize how the categorcal variables affect the price using the correlation matrix.

In [ ]:
```
# Convert categorical variables
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
data_cat_lbl = df_categorical.apply(le.fit_transform)
cat_converted = pd.DataFrame(data_cat_lbl)

#Replace NaN with 0
cat_converted = cat_converted.fillna(0)


#Join the numeric df and the categorical converted variables to main df
df = df_numeric.join(cat_converted, how='left')
```

Correlation Matrix Plot

In order to examine the correlation between the numeric variables and the target variable-price, we found it useful to display the information using a correlation heatmap.

In [ ]:
```
df_corr = df.drop(['date','id','zipcode', 'lat', 'long'], axis=1)
plt.figure(figsize=(15,8))
sns.heatmap(df_corr.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Matrix Plot')
plt.show()
```

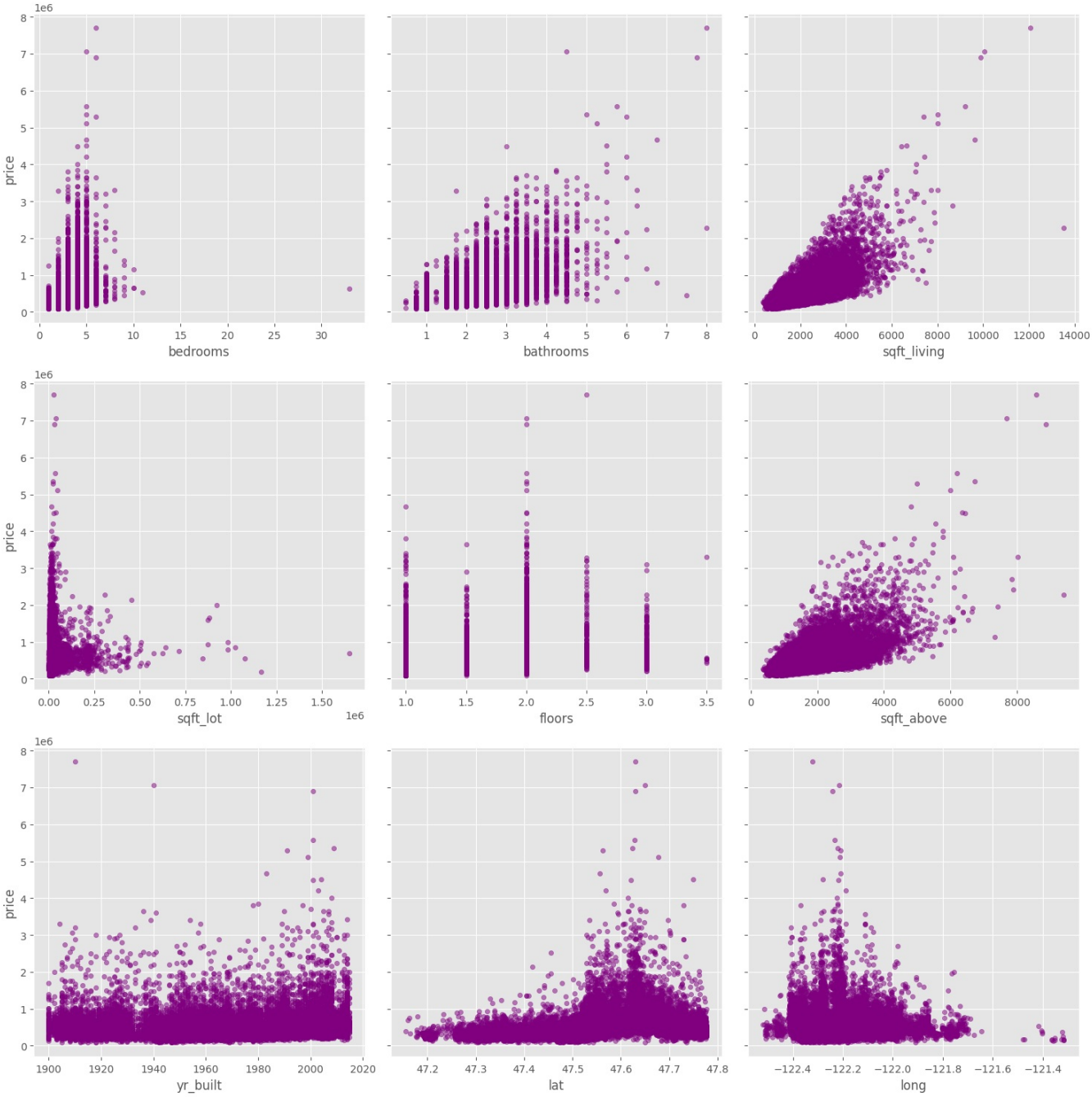


Interpretation

- From the above correlation matrix plot, We can see that numerical variables that have a higher correlation with price are; `sqft_living` , `sqft_above` , `sqft_living15` , `bathrooms` , and `bedrooms` .
- We can also see that the variables with a weak positive correletion to price are `sqft_lot` , `sqft_lot15` , and `yr_built` .

In [ ]:

```
# plotting multiple scatter plots of the various variables

fig = plt.figure(figsize=(18,10))

axes = fig.subplots(nrows=3, ncols=3, sharey=True)

# defining a function to plot multiple scatter plots of the various variables
def multiple_scatterplot(data, columns):
    for xcol, ax in zip(columns, axes.flatten()):
        data.plot(kind='scatter', x=xcol, y='price', ax=ax, alpha=0.5, color='purple', figsize=(15,15))

multiple_scatterplot(df, columns = ['bedrooms', 'bathrooms', 'sqft_living','sqft_lot', 'floors','sqft_above', '

plt.tight_layout()
plt.show()
```



Simple Linear Regression Model

We can perform a regression analysis for the variable that had the highest correlation.

In [ ]:
```
#function to perform regression
from sklearn.metrics import mean_squared_error
def model(data, y, x):
    '''This function takes three arguments and prints the regression summary table'''
    formula = y + '~' + '+'.join(x)
    results = smf.ols(formula, data).fit()
    print(results.summary())
    print()
    print(f'R_squared: {results.rsquared}')

    equation = f'y_pred = {results.params[0]}'
    for i in range(len(x)):
        equation += f' + {results.params[i+1]} * {x[i]}'
    print(equation)

    print()

    y_pred = results.predict(data[x])
    mse = mean_squared_error(data[y], y_pred)
    rmse = np.sqrt(mse)

    print(f'MSE: {mse}')
    print(f'RMSE: {rmse}')

model(df, 'price', ['sqft_living'])
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.492
Model:                            OLS   Adj. R-squared:                  0.492
Method:                 Least Squares   F-statistic:                 2.073e+04
Date:                Thu, 20 Apr 2023   Prob (F-statistic):               0.00
Time:                        21:59:31   Log-Likelihood:            -2.9763e+05
No. Observations:               21420   AIC:                         5.953e+05
Df Residuals:                   21418   BIC:                         5.953e+05
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept  -4.255e+04   4436.470     -9.591      0.000   -5.12e+04   -3.39e+04
sqft_living   280.5436      1.949    143.972      0.000     276.724     284.363
==============================================================================
Omnibus:                    14710.422   Durbin-Watson:                   1.989
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           541541.173
Skew:                           2.827   Prob(JB):                         0.00
Kurtosis:                      26.975   Cond. No.                     5.64e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.64e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

R_squared: 0.491814479424757
y_pred = -42548.03529381461 + 280.54356893661327 * sqft_living

MSE: 68651694402.30904
RMSE: 262014.68356240846
```

Interpretation

- R-squared indicates the proportion of the variance in the dependent variable `price` that is explained by the independent variable `sqft_living`. The R-squared value of 0.492 means that about 49.2% of the variation in house prices can be explained by their square footage of living space.
- Intercept: This is the estimated value of `price` when `sqft_living` is 0. In this case, the intercept is -42,550, which means that a house with 0 square footage of living space would be worth about $42,550, which does not make sense in reality.
- Coefficient of sqft_living: This is the slope of the regression line that represents the change in the `price` for a one-unit increase in the `sqft_living`. In this case, the coefficient is 280.54, which means that on average, the price of a house increases by $280.54 for every one additional square foot of living space.
- The standard error for the coefficient of sqft_living is 1.949, which means that the estimated value of the coefficient may be off by as much as $1.949.
- The p-value for the coefficient is 0.000, which means that we can be confident that the coefficient is statistically significant and that the relationship between square footage and price is not due to chance.
- The MSE value of 68651694402.30904 indicates that the model's predicted values differ from the actual values by approximately 68651694402.30904 squared units. The RMSE value of 262014.68356240846 means that the average difference between the model's predicted values and the actual values is approximately 262014.68356240846 units. Therefore, in this case, the MSE and RMSE values are quite high, indicating that the model may not be very accurate and may need further refinement.

Multiple Linear Regression Model

When performing regression analysis, we select predictor variables that are likely to have an impact on our target variable. To assess the significance of these variables, we can use a p-value approach, where any variable with a p-value greater than 0.05 is considered unimportant and can be removed from the model. Additionally, there are other methods such as Ridge, Lasso, or Elastic Net regression that penalize variables based on their importance in the model. The goal is to find the most important variables that have a significant impact on the target variable, while eliminating any unimportant variables that do not contribute to the model's accuracy.

In [ ]:
```
X = df.drop(['id','price','date', 'zipcode'], axis=1).columns
model(df, 'price', X)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.651
Model:                            OLS   Adj. R-squared:                  0.650
Method:                 Least Squares   F-statistic:                     2344.
Date:                Thu, 20 Apr 2023   Prob (F-statistic):               0.00
Time:                        21:59:31   Log-Likelihood:            -2.9362e+05
No. Observations:               21420   AIC:                         5.873e+05
Df Residuals:                   21402   BIC:                         5.874e+05
Df Model:                          17
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept  -5.316e+07   1.68e+06    -31.637      0.000   -5.65e+07   -4.99e+07
bedrooms   -4.379e+04   2058.753    -21.271      0.000   -4.78e+04   -3.98e+04
bathrooms    5.64e+04   3527.395     15.989      0.000    4.95e+04    6.33e+04
sqft_living   138.3427     19.537      7.081      0.000     100.050     176.636
sqft_lot        0.1458      0.052      2.817      0.005       0.044       0.247
floors       2.777e+04   3879.491      7.159      0.000    2.02e+04    3.54e+04
sqft_above     87.7229     19.553      4.486      0.000      49.397     126.049
sqft_basement  55.1605     19.376      2.847      0.004      17.183      93.138
yr_built    -1606.1324     76.390    -21.025      0.000   -1755.863   -1456.401
yr_renovated   48.4836      4.312     11.244      0.000      40.032      56.935
lat          6.162e+05   1.12e+04     55.017      0.000    5.94e+05    6.38e+05
long        -2.229e+05   1.27e+04    -17.483      0.000   -2.48e+05   -1.98e+05
sqft_living15  74.3028      3.543     20.973      0.000      67.359      81.247
sqft_lot15     -0.4070      0.079     -5.135      0.000      -0.562      -0.252
waterfront   6662.5081   2365.830      2.816      0.005    2025.304    1.13e+04
view        -4.217e+04   1708.719    -24.677      0.000   -4.55e+04   -3.88e+04
condition    1.935e+04   1323.420     14.624      0.000    1.68e+04    2.19e+04
grade       -2.022e+04    701.306    -28.826      0.000   -2.16e+04   -1.88e+04
==============================================================================
Omnibus:                    17518.333   Durbin-Watson:                   2.007
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          1371241.184
Skew:                           3.428   Prob(JB):                         0.00
Kurtosis:                      41.593   Cond. No.                     5.74e+07
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.74e+07. This might indicate that there are
strong multicollinearity or other numerical problems.

R_squared: 0.6505771292233424
y_pred = -53163091.67806296 + -43791.508783980826 * bedrooms + 56400.88088559122 * bathrooms + 138.342670652615
77 * sqft_living + 0.14577963679088865 * sqft_lot + 27773.31819567277 * floors + 87.72285064378974 * sqft_above
+ 55.16051045419367 * sqft_basement + -1606.1324220410552 * yr_built + 48.48360676817674 * yr_renovated + 61621
8.0887068984 * lat + -222873.4148521392 * long + 74.30282152032422 * sqft_living15 + -0.4069926848309239 * sqft
_lot15 + 6662.508060345417 * waterfront + -42166.09284938266 * view + 19353.535815018004 * condition + -20215.5
5410822212 * grade

MSE: 47204162988.71078
RMSE: 217265.19046711276
```

- From the above summary, R-squared value of 0.651 indicates that about 65.1% of the variation in the `price` variable is explained by the independent variables included in the model.
- Intercept value of -5.316e+07 represents the estimated price when all the independent variables are equal to zero, which is not necessarily meaningful in this context.
- The coefficient of 138.34 for `sqft_living` indicates that each extra square foot of living space is associated with an average increase in `price` of `$138.34`, all other variables remaining constant.
- `The mean squared error (MSE)` has `decreased` from the previous value of `68,651,694,402.31` to `47,204,162,988.71` and the `root mean squared error (RMSE)` has also `decreased` from `262,014.68` to `217,265.19`. This indicates that the model has been improved and is better able to predict the price of a house based on its features.

Conclusion

In conclusion, we have performed an analysis of a dataset on housing prices in King County, and made several recommendations to our agency based on our findings. We found that property size, number of bathrooms and bedrooms, condition and grade, view, and location all have significant correlations with housing prices. Home owners can increase the value of their property by increasing the square footage, adding an extra bathroom or bedroom, upgrading the condition or grade of their property, and considering the location and view. These recommendations can help home owners to maximize the value of their property when putting it on the market. However, it is important to keep in mind that these recommendations may not apply universally and may depend on various factors such as the local real estate market and the specific characteristics of the property.

Recommendations

Recommendations

Based on the analysis we performed, we can make the following recommendations to our agency:

- Property size matters: Home owners should consider increasing the square footage of the houses before putting it on the market.
- Bathrooms and bedrooms add value: Bathrooms and bedrooms have moderate positive correlations with the price. Therefore, home owners should consider adding an extra bathroom or bedroom to increase the value of their property.
- Condition and grade matter: The condition and grade of a property have a strong negative correlation with the price. Therefore, a home owner consider upgrading the condition or grade of their property to increase the value of the house.
- View can affect the price: The view has a strong negative correlation with the price, which suggests that properties with a better view may be priced lower. However, this may also depend on other factors such as the size and condition of the property.
- Location matters: The latitude and longitude (lat and long) have moderate positive and negative correlations with the price, respectively. a home owner should consider the location of the property as it may affect its value. In some countries, houses that may fall within the tropics may attract more buyers that those on the furthest end of the tropics.