Ridge Regression is a regularized version of linear Regression α∑(slope)^2 is added to the cost function. Ridge Regression= 1/n∑(actual - Predicted)^2+α∑(slope)^2 .This force the learning alagorithm to not only for the data but also keep the model weights as small as possible.The regularization term should only be added to the cost function during training.Once the model is trained we want to use the unregularized performance measure to evaluate the model Performance. KeyPoint:The hyperparameter alpha(α) controls how much you want to regularize the model.If α=0,the Ridge Regression is just Linear Regression.If α is very large ,then all weights end up very close to Zero and the Result is flat line going through the data mean's.

Lasso Regression :least Absolute Shrinkage and selection operator Regression is another version of regularized Version of Linear Regression,its add a regularization term to the cost function like Ridge Regression ,but it uses the L1 norm of the Weight vector instead of the half the sqaure of the L2 norm Lasso Regression = 1/n∑(actual - Predicted)^2+α∑|slope| using Lasso regression we cant use only for add regularization we can use for Feature selection also due to if any features are not correlated to dependent feature or target variable then then it add regularization and remove the feature based on the Score.

We are using this Regularization techniques when our model fall in overfitting and feature selection.

In [80]:
```
#import library's
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import dtale
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn import linear_model
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error
```

In [ ]:

In [2]:
```
cd \Users\758449\Downloads
```
```
C:\Users\758449\Downloads
```

Description : Melbourne is currently experiencing a housing bubble (some experts say it may burst soon). Maybe someone can find a trend or give a prediction? Which suburbs are the best to buy in? Which ones are value for money? Where's the expensive side of town? And more importantly where should I buy a 2 bedroom unit?

In [3]:
```
#read the data using pandas
data = pd.read_csv('Melbourne_housing_FULL.csv')
```

In [4]:
```
data
```

Out[4]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Postcode	...	Bathroom	Car	Lan
0	Abbotsford	68 Studley St	2	h	NaN	SS	Jellis	3/09/2016	2.5	3067.0	...	1.0	1.0	
1	Abbotsford	85 Turner St	2	h	1480000.0	S	Biggin	3/12/2016	2.5	3067.0	...	1.0	1.0	
2	Abbotsford	25 Bloomburg St	2	h	1035000.0	S	Biggin	4/02/2016	2.5	3067.0	...	1.0	0.0	
3	Abbotsford	18/659 Victoria St	3	u	NaN	VB	Rounds	4/02/2016	2.5	3067.0	...	2.0	1.0	
4	Abbotsford	5 Charles St	3	h	1465000.0	SP	Biggin	4/03/2017	2.5	3067.0	...	2.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
34852	Yarraville	13 Burns St	4	h	1480000.0	PI	Jas	24/02/2018	6.3	3013.0	...	1.0	3.0	
34853	Yarraville	29A Murray St	2	h	888000.0	SP	Sweeney	24/02/2018	6.3	3013.0	...	2.0	1.0	
34854	Yarraville	147A Severn St	2	t	705000.0	S	Jas	24/02/2018	6.3	3013.0	...	1.0	2.0	
34855	Yarraville	12/37 Stephen St	3	h	1140000.0	SP	hockingstuart	24/02/2018	6.3	3013.0	...	NaN	NaN	
34856	Yarraville	3 Tarrengower St	2	h	1020000.0	PI	RW	24/02/2018	6.3	3013.0	...	1.0	0.0	

34857 rows × 21 columns

In [5]:
```
#Get the all the information about the data using Info() function from python inbuilt
data.info()
```
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34857 entries, 0 to 34856
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Suburb         34857 non-null  object
 1   Address        34857 non-null  object
 2   Rooms          34857 non-null  int64
 3   Type           34857 non-null  object
 4   Price          27247 non-null  float64
 5   Method         34857 non-null  object
 6   SellerG        34857 non-null  object
 7   Date           34857 non-null  object
 8   Distance       34856 non-null  float64
 9   Postcode       34856 non-null  float64
 10  Bedroom2       26640 non-null  float64
 11  Bathroom       26631 non-null  float64
 12  Car            26129 non-null  float64
 13  Landsize       23047 non-null  float64
 14  BuildingArea   13742 non-null  float64
 15  YearBuilt      15551 non-null  float64
 16  CouncilArea    34854 non-null  object
 17  Lattitude      26881 non-null  float64
 18  Longtitude     26881 non-null  float64
 19  Regionname     34854 non-null  object
 20  Propertycount  34854 non-null  float64
dtypes: float64(12), int64(1), object(8)
memory usage: 5.6+ MB
```

In [8]:
```
#Describe the summary of the data here we can analysis the some statistical terms
data.describe()
```

Out[8]:

	Rooms	Price	Distance	Postcode	Bedroom2	Bathroom	Car	Landsize	BuildingArea
count	34857.000000	2.724700e+04	34856.000000	34856.000000	26640.000000	26631.000000	26129.000000	23047.000000	13742.00000
mean	3.031012	1.050173e+06	11.184929	3116.062859	3.084647	1.624798	1.728845	593.598993	160.25640
std	0.969933	6.414671e+05	6.788892	109.023903	0.980690	0.724212	1.010771	3398.841946	401.26706
min	1.000000	8.500000e+04	0.000000	3000.000000	0.000000	0.000000	0.000000	0.000000	0.00000
25%	2.000000	6.350000e+05	6.400000	3051.000000	2.000000	1.000000	1.000000	224.000000	102.00000
50%	3.000000	8.700000e+05	10.300000	3103.000000	3.000000	2.000000	2.000000	521.000000	136.00000
75%	4.000000	1.295000e+06	14.000000	3156.000000	4.000000	2.000000	2.000000	670.000000	188.00000
max	16.000000	1.120000e+07	48.100000	3978.000000	30.000000	12.000000	26.000000	433014.000000	44515.00000

In [12]:
```
#checking for the null values in the given DataSet
data.isnull().sum()
```

Out[12]:
```
Suburb               0
Address              0
Rooms                0
Type                 0
Price             7610
Method               0
SellerG              0
Date                 0
Distance             1
Postcode             1
Bedroom2          8217
Bathroom          8226
Car               8728
Landsize         11810
BuildingArea     21115
YearBuilt        19306
CouncilArea          3
Lattitude         7976
Longtitude        7976
Regionname           3
Propertycount        3
dtype: int64
```

In [11]:
```
#Using Seaborn Heatmap we can visualize the null values in our data
sns.heatmap(data.isnull(),cmap='viridis')
```

Out[11]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2e03b4e0cd0>
```

In [18]:
```
#fill null values with ZERO's and median and drop out the features
fill_zero = ['Distance','Postcode','CouncilArea','Bedroom2','Bathroom','Regionname','Propertycount','Ca
r']

data[fill_zero]=data[fill_zero].fillna(0)
```

In [19]:
```
data.isnull().sum()
```

Out[19]:
```
Suburb               0
Address              0
Rooms                0
Type                 0
Price             7610
Method               0
SellerG              0
Date                 0
Distance             0
Postcode             0
Bedroom2             0
Bathroom             0
Car                  0
Landsize         11810
BuildingArea     21115
YearBuilt        19306
CouncilArea          0
Lattitude         7976
Longtitude        7976
Regionname           0
Propertycount        0
dtype: int64
```

In [20]:
```
data['Landsize'] = data['Landsize'].fillna(data.Landsize.median())
data['BuildingArea'] = data['BuildingArea'].fillna(data.BuildingArea.median())
data['YearBuilt'] = data['YearBuilt'].fillna(data.YearBuilt.median())
```

In [22]:
```
data.isnull().sum()
```

Out[22]:
```
Suburb              0
Address             0
Rooms               0
Type                0
Price            7610
Method              0
SellerG             0
Date                0
Distance            0
Postcode            0
Bedroom2            0
Bathroom            0
Car                 0
Landsize            0
BuildingArea        0
YearBuilt           0
CouncilArea         0
Lattitude        7976
Longtitude       7976
Regionname          0
Propertycount       0
dtype: int64
```

In [31]:
```
data.drop(['Lattitude','Longtitude'],axis = 1,inplace =True)
```

In [32]:
```
data.isnull().sum()
```

Out[32]:
```
Suburb              0
Address             0
Rooms               0
Type                0
Price            7610
Method              0
SellerG             0
Date                0
Distance            0
Postcode            0
Bedroom2            0
Bathroom            0
Car                 0
Landsize            0
BuildingArea        0
YearBuilt           0
CouncilArea         0
Regionname          0
Propertycount       0
dtype: int64
```

In [34]:
```
data.dropna(inplace =True)
```

In [ ]:

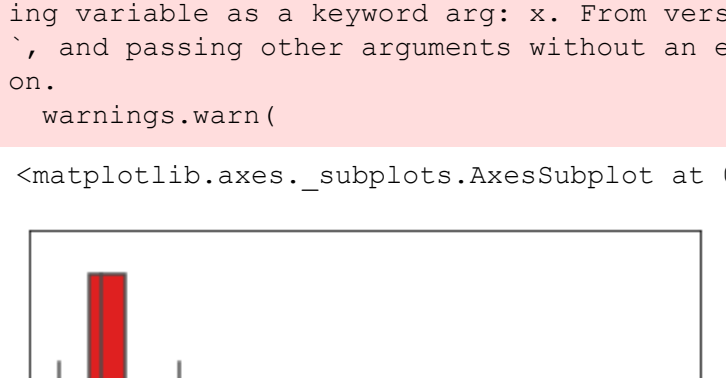
In [36]:
```
data.head()
```

Out[36]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Postcode	Bedroom2	Bathroom	Car	Landsize
1	Abbotsford	85 Turner St	2	h	1480000.0	S	Biggin	3/12/2016	2.5	3067.0	2.0	1.0	1.0	202.0
2	Abbotsford	25 Bloomburg St	2	h	1035000.0	S	Biggin	4/02/2016	2.5	3067.0	2.0	1.0	0.0	156.0
4	Abbotsford	5 Charles St	3	h	1465000.0	SP	Biggin	4/03/2017	2.5	3067.0	3.0	2.0	0.0	134.0
5	Abbotsford	40 Federation La	3	h	850000.0	PI	Biggin	4/03/2017	2.5	3067.0	3.0	2.0	1.0	94.0
6	Abbotsford	55a Park St	4	h	1600000.0	VB	Nelson	4/06/2016	2.5	3067.0	3.0	1.0	2.0	120.0

In [100]:
```
sns.boxplot(data['Price'],color='red')
```
```
C:\Users\758449\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the follow
ing variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data
`, and passing other arguments without an explicit keyword will result in an error or misinterpretati
on.
  warnings.warn(
```

Out[100]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2e0185070a0>
```



In [37]:
```
data.drop(['Type','Method','Address'],axis = 1,inplace =True)
```

In [39]:
```
data.head()
```

Out[39]:

	Suburb	Rooms	Price	SellerG	Date	Distance	Postcode	Bedroom2	Bathroom	Car	Landsize	BuildingArea	YearBuilt
1	Abbotsford	2	1480000.0	Biggin	3/12/2016	2.5	3067.0	2.0	1.0	1.0	202.0	136.0	1970.0
2	Abbotsford	2	1035000.0	Biggin	4/02/2016	2.5	3067.0	2.0	1.0	0.0	156.0	79.0	1900.0
4	Abbotsford	3	1465000.0	Biggin	4/03/2017	2.5	3067.0	3.0	2.0	0.0	134.0	150.0	1900.0
5	Abbotsford	3	850000.0	Biggin	4/03/2017	2.5	3067.0	3.0	2.0	1.0	94.0	136.0	1970.0
6	Abbotsford	4	1600000.0	Nelson	4/06/2016	2.5	3067.0	3.0	1.0	2.0	120.0	142.0	2014.0

In [46]:
```
data.drop(['Date'],axis =1,inplace = True)
```

In [40]:
```
data['Suburb'].unique()
```

Out[40]:
```
array(['Abbotsford', 'Airport West', 'Albert Park', 'Alphington',
       'Altona', 'Altona North', 'Armadale', 'Ascot Vale', 'Ashburton',
       'Ashwood', 'Avondale Heights', 'Balaclava', 'Balwyn',
       'Balwyn North', 'Bentleigh', 'Bentleigh East', 'Box Hill',
       'Braybrook', 'Brighton', 'Brighton East', 'Brunswick',
       'Brunswick West', 'Bulleen', 'Burwood', 'Camberwell', 'Canterbury',
       'Carlton North', 'Carnegie', 'Caulfield', 'Caulfield North',
       'Caulfield South', 'Chadstone', 'Clifton Hill', 'Coburg',
       'Coburg North', 'Collingwood', 'Doncaster', 'Eaglemont',
       'Elsternwick', 'Elwood', 'Essendon', 'Essendon North', 'Fairfield',
       'Fitzroy', 'Fitzroy North', 'Flemington', 'Footscray', 'Glen Iris',
       'Glenroy', 'Gowanbrae', 'Hadfield', 'Hampton', 'Hampton East',
       'Hawthorn', 'Heidelberg Heights', 'Heidelberg West', 'Hughesdale',
       'Ivanhoe', 'Kealba', 'Keilor East', 'Kensington', 'Kew',
       'Kew East', 'Kooyong', 'Maidstone', 'Malvern', 'Malvern East',
       'Maribyrnong', 'Melbourne', 'Middle Park', 'Mont Albert',
       'Moonee Ponds', 'Moorabbin', 'Newport', 'Niddrie',
       'North Melbourne', 'Northcote', 'Oak Park', 'Oakleigh South',
       'Parkville', 'Pascoe Vale', 'Port Melbourne', 'Prahran', 'Preston',
       'Reservoir', 'Richmond', 'Rosanna', 'Seddon', 'South Melbourne',
       'South Yarra', 'Southbank', 'Spotswood', 'St Kilda', 'Strathmore',
       'Sunshine', 'Sunshine North', 'Sunshine West', 'Surrey Hills',
       'Templestowe Lower', 'Thornbury', 'Toorak', 'Viewbank', 'Watsonia',
       'West Melbourne', 'Williamstown', 'Williamstown North', 'Windsor',
       'Yallambie', 'Yarraville', 'Aberfeldie', 'Bellfield',
       'Brunswick East', 'Burnley', 'Campbellfield', 'Carlton',
       'East Melbourne', 'Essendon West', 'Fawkner', 'Hawthorn East',
       'Heidelberg', 'Ivanhoe East', 'Jacana', 'Kingsbury', 'Kingsville',
       'Murrumbeena', 'Ormond', 'West Footscray', 'Albion', 'Brooklyn',
       'Glen Huntly', 'Oakleigh', 'Ripponlea', 'Cremorne', 'Docklands',
       'South Kingsville', 'Strathmore Heights', 'Travancore',
       'Caulfield East', 'Seaholme', 'Keilor Park', 'Gardenvale',
       'Princes Hill', 'Ardeer', 'Attwood', 'Bayswater',
       'Bayswater North', 'Beaumaris', 'Berwick', 'Blackburn',
       'Blackburn South', 'Boronia', 'Briar Hill', 'Broadmeadows',
       'Bundoora', 'Burnside Heights', 'Burwood East', 'Cairnlea',
       'Caroline Springs', 'Cheltenham', 'Clyde North', 'Craigieburn',
       'Cranbourne', 'Croydon', 'Croydon North', 'Dandenong',
       'Dandenong North', 'Diamond Creek', 'Dingley Village',
       'Doncaster East', 'Donvale', 'Doreen', 'Edithvale', 'Eltham',
       'Eltham North', 'Epping', 'Eumemmerring', 'Ferntree Gully',
       'Forest Hill', 'Frankston', 'Frankston North', 'Frankston South',
       'Gisborne', 'Gladstone Park', 'Glen Waverley', 'Greensborough',
       'Greenvale', 'Hallam', 'Healesville', 'Highett', 'Hillside',
       'Hoppers Crossing', 'Huntingdale', 'Keilor Downs', 'Keilor Lodge',
       'Keysborough', 'Kings Park', 'Lalor', 'Lower Plenty', 'Lynbrook',
       'MacLeod', 'Melton', 'Melton South', 'Mentone', 'Mernda',
       'Mill Park', 'Mitcham', 'Montmorency', 'Mordialloc',
       'Mount Waverley', 'Narre Warren', 'Noble Park', 'Nunawading',
       'Oakleigh East', 'Parkdale', 'Point Cook', 'Ringwood',
       'Ringwood East', 'Rockbank', 'Rowville', 'Roxburgh Park',
       'Sandringham', 'Seabrook', 'Seaford', 'Skye', 'South Morang',
       'Springvale', 'St Albans', 'St Helena', 'Sunbury', 'Sydenham',
       'Tarneit', 'Taylors Hill', 'Taylors Lakes', 'Tecoma', 'The Basin',
       'Thomastown', 'Truganina', 'Tullamarine', 'Vermont', 'Wantirna',
       'Wantirna South', 'Werribee', 'Werribee South', 'Westmeadows',
       'Williams Landing', 'Wollert', 'Wyndham Vale', 'Altona Meadows',
       'Aspendale', 'Black Rock', 'Blackburn North', 'Bonbeach', 'Carrum',
       'Chelsea', 'Clayton', 'Clayton South', 'Dallas', 'Delahey',
       'Doveton', 'Heathmont', 'McKinnon', 'Melton West', 'Mooroolbark',
       'Mulgrave', 'Pakenham', 'Ringwood North', 'Templestowe',
       'Vermont South', 'Warrandyte', 'Watsonia North', 'Wattle Glen',
       'Wheelers Hill', 'Aspendale Gardens', 'Carrum Downs',
       'Cranbourne East', 'Deer Park', 'Heatherton', 'Langwarrin',
       'Notting Hill', 'Albanvale', 'Beaconsfield Upper',
       'Chelsea Heights', 'Chirnside Park', 'Coolaroo', 'Keilor',
       'Kilsyth', 'Meadow Heights', 'Mount Evelyn', 'North Warrandyte',
       'Riddells Creek', 'Sandhurst', 'Scoresby', 'Silvan',
       'Croydon Hills', 'Croydon South', 'Derrimut', 'Diggers Rest',
       'Hampton Park', 'Knoxfield', 'Upwey', 'Bacchus Marsh',
       'Cranbourne North', 'Montrose', 'Bullengarook', 'Clarinda',
       'Deepdene', 'Hurstbridge', 'Kurunjang', 'Wonga Park',
       'Endeavour Hills', 'Mickleham', 'Officer', 'Waterways',
       'Beaconsfield', 'Springvale South', 'Yarra Glen', 'Brookfield',
       'Emerald', 'Plenty', 'Whittlesea', 'Burnside', 'New Gisborne',
       'Patterson Lakes', 'Wallan', 'Laverton', 'Lilydale', 'Plumpton',
       'croydon', 'Monbulk', 'Warranwood', 'Wildwood', 'Gisborne South',
       'Research', 'viewbank', 'Botanic Ridge', 'Bulla', 'Coldstream',
       'Cranbourne West', 'Darley', 'Eynesbury', 'Fawkner Lot',
       'Ferny Creek', 'Wandin North', 'Lysterfield', 'Kalkallo'],
      dtype=object)
```

In [43]:
```
label_encoder = preprocessing.LabelEncoder()
```

In [55]:
```
obj_df = data.select_dtypes(include=['object'])
obj_df
```

Out[55]:

	Suburb	SellerG	CouncilArea	Regionname
1	Abbotsford	Biggin	Yarra City Council	Northern Metropolitan
2	Abbotsford	Biggin	Yarra City Council	Northern Metropolitan
4	Abbotsford	Biggin	Yarra City Council	Northern Metropolitan
5	Abbotsford	Biggin	Yarra City Council	Northern Metropolitan
6	Abbotsford	Nelson	Yarra City Council	Northern Metropolitan
...	...	...	...	...
34852	Yarraville	Jas	Maribyrnong City Council	Western Metropolitan
34853	Yarraville	Sweeney	Maribyrnong City Council	Western Metropolitan
34854	Yarraville	Jas	Maribyrnong City Council	Western Metropolitan
34855	Yarraville	hockingstuart	Maribyrnong City Council	Western Metropolitan
34856	Yarraville	RW	Maribyrnong City Council	Western Metropolitan

27247 rows × 4 columns

In [59]:
```
data = pd.get_dummies(data,drop_first= True)
data.head()
```

Out[59]:

	Suburb	Rooms	Price	SellerG	Distance	Postcode	Bedroom2	Bathroom	Car	Landsize	...	CouncilArea_Yarra City Council	CouncilArea_Ya Ranges Sh Cour
1	0	2	1480000.0	32	2.5	3067.0	2.0	1.0	1.0	202.0	...	1	
2	0	2	1035000.0	32	2.5	3067.0	2.0	1.0	0.0	156.0	...	1	
4	0	3	1465000.0	32	2.5	3067.0	3.0	2.0	0.0	134.0	...	1	
5	0	3	850000.0	32	2.5	3067.0	3.0	2.0	1.0	94.0	...	1	
6	0	4	1600000.0	206	2.5	3067.0	3.0	1.0	2.0	120.0	...	1	

5 rows × 54 columns

In [57]:
```
data['Suburb']
```

Out[57]:
```
1          0
2          0
4          0
5          0
6          0
        ...
34852    342
34853    342
34854    342
34855    342
34856    342
Name: Suburb, Length: 27247, dtype: int32
```

In [62]:
```
X = data.drop('Price',axis =1)
y = data['Price']
```

In [93]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=10)
```

In [66]:
```
X_train.shape
```

Out[66]:
```
(18255, 53)
```

In [69]:
```
X_test.shape
```

Out[69]:
```
(8992, 53)
```

In [74]:
```
reg =LinearRegression().fit(X_train,y_train)
```

In [94]:
```
reg.score(X_test,y_test)
```

Out[94]:
```
0.5546052421503496
```

In [95]:
```
reg.score(X_train,y_train)
```

Out[95]:
```
0.20647958005995748
```

In [90]:
```
lasso_reg=linear_model.Lasso(alpha=50,max_iter=100,tol=0.1)
lasso_reg.fit(X_train,y_train)
```
```
C:\Users\758449\Anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:529: Converge
nceWarning: Objective did not converge. You might want to increase the number of iterations. Duality
gap: 1464232440445804.2, tolerance: 753447144002420.0
  model = cd_fast.enet_coordinate_descent(
```

Out[90]:
```
Lasso(alpha=50, max_iter=100, tol=0.1)
```

In [96]:
```
lasso_reg.score(X_test,y_test)
```

Out[96]:
```
0.5603603051637677
```

In [97]:
```
lasso_reg.score(X_train,y_train)
```

Out[97]:
```
0.565789226059445
```