In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
sns.set()

import warnings
warnings.filterwarnings('ignore')
```

In [67]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,r2_score
```

In [221]:

```
df = pd.read_csv("melb_data.csv")
```

In [135]:

```
df.head()
```

Out[135]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Postc
0	Abbotsford	85 Turner St	2	h	1480000.0	S	Biggin	3/12/2016	2.5	30
1	Abbotsford	25 Bloomburg St	2	h	1035000.0	S	Biggin	4/02/2016	2.5	30
2	Abbotsford	5 Charles St	3	h	1465000.0	SP	Biggin	4/03/2017	2.5	30
3	Abbotsford	40 Federation La	3	h	850000.0	PI	Biggin	4/03/2017	2.5	30
4	Abbotsford	55a Park St	4	h	1600000.0	VB	Nelson	4/06/2016	2.5	30

5 rows × 21 columns

In [70]:

```
df.describe(include="all")
```

Out[70]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	
count	13580	13580	13580.000000	13580	1.358000e+04	13580	13580	13580	1:
unique	314	13378	NaN	3	NaN	5	268	58	
top	Reservoir	28 Blair St	NaN	h	NaN	S	Nelson	27/05/2017	
freq	359	3	NaN	9449	NaN	9022	1565	473	
mean	NaN	NaN	2.937997	NaN	1.075684e+06	NaN	NaN	NaN	
std	NaN	NaN	0.955748	NaN	6.393107e+05	NaN	NaN	NaN	
min	NaN	NaN	1.000000	NaN	8.500000e+04	NaN	NaN	NaN	
25%	NaN	NaN	2.000000	NaN	6.500000e+05	NaN	NaN	NaN	
50%	NaN	NaN	3.000000	NaN	9.030000e+05	NaN	NaN	NaN	
75%	NaN	NaN	3.000000	NaN	1.330000e+06	NaN	NaN	NaN	
max	NaN	NaN	10.000000	NaN	9.000000e+06	NaN	NaN	NaN	

11 rows × 21 columns

In [71]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13580 entries, 0 to 13579
Data columns (total 21 columns):
Suburb           13580 non-null object
Address          13580 non-null object
Rooms            13580 non-null int64
Type             13580 non-null object
Price            13580 non-null float64
Method           13580 non-null object
SellerG          13580 non-null object
Date             13580 non-null object
Distance         13580 non-null float64
Postcode         13580 non-null float64
Bedroom2         13580 non-null float64
Bathroom         13580 non-null float64
Car              13518 non-null float64
Landsize         13580 non-null float64
BuildingArea     7130 non-null float64
YearBuilt        8205 non-null float64
CouncilArea      12211 non-null object
Lattitude        13580 non-null float64
Longtitude       13580 non-null float64
Regionname       13580 non-null object
Propertycount    13580 non-null float64
dtypes: float64(12), int64(1), object(8)
memory usage: 2.2+ MB
```

In [72]:

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

Out[72]:

```
Suburb              0
Address             0
Rooms               0
Type                0
Price               0
Method              0
SellerG             0
Date                0
Distance            0
Postcode            0
Bedroom2            0
Bathroom            0
Car                62
Landsize            0
BuildingArea     6450
YearBuilt        5375
CouncilArea      1369
Lattitude           0
Longtitude          0
Regionname          0
Propertycount       0
dtype: int64
```

In [222]:

```
df["Car"].fillna(df["Car"].mean(),inplace=True)
```

In [223]:

```
df["BuildingArea"].fillna(df["BuildingArea"].mean(),inplace=True)
```

In [224]:

```
df.drop("YearBuilt",axis=1,inplace=True)
```

In [225]:

```
df["CouncilArea"].fillna("Moreland",inplace=True)
```

In [226]:

```
#df["Date"] = pd.to_datetime(df['Date'], format='%d/%m/%Y')
```

In [227]:

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

In [228]:

```
df['Date'] = df['Date'].astype("datetime64[ns]")
```

In [229]:

```
# Create new columns
df['Day'] = df['Date'].dt.day
df['Month'] = df['Date'].dt.month
df['Year'] = df['Date'].dt.year
```

In [230]:

```
df.drop("Date",axis=1,inplace=True)
```

In [231]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13580 entries, 0 to 13579
Data columns (total 22 columns):
Suburb           13580 non-null object
Address          13580 non-null object
Rooms            13580 non-null int64
Type             13580 non-null object
Price            13580 non-null float64
Method           13580 non-null object
SellerG          13580 non-null object
Distance         13580 non-null float64
Postcode         13580 non-null float64
Bedroom2         13580 non-null float64
Bathroom         13580 non-null float64
Car              13580 non-null float64
Landsize         13580 non-null float64
BuildingArea     13580 non-null float64
CouncilArea      13580 non-null object
Lattitude        13580 non-null float64
Longtitude       13580 non-null float64
Regionname       13580 non-null object
Propertycount    13580 non-null float64
Day              13580 non-null int64
Month            13580 non-null int64
Year             13580 non-null int64
dtypes: float64(11), int64(4), object(7)
memory usage: 2.3+ MB
```

In [11]:

```
cor = df.corr()
cor
```

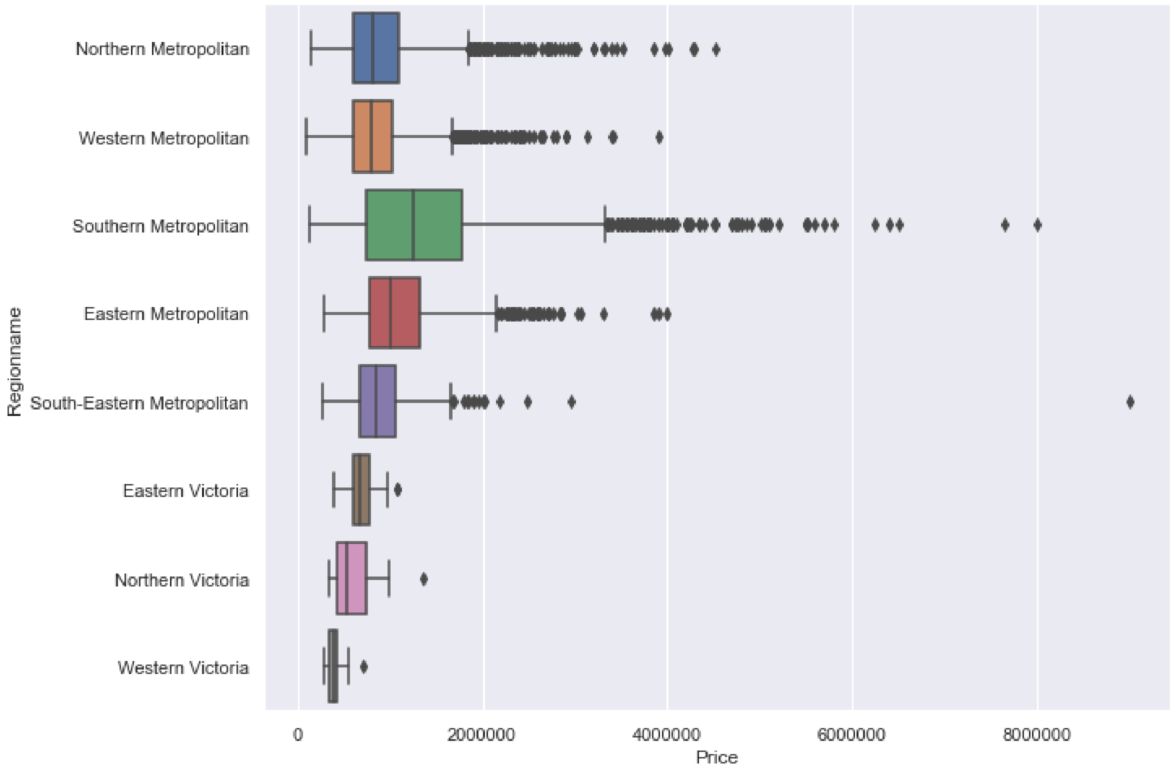
Out[11]:

	Rooms	Price	Distance	Postcode	Bedroom2	Bathroom	Car	Lanc
Rooms	1.000000	0.496634	0.294203	0.055303	0.944190	0.592934	0.407843	0.02
Price	0.496634	1.000000	-0.162522	0.107867	0.475951	0.467038	0.238637	0.03
Distance	0.294203	-0.162522	1.000000	0.431514	0.295927	0.127155	0.262074	0.02
Postcode	0.055303	0.107867	0.431514	1.000000	0.060584	0.113664	0.050201	0.02
Bedroom2	0.944190	0.475951	0.295927	0.060584	1.000000	0.584685	0.404721	0.02
Bathroom	0.592934	0.467038	0.127155	0.113664	0.584685	1.000000	0.321788	0.03
Car	0.407843	0.238637	0.262074	0.050201	0.404721	0.321788	1.000000	0.02
Landsize	0.025678	0.037507	0.025004	0.024558	0.025646	0.037130	0.026759	1.00
BuildingArea	0.091373	0.069570	0.073990	0.040714	0.089102	0.084462	0.068389	0.09
Lattitude	0.015948	-0.212934	-0.130723	-0.406104	0.015925	-0.070594	-0.001961	0.00
Longtitude	0.100771	0.203656	0.239425	0.445357	0.102238	0.118971	0.063304	0.01
Propertycount	-0.081530	-0.042153	-0.054910	0.062304	-0.081350	-0.052201	-0.024255	-0.00

Outliers

In [12]:

```
plt.figure(figsize=(10,8))
sns.boxplot(x="Price",y="Regionname",data=df)
plt.show()
```



In [232]:

```
df[(df["Regionname"] == "Western Victoria") & (df["Price"] > 500000)]
```

Out[232]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Distance	Postcode	Bedro
10740	Melton	28 Atkin St	4	h	550000.0	PI	Ryder	31.7	3337.0	
13487	Melton	21D Yuille St	5	h	710000.0	PI	Ryder	31.7	3337.0	

2 rows × 22 columns

In [233]:

```
df.drop([10740,13487],inplace=True)
```

In [234]:

```
df[(df["Regionname"] == "Northern Victoria") & (df["Price"] > 1000000)]
```

Out[234]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Distance	Postcode	Bed
13245	New Gisborne	71 Hamilton Rd	5	h	1355000.0	S	Raine	48.1	3438.0	

1 rows × 22 columns

In [235]:

```
df.drop([13245],inplace=True)
```

In [236]:

```
df[(df["Regionname"] == "Eastern Victoria") & (df["Price"] > 1000000)]
```

Out[236]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Distance	Postcode	Bed
10045	Silvan	1 Parker Rd	4	h	1070000.0	S	Ray	34.6	3795.0	
10504	Silvan	16 Eleanor Dr	3	h	1085000.0	S	Harcourts	34.6	3795.0	

2 rows × 22 columns

In [237]:

```
df.drop([10504,10045],inplace=True)
```

In [238]:

```
df[(df["Regionname"] == "South-Eastern Metropolitan") & (df["Price"] > 1700000)]
```

Out[238]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Distance	Postcod
4559	Oakleigh South	1172 North Rd	6	h	1900000.0	S	Harcourts	14.7	3167.
9647	Mulgrave	10 Adela Ct	4	h	2000000.0	S	Ray	18.8	3170.
10240	Edithvale	91 Edithvale Rd	6	h	1850000.0	S	O'Brien	27.0	3196.
10789	Parkdale	1/168 Beach Rd	4	h	2185000.0	S	Ray	21.5	3195.
11119	Mentone	5 Napier St	4	h	2025000.0	S	Greg	20.0	3194.
11127	Mordialloc	8 Ashmore Av	4	h	1800000.0	SP	Buxton	21.5	3195.
11318	Clayton	32 Wellington Rd	3	h	1950000.0	S	Buxton	16.7	3168.
12094	Mulgrave	35 Bevis St	3	h	9000000.0	PI	Hall	18.8	3170.
12307	Frankston South	30 Grange Rd	4	h	1905000.0	S	Eview	38.0	3199.
12656	Bonbeach	16B Newberry Av	4	t	1830000.0	S	hockingstuart	27.0	3196.
12691	Clayton	22 Burton Av	3	h	2950000.0	S	Darras	16.7	3168.
13518	Parkdale	63 The Corso	4	h	2475000.0	PI	Buxton	21.5	3195.

12 rows × 22 columns

In [239]:

```
df.drop([4559,9647,10240,10789,11119,11127,11318,12094,12307,12656,12691,13518],inplace=Tru
```

In [240]:

```
df[(df["Regionname"] == "Eastern Metropolitan") & (df["Price"] > 2050000)].index
```

Out[240]:

```
Int64Index([  979,   995,  2141,  2203,  2204,  2208,  2211,  3354,  3357,
             3360,  3396,  3411,  3416,  4028,  4030,  4034,  4035,  4036,
             4040,  4044,  5451,  6234,  7063,  7066,  7070,  7544,  8005,
             8011,  8012,  8013,  8088,  8091,  8588,  8793,  8879,  8880,
             9121,  9200,  9509,  9555,  9557,  9803,  9926, 10239, 10588,
            10709, 11081, 11131, 11222, 11226, 11227, 11446, 11977, 12045,
            12046, 12236, 12787, 12793, 13188],
           dtype='int64')
```

In [241]:

```
df.drop(df[(df["Regionname"] == "Eastern Metropolitan") & (df["Price"] > 2050000)].index,ir
```

In [242]:

```
df[(df["Regionname"] == "Southern Metropolitan") & (df["Price"] > 3300000)].index
```

Out[242]:

```
Int64Index([  108,   112,   233,   251,   270,   272,   273,   275,   388,
              515,
            ...
            12253, 12557, 12616, 12646, 12762, 12772, 13013, 13199, 13341,
            13468],
           dtype='int64', length=126)
```

In [243]:

```
df.drop(df[(df["Regionname"] == "Southern Metropolitan") & (df["Price"] > 3300000)].index,i
```

In [244]:

```
df[(df["Regionname"] == "Western Metropolitan") & (df["Price"] > 1600000)].index
```

Out[244]:

```
Int64Index([  146,   288,   292,   321,   330,   367,  2345,  2350,  2354,
             2356,
            ...
            12632, 12846, 13084, 13087, 13171, 13298, 13413, 13483, 13544,
            13578],
           dtype='int64', length=142)
```

In [245]:

```
df.drop(df[(df["Regionname"] == "Western Metropolitan") & (df["Price"] > 1600000)].index,ir
```

In [246]:

```
df[(df["Regionname"] == "Northern Metropolitan") & (df["Price"] > 1750000)].index
```

Out[246]:

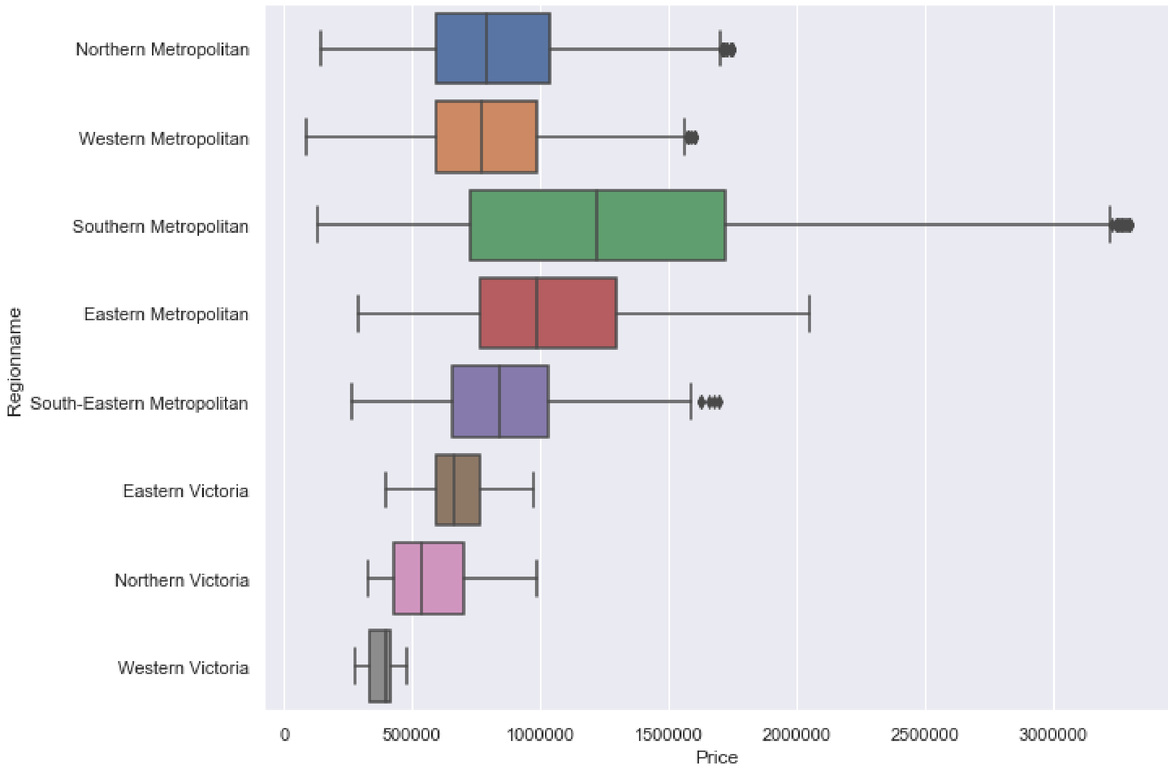
```
Int64Index([    6,   120,   124,   128,   131,   132,   138,   140,  1256,
             1258,
            ...
            12299, 12393, 12665, 12975, 13127, 13281, 13377, 13389, 13419,
            13531],
           dtype='int64', length=165)
```

In [247]:

```
df.drop(df[(df["Regionname"] == "Northern Metropolitan") & (df["Price"] > 1750000)].index,i
```

In [248]:

```
plt.figure(figsize=(10,8))
sns.boxplot(x="Price",y="Regionname",data=df)
plt.show()
```

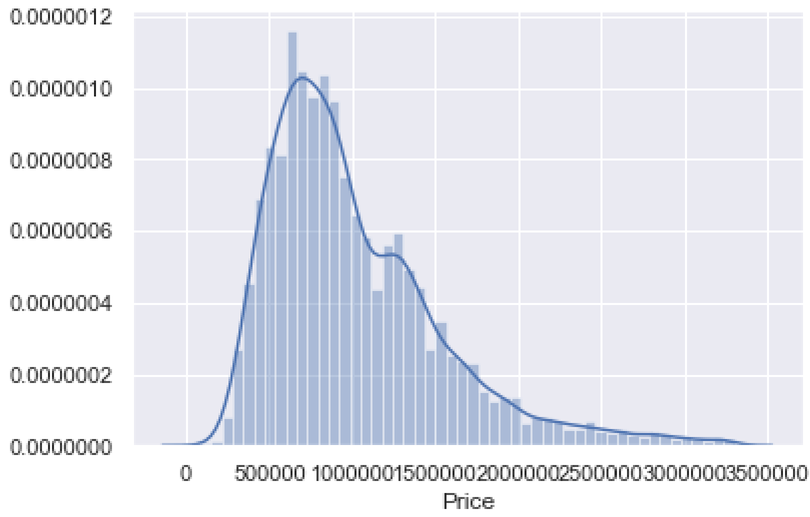


In [38]:

```
sns.distplot(df["Price"])
```

Out[38]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2e7275f08c8>
```



Separate Categorical and Numerical data

In [249]:

```
df_cat = df.select_dtypes("object")
```

In [250]:

```
df_num = df.select_dtypes(["int64","float64"])
```

In [251]:

```
df_cat.head()
```

Out[251]:

	Suburb	Address	Type	Method	SellerG	CouncilArea	Regionname
0	Abbotsford	85 Turner St	h	S	Biggin	Yarra	Northern Metropolitan
1	Abbotsford	25 Bloomburg St	h	S	Biggin	Yarra	Northern Metropolitan
2	Abbotsford	5 Charles St	h	SP	Biggin	Yarra	Northern Metropolitan
3	Abbotsford	40 Federation La	h	PI	Biggin	Yarra	Northern Metropolitan
4	Abbotsford	55a Park St	h	VB	Nelson	Yarra	Northern Metropolitan

In [252]:

```
df_num.head()
```

Out[252]:

	Rooms	Price	Distance	Postcode	Bedroom2	Bathroom	Car	Landsize	BuildingArea
0	2	1480000.0	2.5	3067.0	2.0	1.0	1.0	202.0	151.96765
1	2	1035000.0	2.5	3067.0	2.0	1.0	0.0	156.0	79.00000
2	3	1465000.0	2.5	3067.0	3.0	2.0	0.0	134.0	150.00000
3	3	850000.0	2.5	3067.0	3.0	2.0	1.0	94.0	151.96765
4	4	1600000.0	2.5	3067.0	3.0	1.0	2.0	120.0	142.00000

In [253]:

```
df_cat.describe(include="all")
```

Out[253]:

	Suburb	Address	Type	Method	SellerG	CouncilArea	Regionname
count	13071	13071	13071	13071	13071	13071	13071
unique	312	12897	3	5	263	33	8
top	Reservoir	5 Charles St	h	S	Nelson	Moreland	Southern Metropolitan
freq	359	3	8947	8687	1472	2468	4569

In [254]:

```
df_cat.drop(["Suburb","Address","SellerG"],axis=1,inplace=True)
```

In [255]:

```
df_cat.head()
```

Out[255]:

	Type	Method	CouncilArea	Regionname
0	h	S	Yarra	Northern Metropolitan
1	h	S	Yarra	Northern Metropolitan
2	h	SP	Yarra	Northern Metropolitan
3	h	PI	Yarra	Northern Metropolitan
4	h	VB	Yarra	Northern Metropolitan

In [256]:

```
## Label encoding
from sklearn.preprocessing import LabelEncoder
```

In [257]:

```
for col in df_cat:
    le = LabelEncoder()
    df_cat[col] = le.fit_transform(df_cat[col])
```

In [258]:

```
df_cat.head()
```

Out[258]:

	Type	Method	CouncilArea	Regionname
0	0	1	31	2
1	0	1	31	2
2	0	3	31	2
3	0	0	31	2
4	0	4	31	2

Skewness

In [259]:

```
df_num.head()
```

Out[259]:

	Rooms	Price	Distance	Postcode	Bedroom2	Bathroom	Car	Landsize	BuildingArea
0	2	1480000.0	2.5	3067.0	2.0	1.0	1.0	202.0	151.96765
1	2	1035000.0	2.5	3067.0	2.0	1.0	0.0	156.0	79.00000
2	3	1465000.0	2.5	3067.0	3.0	2.0	0.0	134.0	150.00000
3	3	850000.0	2.5	3067.0	3.0	2.0	1.0	94.0	151.96765
4	4	1600000.0	2.5	3067.0	3.0	1.0	2.0	120.0	142.00000

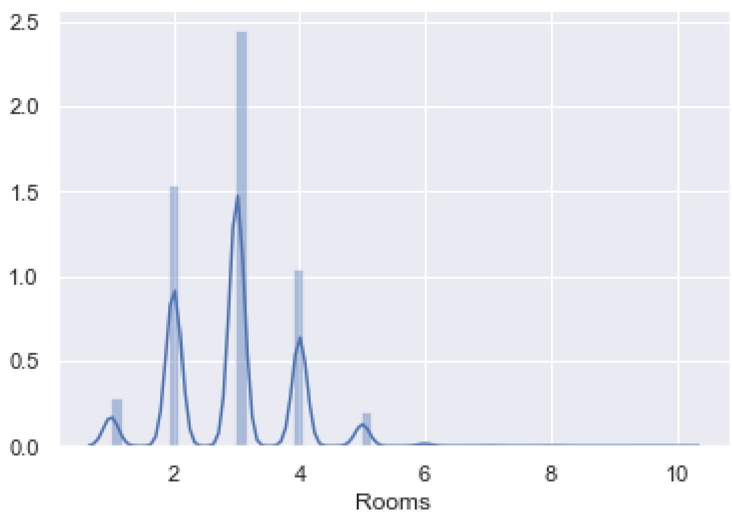
In [260]:

```
from scipy.stats import skew
```

In [51]:

```
for col in df_num:
    print("column:",col)
    print("Skewness:",skew(df_num[col]))

    plt.figure()
    sns.distplot(df_num[col])
    plt.show()

    print("----------------------------------------------------------")
```

```
column: Rooms
Skewness: 0.361408589530768
```



In [52]:

```
cor
```

Out[52]:

	Rooms	Price	Distance	Postcode	Bedroom2	Bathroom	Car	Lanc
Rooms	1.000000	0.496634	0.294203	0.055303	0.944190	0.592934	0.407843	0.02
Price	0.496634	1.000000	-0.162522	0.107867	0.475951	0.467038	0.238637	0.03
Distance	0.294203	-0.162522	1.000000	0.431514	0.295927	0.127155	0.262074	0.02
Postcode	0.055303	0.107867	0.431514	1.000000	0.060584	0.113664	0.050201	0.02
Bedroom2	0.944190	0.475951	0.295927	0.060584	1.000000	0.584685	0.404721	0.02
Bathroom	0.592934	0.467038	0.127155	0.113664	0.584685	1.000000	0.321788	0.03
Car	0.407843	0.238637	0.262074	0.050201	0.404721	0.321788	1.000000	0.02
Landsize	0.025678	0.037507	0.025004	0.024558	0.025646	0.037130	0.026759	1.00
BuildingArea	0.091373	0.069570	0.073990	0.040714	0.089102	0.084462	0.068389	0.09
Lattitude	0.015948	-0.212934	-0.130723	-0.406104	0.015925	-0.070594	-0.001961	0.00
Longtitude	0.100771	0.203656	0.239425	0.445357	0.102238	0.118971	0.063304	0.01
Propertycount	-0.081530	-0.042153	-0.054910	0.062304	-0.081350	-0.052201	-0.024255	-0.00

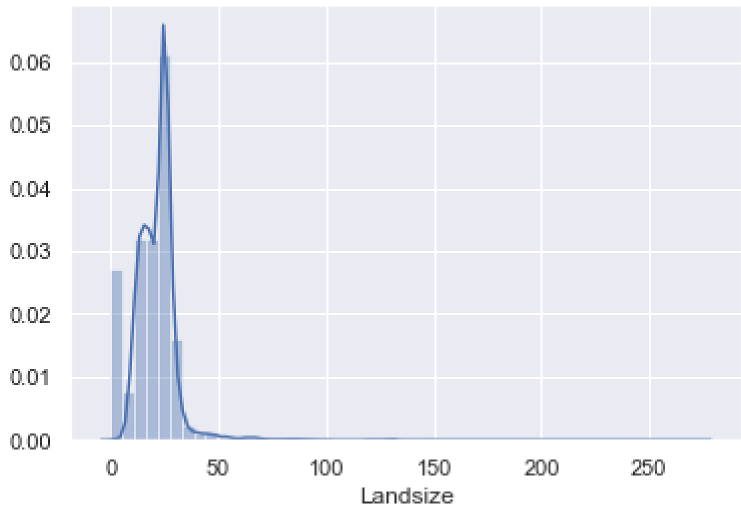
In [261]:

```
df_num["Landsize"] = np.sqrt(df_num["Landsize"])
df_num["BuildingArea"] = np.sqrt(df_num["BuildingArea"])
```

In [55]:

```
print(skew(df_num["Landsize"]))
sns.distplot(df_num["Landsize"])
plt.show()
```

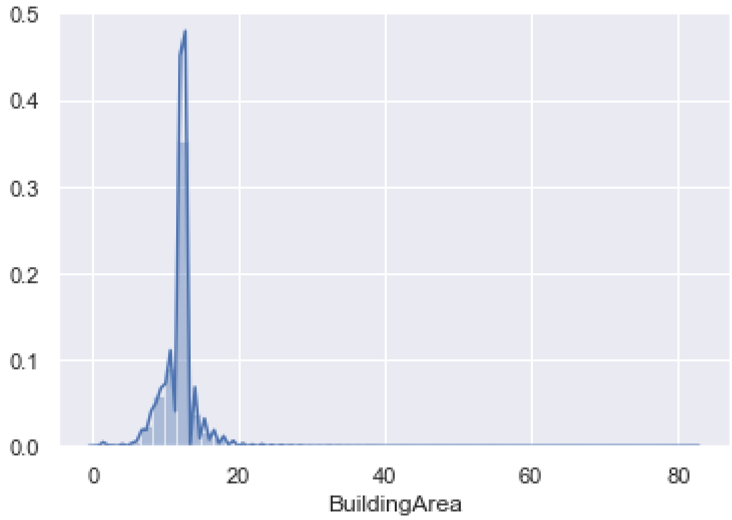
2.945231377729951



In [262]:

```
print(skew(df_num["BuildingArea"]))
sns.distplot(df_num["BuildingArea"])
plt.show()
```

3.2644538960148206



Concat both dataset

In [263]:

```
df_new = pd.concat([df_num,df_cat],axis=1)
```

In [264]:

```
df_new.head()
```

Out[264]:

	Rooms	Price	Distance	Postcode	Bedroom2	Bathroom	Car	Landsize	BuildingArea
0	2	1480000.0	2.5	3067.0	2.0	1.0	1.0	14.212670	12.327516
1	2	1035000.0	2.5	3067.0	2.0	1.0	0.0	12.489996	8.888194
2	3	1465000.0	2.5	3067.0	3.0	2.0	0.0	11.575837	12.247449
3	3	850000.0	2.5	3067.0	3.0	2.0	1.0	9.695360	12.327516
4	4	1600000.0	2.5	3067.0	3.0	1.0	2.0	10.954451	11.916375

Standardization

In [87]:

```
from sklearn.preprocessing import StandardScaler
```

In [88]:

```
for col in X:
    ss = StandardScaler()
    X[col] = ss.fit_transform(X)
```

In [89]:

```
X.head()
```

Out[89]:

	Rooms	Distance	Postcode	Bedroom2	Bathroom	Car	Landsize	BuildingArea	L
0	-0.960891	-0.960891	-0.960891	-0.960891	-0.960891	-0.960891	-0.960891	-0.960891	-C
1	-0.960891	-0.960891	-0.960891	-0.960891	-0.960891	-0.960891	-0.960891	-0.960891	-C
2	0.110034	0.110034	0.110034	0.110034	0.110034	0.110034	0.110034	0.110034	C
3	0.110034	0.110034	0.110034	0.110034	0.110034	0.110034	0.110034	0.110034	C
4	1.180958	1.180958	1.180958	1.180958	1.180958	1.180958	1.180958	1.180958	1

Building seperate model

In [265]:

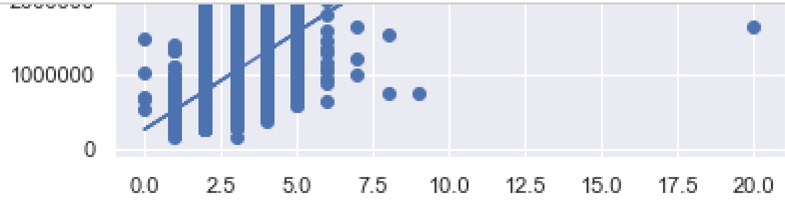
```
def ln_model(X,y):
    X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3, random_state=1)
    model = LinearRegression()
    model.fit(X_train,y_train)

    print("Modeling with ",X.columns[0])
    print("intercept: ",model.intercept_)
    print("Coef: ",model.coef_)

    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test,y_pred)
    rmse = np.sqrt(mse)

    r2 = r2_score(y_test,y_pred)

    print("mse: {},\nrmse: {},\nr2: {}".format(mse,rmse,r2))

    # Plot the model
    plt.figure()
    plt.scatter(X_test,y_test)
    plt.plot(X_test,y_pred)
    plt.show()
    print("---------------------------------------------------------------------")
```

In [266]:

```
X = df_new.drop("Price",axis=1)
y = df_new["Price"]
```

In [267]:

```
for col in X:
    ln_model(X[[col]],y)
```




```
----------------------------------------------------------------------
Modeling with  Bathroom
intercept:  505489.39408992097
Coef:  [336557.79134195]
mse: 229540871581.65268,
rmse: 479104.2387431494,
r2: 0.19376296322809017
```

In [70]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
```

Baseline Model

In [268]:

```
ln = LinearRegression()
```

In [269]:

```
ln.fit(X_train,y_train)
```

Out[269]:

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

In [270]:

```
ln.intercept_
```

Out[270]:

```
-111558033884.8408
```

In [271]:

```
ln.coef_.round(3)
# shows Multicolinearity
```

Out[271]:

```
array([ 3.39070000e+01, -1.89214284e+07,  2.15802521e+07, -7.46160876e+05,
        9.17653108e+07, -2.13954437e+07, -2.85772277e+06, -4.91263991e+06,
       -1.76975752e+09,  1.09232436e+09, -1.81545084e+05, -2.07098239e+07,
       -2.93800630e+04, -9.58991300e+03,  4.33091000e+02, -8.27743800e+03,
        3.75685850e+04, -8.44471000e+02,  1.70769100e+03, -2.31909040e+04,
        1.17512569e+05, -8.30055000e+02,  3.39968300e+03,  2.00382900e+03,
       -2.13300000e+00,  2.18776830e+04, -6.63059500e+03,  9.72840000e+01,
       -6.43828000e+02,  1.20945510e+04, -1.45720567e+05,  1.41903000e+02,
       -9.42124100e+03, -3.12100000e+00, -2.71169000e+02,  1.52970000e+02,
       -5.81100000e+00,  5.53320000e+01,  3.75149000e+02,  5.36308100e+03,
       -5.63600000e+00,  5.15125000e+02,  1.80437290e+04,  2.59939920e+04,
       -7.46258800e+03, -8.67300900e+03,  1.72149668e+05, -5.83141712e+05,
        2.12122000e+02, -2.00812040e+04, -3.38368340e+04,  3.30773000e+03,
        4.06806400e+03, -9.09156950e+04,  1.20726465e+05, -5.61497000e+02,
        1.25329700e+04, -5.02970000e+01, -3.80450000e+01, -1.93339880e+04,
        1.48834690e+04,  5.33900000e+00,  6.42637000e+02, -4.00570000e+02,
        4.15578100e+03,  3.39621080e+04, -1.74450000e+02,  1.57727600e+03,
       -2.58716562e+06,  1.08164997e+07,  3.47229260e+04,  8.35116635e+05,
       -2.41462559e+06,  1.03936160e+04,  3.49447428e+05,  2.84978000e+02,
       -3.52321000e+02,  1.27379690e+04])
```

In [273]:

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

In [274]:

```
mse = mean_squared_error(y_test,y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test,y_pred)

print("\nmse: {},\nrmse: {},\nr2: {}".format(mse,rmse,r2))
```

```
mse: 72823607285.52069,
rmse: 269858.4949293253,
r2: 0.7442150979894927
```

Feature selection

Wrapping Method

In [275]:

```
X.columns.values
```

Out[275]:

```
array(['Rooms', 'Distance', 'Postcode', 'Bedroom2', 'Bathroom', 'Car',
       'Landsize', 'BuildingArea', 'Lattitude', 'Longtitude',
       'Propertycount', 'Day', 'Month', 'Year', 'Type', 'Method',
       'CouncilArea', 'Regionname'], dtype=object)
```

In [380]:

```
X = df_new[['Rooms','Distance','Postcode','Type','Bathroom','Landsize','BuildingArea','Latt
y = df_new["Price"]
```

In [381]:

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3, random_state=1)
model = LinearRegression()
model.fit(X_train,y_train)

print("intercept: ",model.intercept_)
c = -1
for col in X:
    c = c + 1
    print(f"Coef of {col}:",model.coef_[c])

y_pred = model.predict(X_test)
mse = mean_squared_error(y_test,y_pred)
rmse = np.sqrt(mse)

r2 = r2_score(y_test,y_pred)

print("\nmse: {},\nrmse: {},\nr2: {}".format(mse,rmse,r2))
```

```
intercept:  -225641310.17938125
Coef of Rooms: 144807.95192798824
Coef of Distance: -46728.18572806922
Coef of Postcode: 841.4658010590247
Coef of Type: -227291.00612588192
Coef of Bathroom: 124715.6231421717
Coef of Landsize: 3494.3843017968193
Coef of BuildingArea: 22094.041911271823
Coef of Lattitude: -931623.0445571764
Coef of Longtitude: 1299202.6639131557
Coef of CouncilArea: -2630.9576033847216
Coef of Regionname: 38357.84574617682

mse: 105419016622.29375,
rmse: 324682.9478465011,
r2: 0.6297273117622844
```

In [382]:

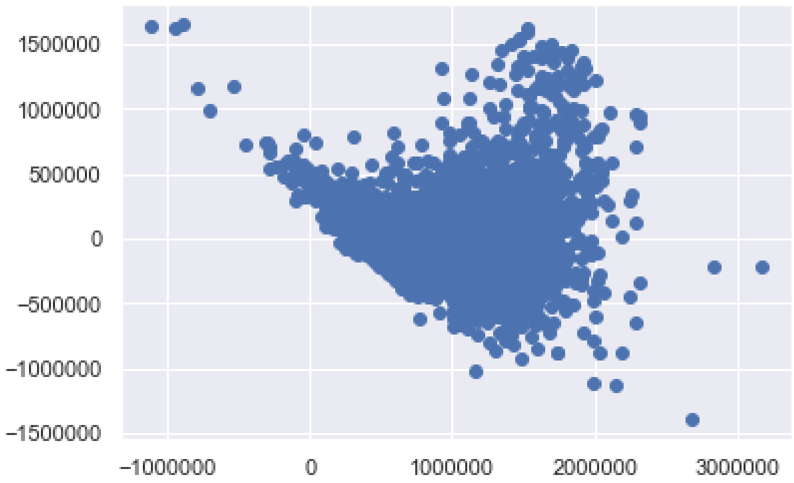
```
# multi-dimension trick - y_pred vs residuals
```

In [383]:

```
residuals = y_test - y_pred
```

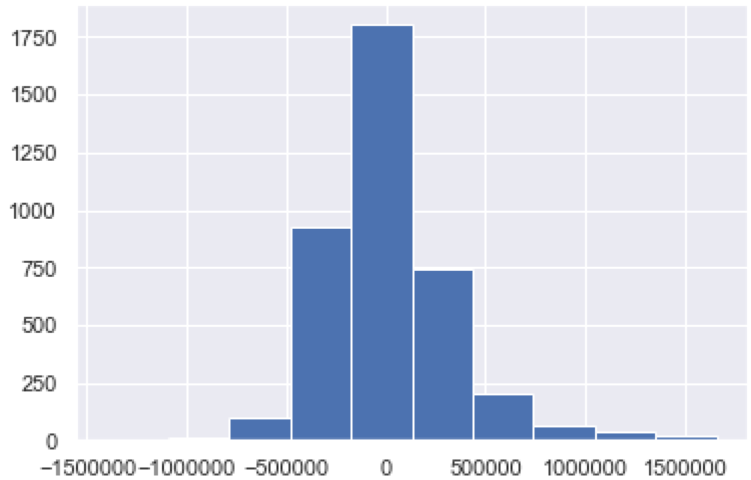
In [384]:

```
plt.figure()
plt.scatter(y_pred,residuals)
plt.show()
```



In [385]:

```
# residual histogram
plt.figure()
plt.hist(residuals)
plt.show()
# Normal distribution
```



In [386]:

```
# This show clearly no linear relationship
# Try for Polynomial Regression
```

Polynomial Redression

In [387]:

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

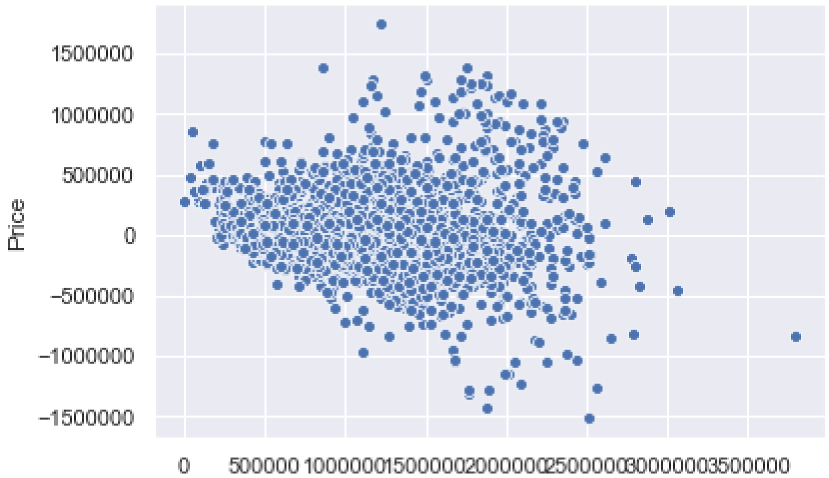
In [388]:

```
# Poly_2 - one curve
poly = PolynomialFeatures(2)
X_poly = poly.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_poly,y,test_size=0.3, random_state=1)

# Linear Regression
model = LinearRegression()
model.fit(X_train,y_train)
y_pred = model.predict(X_test)

mse = mean_squared_error(y_test,y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test,y_pred)

print("mse: {},\nrmse: {},\nr2: {}".format(mse,rmse,r2))
```

```
mse: 72823607285.52069,
rmse: 269858.4949293253,
r2: 0.7442150979894927
```

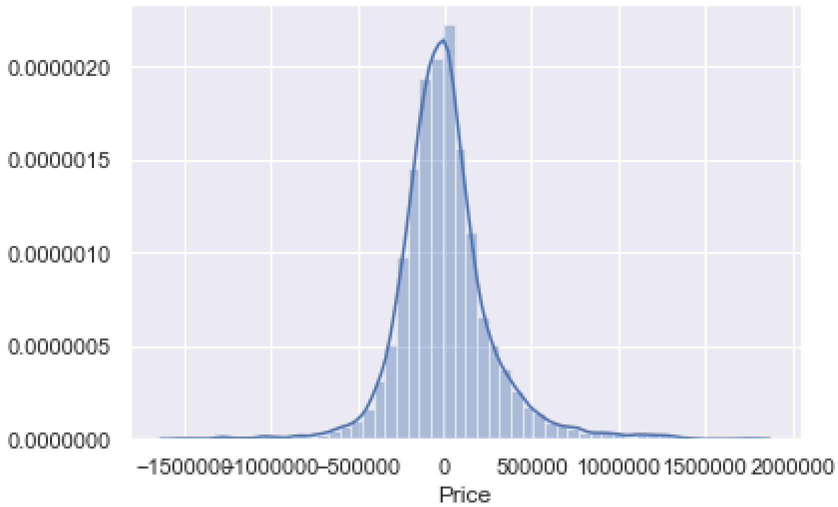
In [389]:

```
residuals = y_test - y_pred

plt.figure()
sns.scatterplot(y_pred,residuals)
plt.show()
```



In [390]:

```
plt.figure()
#plt.hist(residuals)
sns.distplot(residuals)
plt.show()
# Great our model residual is normally distributed
```



Gradient Descent

In [391]:

```
from sklearn import linear_model
```

In [392]:

```
gdm = linear_model.SGDRegressor(max_iter=100, tol=1e-3)
gdm.fit(X_train,y_train)
```

Out[392]:

```
SGDRegressor(alpha=0.0001, average=False, early_stopping=False, epsilon=0.1,
             eta0=0.01, fit_intercept=True, l1_ratio=0.15,
             learning_rate='invscaling', loss='squared_loss', max_iter=100,
             n_iter_no_change=5, penalty='l2', power_t=0.25, random_state=No
ne,
             shuffle=True, tol=0.001, validation_fraction=0.1, verbose=0,
             warm_start=False)
```

In [393]:

```
gdm.intercept_
```

Out[393]:

```
array([1.53926829e+09])
```

In [394]:

```
c = -1
for col in X:
    c = c + 1
    print(f"Coef of {col}:",gdm.coef_[c])
```

```
Coef of Rooms: 1502937347.5007973
Coef of Distance: 183283434999.49454
Coef of Postcode: -908645859136.8582
Coef of Type: 2218286971750.5605
Coef of Bathroom: 343540018519.4883
Coef of Landsize: 372625237818.25793
Coef of BuildingArea: -1474130507409.8137
Coef of Lattitude: -665323721361.8375
Coef of Longtitude: -84083386693.64734
Coef of CouncilArea: 234010802889.9794
Coef of Regionname: -1046870599997.1674
```

In [395]:

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

In [396]:

```
mse = mean_squared_error(y_test,y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test,y_pred)

print("\nmse: {},\nrmse: {},\nr2: {}".format(mse,rmse,r2))
```

```

mse: 5.4488603286176185e+44,
rmse: 2.334279402431855e+22,
r2: -1.913852193231728e+33
```

In [ ]:

```
# In Gradient Descent since we minimize cost function-MSE value therefore r2 score can be i
```

Regularization

In [292]:

```
print(model.score(X_train,y_train))
print(model.score(X_test,y_test))
```

```
0.7599271395699001
0.7442150979894927
```

In [293]:

```
from sklearn.linear_model import Lasso # Lambda*sum(abs(coef))
from sklearn.linear_model import Ridge # Lambda*sum(square(coef))
```

In [294]:

```
# Finding Right Lambda value
```

In [295]:

```
# Ridge
for i in range(1,200,10):
    l2 = Ridge(i)
    l2.fit(X_train,y_train)
    print(i,":",l2.score(X_test,y_test))
```

```
1 : 0.73573716255834
11 : 0.7350403112697748
21 : 0.7347582450587626
31 : 0.7345156106336154
41 : 0.7343213430239457
51 : 0.7341671294842249
61 : 0.7340439240309033
71 : 0.7339445355644829
81 : 0.7338635828078292
91 : 0.7337970718510545
101 : 0.7337420169236393
111 : 0.7336961587234747
121 : 0.7336577647962127
131 : 0.7336254892208083
141 : 0.7335982734431449
151 : 0.7335752754147304
161 : 0.7335558182968853
171 : 0.7335393528584379
181 : 0.7335254295282474
191 : 0.7335136773930049
```

In [296]:

```
# Lasso
for i in range(200,600,50):
    l1 = Lasso(i)
    l1.fit(X_train,y_train)
    print(i,":",l1.score(X_test,y_test))
```

```
200 : 0.7268454945601779
250 : 0.7269480079140196
300 : 0.7271014722305962
350 : 0.7271956955095193
400 : 0.7272751411794789
450 : 0.7273493885619491
500 : 0.7273618461173633
550 : 0.7273803737509982
```

In [212]:

```
# Ridge - 11
# Lasso - 300
```

Regularization model

In [297]:

```
l2 = Ridge(alpha=11)
l2.fit(X_train,y_train)
print(l2.score(X_test,y_test))
```

```
0.7350403112697748
```

In [299]:

```
c = -1
for col in X:
    c = c + 1
    print(f"Coef of {col}:",l2.coef_[c])
```

```
Coef of Rooms: 0.0
Coef of Distance: -9084.298596239567
Coef of Postcode: 78397.81436569493
Coef of Type: 201473.19652433204
Coef of Bathroom: 7618.775591941714
Coef of Landsize: -6179.229930907951
Coef of BuildingArea: -74138.4879705636
Coef of Lattitude: -16488.324779055925
Coef of Longtitude: -399.7071035871236
Coef of CouncilArea: 561.7405963991837
Coef of Regionname: 3561.6422402001235
```

In [219]:

```
l1 = Lasso(alpha=300)
l1.fit(X_train,y_train)
print(l1.score(X_test,y_test))
```

```
0.7271014722305962
```

In [300]:

```
c = -1
for col in X:
    c = c + 1
    print(f"Coef of {col}:",l1.coef_[c])
```

```
Coef of Rooms: 0.0
Coef of Distance: -0.0
Coef of Postcode: -13202.29706151846
Coef of Type: 1567.2071181289332
Coef of Bathroom: -0.0
Coef of Landsize: 0.0
Coef of BuildingArea: 5368.239207678092
Coef of Lattitude: 0.0
Coef of Longtitude: -0.0
Coef of CouncilArea: 0.0
Coef of Regionname: -1799.3249824053958
```

In [312]:

```
X = X[['Postcode','Type','BuildingArea','Regionname']]
```

Cross Validation / K-fold Validation

In [302]:

```
from sklearn.model_selection import cross_val_score
```

In [333]:

```
l2_cross = cross_val_score(l2,X,y,cv=5)
```

In [334]:

```
l2_cross
```

Out[334]:

```
array([0.64832921, 0.62533289, 0.64012033, 0.56941451, 0.54127199])
```

In [335]:

```
np.mean(l2_cross)
```

Out[335]:

```
0.6048937872105933
```

In [336]:

```
l1_cross = cross_val_score(l1,X,y,cv=5)
```

In [337]:

```
l1_cross
```

Out[337]:

```
array([0.64859634, 0.62763589, 0.64155377, 0.56965708, 0.54038565])
```

In [338]:

```
l1_cross.mean()
```

Out[338]:

```
0.605565744139895
```

In [ ]:

```
#  Lasso having little higher score than Ridge. Hence Lasso is better model option
```