In [80]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [81]:
```
df = pd.read_csv('/home/joseph/Desktop/ml lab/lab1/dataset/Melbourne_housin
df.head(20)
```

Out[81]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Pos
0	Abbotsford	68 Studley St	2	h	NaN	SS	Jellis	3/09/2016	2.5	3
1	Abbotsford	85 Turner St	2	h	1480000.0	S	Biggin	3/12/2016	2.5	3
2	Abbotsford	25 Bloomburg St	2	h	1035000.0	S	Biggin	4/02/2016	2.5	3
3	Abbotsford	18/659 Victoria St	3	u	NaN	VB	Rounds	4/02/2016	2.5	3
4	Abbotsford	5 Charles St	3	h	1465000.0	SP	Biggin	4/03/2017	2.5	3
5	Abbotsford	40 Federation La	3	h	850000.0	PI	Biggin	4/03/2017	2.5	3
6	Abbotsford	55a Park St	4	h	1600000.0	VB	Nelson	4/06/2016	2.5	3
7	Abbotsford	16 Maugie St	4	h	NaN	SN	Nelson	6/08/2016	2.5	3
8	Abbotsford	53 Turner St	2	h	NaN	S	Biggin	6/08/2016	2.5	3
9	Abbotsford	99 Turner St	2	h	NaN	S	Collins	6/08/2016	2.5	3
10	Abbotsford	129 Charles St	2	h	941000.0	S	Jellis	7/05/2016	2.5	3
11	Abbotsford	124 Yarra St	3	h	1876000.0	S	Nelson	7/05/2016	2.5	3
12	Abbotsford	121/56 Nicholson St	2	u	NaN	PI	Biggin	7/11/2016	2.5	3
13	Abbotsford	17 Raphael St	4	h	NaN	W	Biggin	7/11/2016	2.5	3
14	Abbotsford	98 Charles St	2	h	1636000.0	S	Nelson	8/10/2016	2.5	3
15	Abbotsford	217 Langridge St	3	h	1000000.0	S	Jellis	8/10/2016	2.5	3
16	Abbotsford	18a Mollison St	2	t	745000.0	S	Jellis	8/10/2016	2.5	3
17	Abbotsford	6/241 Nicholson St	1	u	300000.0	S	Biggin	8/10/2016	2.5	3
18	Abbotsford	10 Valiant St	2	h	1097000.0	S	Biggin	8/10/2016	2.5	3
19	Abbotsford	403/609 Victoria St	2	u	542000.0	S	Dingle	8/10/2016	2.5	3

20 rows × 21 columns

Finding Unique Values

```
uniqueCounts = df.nunique();
```

In [82]:
```
print("Unique count across columns:")
print(uniqueCounts);
```

```
Unique count across columns:
Suburb            351
Address         34009
Rooms              12
Type                3
Price            2871
Method              9
SellerG           388
Date               78
Distance          215
Postcode          211
Bedroom2           15
Bathroom           11
Car                15
Landsize         1684
BuildingArea      740
YearBuilt         160
CouncilArea        33
Lattitude       13402
Longtitude      14524
Regionname          8
Propertycount     342
dtype: int64
```

Finding total number of null values

In [83]:
```
df.isnull().sum()
```

Out[83]:
```
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
```

Handling missing values using mean

In [84]:
```
df['Price'].fillna(value = df.Price.mean(), inplace = True)
df['Distance'].fillna(value = df.Distance.mean(), inplace = True)
df['Postcode'].fillna(value = df.Postcode.mean(), inplace = True)
df['Bedroom2'].fillna(value = df.Bedroom2.mean(), inplace = True)
df['Bathroom'].fillna(value = df.Bathroom.mean(), inplace = True)
df['Car'].fillna(value = df.Car.mean(), inplace = True)
df['Landsize'].fillna(value = df.Landsize.mean(), inplace = True)
df['Bedroom2'].fillna(value = df.Bedroom2.mean(), inplace = True)
```

```
df['YearBuilt'].fillna(value = df.YearBuilt.mean(), inplace = True)
df['Lattitude'].fillna(value = df.Lattitude.mean(), inplace = True)
df['Longtitude'].fillna(value = df.Longtitude.mean(), inplace = True)
df['Propertycount'].fillna(value = df.Propertycount.mean(), inplace = True)
df['BuildingArea'].fillna(value = df.BuildingArea.mean(), inplace = True)
```

In [85]:
```
df.isnull().sum()
```

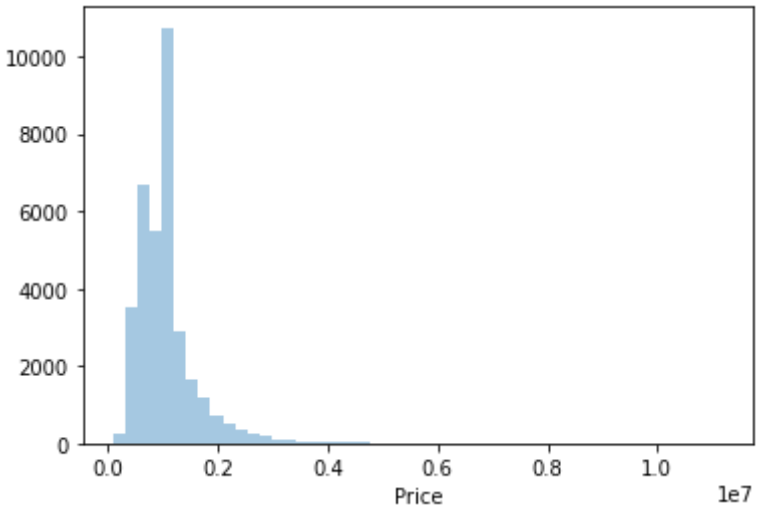
Out[85]:
```
Suburb          0
Address         0
Rooms           0
Type            0
Price           0
Method          0
SellerG         0
Date            0
Distance        0
Postcode        0
Bedroom2        0
Bathroom        0
Car             0
Landsize        0
BuildingArea    0
YearBuilt       0
CouncilArea     3
Lattitude       0
Longtitude      0
Regionname      3
Propertycount   0
dtype: int64
```

In [86]:
```
df = df.fillna(0)
```

Price before Scaling - Right skewed

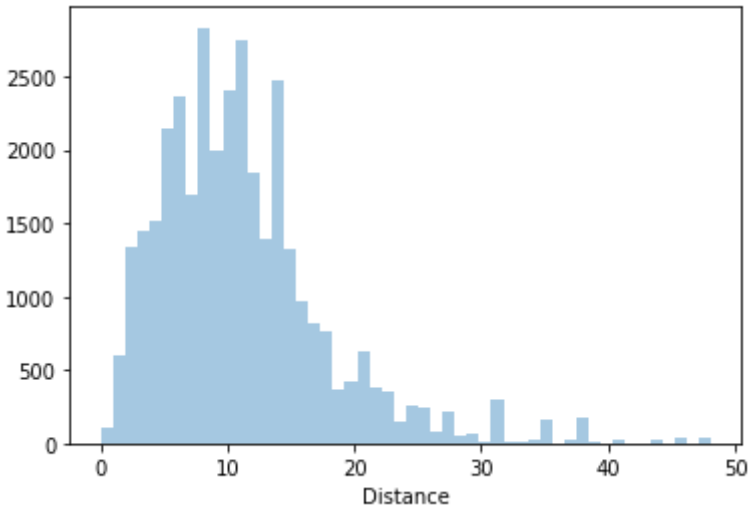
In [87]:
```
import seaborn as sb
from matplotlib import pyplot as plt
sb.distplot(df['Price'],kde = False)
plt.show()
```

```
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/seabor
n/distributions.py:2619: FutureWarning: `distplot` is a deprecated function
and will be removed in a future version. Please adapt your code to use eith
er `displot` (a figure-level function with similar flexibility) or `histplo
t` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



Distance before Scaling - Right skewed

In [88]:
```
import seaborn as sb
from matplotlib import pyplot as plt
sb.distplot(df['Distance'],kde = False)
plt.show()
```



Standard Scaler

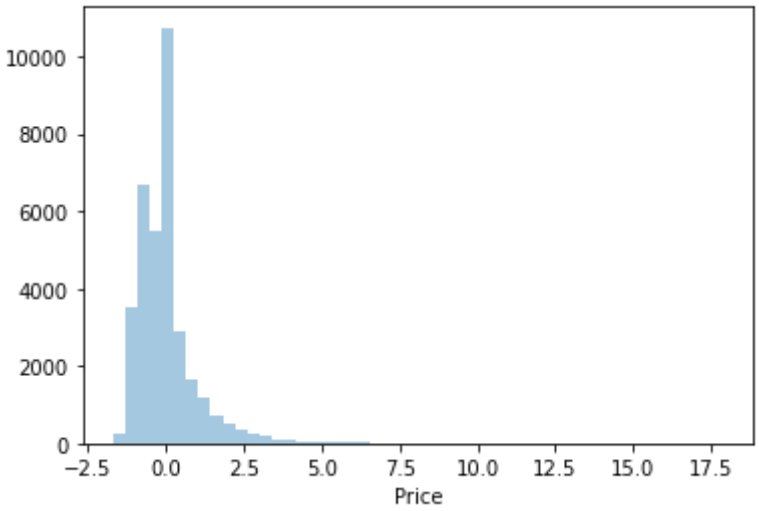
In [89]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

In [90]:
```
df[['Price','Distance','Landsize','Propertycount']] = scaler.fit_transform(
```

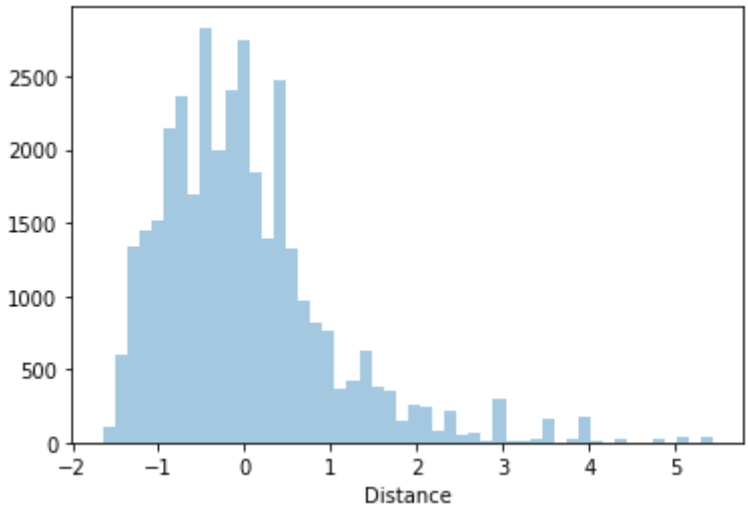
After Scaling

In [91]:
```
import seaborn as sb
from matplotlib import pyplot as plt
sb.distplot(df['Price'],kde = False)
plt.show()
```

```
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/seabor
n/distributions.py:2619: FutureWarning: `distplot` is a deprecated function
and will be removed in a future version. Please adapt your code to use eith
er `displot` (a figure-level function with similar flexibility) or `histplo
t` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [92]:
```
import seaborn as sb
from matplotlib import pyplot as plt
sb.distplot(df['Distance'],kde = False)
plt.show()
```



In [93]:
```
df.head(10)
```

Out[93]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Post
0	Abbotsford	68 Studley St	2	h	0.000000	SS	Jellis	3/09/2016	-1.279322	3
1	Abbotsford	85 Turner St	2	h	0.757901	S	Biggin	3/12/2016	-1.279322	3
2	Abbotsford	25 Bloomburg St	2	h	-0.026755	S	Biggin	4/02/2016	-1.279322	3
3	Abbotsford	18/659 Victoria St	3	u	0.000000	VB	Rounds	4/02/2016	-1.279322	3
4	Abbotsford	5 Charles St	3	h	0.731452	SP	Biggin	4/03/2017	-1.279322	3
5	Abbotsford	40 Federation La	3	h	-0.352960	PI	Biggin	4/03/2017	-1.279322	3
6	Abbotsford	55a Park St	4	h	0.969494	VB	Nelson	4/06/2016	-1.279322	3
7	Abbotsford	16 Maugie St	4	h	0.000000	SN	Nelson	6/08/2016	-1.279322	3
8	Abbotsford	53 Turner St	2	h	0.000000	S	Biggin	6/08/2016	-1.279322	3
9	Abbotsford	99 Turner St	2	h	0.000000	S	Collins	6/08/2016	-1.279322	3

10 rows × 21 columns

K Means Clustering

In [94]:
```
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split
```

In [95]:
```
from sklearn import preprocessing

label_encoder = preprocessing.LabelEncoder()
df['Type']= label_encoder.fit_transform(df['Type'])
```

In [96]:
```
df
```

Out[96]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Dis
0	Abbotsford	68 Studley St	2	0	0.000000	SS	Jellis	3/09/2016	-1.2
1	Abbotsford	85 Turner St	2	0	0.757901	S	Biggin	3/12/2016	-1.2
2	Abbotsford	25 Bloomburg St	2	0	-0.026755	S	Biggin	4/02/2016	-1.2
3	Abbotsford	18/659 Victoria St	3	2	0.000000	VB	Rounds	4/02/2016	-1.2
4	Abbotsford	5 Charles St	3	0	0.731452	SP	Biggin	4/03/2017	-1.2
...	...	...	...	...	...	...	...	...	
34852	Yarraville	13 Burns St	4	0	0.757901	PI	Jas	24/02/2018	-0.7
34853	Yarraville	29A Murray St	2	0	-0.285956	SP	Sweeney	24/02/2018	-0.7
34854	Yarraville	147A Severn St	2	1	-0.608634	S	Jas	24/02/2018	-0.7
34855	Yarraville	12/37 Stephen St	3	0	0.158389	SP	hockingstuart	24/02/2018	-0.7
34856	Yarraville	3 Tarrengower St	2	0	-0.053204	PI	RW	24/02/2018	-0.7

34857 rows × 21 columns

Type column is classified with the dataset

In [97]:
```
df = df.drop(['BuildingArea','YearBuilt','Bedroom2','Address','Postcode'],
```

In [98]:
```
df.loc[df.Lattitude.isnull(), 'Lattitude']  = df.groupby('Suburb')['Lattitu
df.loc[df.Longtitude.isnull(), 'Longtitude']  = df.groupby('Suburb')['Longt
```

In [99]:
```
df.drop(['Suburb','SellerG'], axis=1,inplace=True)
```

In [100…
```
df= pd.get_dummies(df, columns = ['Type','Method','CouncilArea','Regionname
```

In [101…
```
df.drop(['Date'], axis=1,inplace=True)
```

In [102…
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from scipy import stats
from sklearn.cluster import KMeans
```

```
from sklearn import metrics
from sklearn.preprocessing import scale
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge, RidgeCV, Lasso, LassoCV
from sklearn.metrics import mean_squared_error
%matplotlib inline
```

In [103…
```
df_c=df
df_c.drop(['Price'], axis=1,inplace=True)
```

In [104…
```
df_c.head()
```

Out[104]:

	Rooms	Distance	Bathroom	Car	Landsize	Lattitude	Longtitude	Propertycount	Type_0
0	2	-1.279322	1.0	1.0	-0.169196	-37.8014	144.9958	-0.802624	1
1	2	-1.279322	1.0	1.0	-0.141696	-37.7996	144.9984	-0.802624	1
2	2	-1.279322	1.0	0.0	-0.158341	-37.8079	144.9934	-0.802624	1
3	3	-1.279322	2.0	1.0	-0.214788	-37.8114	145.0116	-0.802624	0
4	3	-1.279322	2.0	0.0	-0.166301	-37.8093	144.9944	-0.802624	1

5 rows × 63 columns

In [106…
```
#df_imp=df_c[['Rooms', 'Car']]
#df_imp=df_c[['Propertycount', 'Rooms']]

#df_imp=df_c[['Distance', 'Propertycount']]
df_imp=df_c[['Lattitude','Longtitude']]
from sklearn import preprocessing

x = df_imp #returns a numpy array
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
df = pd.DataFrame(x_scaled)
```

Columns to cluster

In [107…
```
df.head()
```

Out[107]:

	0	1
0	0.486148	0.518802
1	0.488397	0.521160
2	0.478025	0.516625
3	0.473651	0.533132
4	0.476276	0.517532

In [117…
```
# Choosing the optimal k
from scipy.spatial.distance import cdist, pdist

k_range = range(1,14)
# Try clustering the data for k values ranging 1 to 10
k_means_var = [KMeans(n_clusters = k).fit(df) for k in k_range]
centroids = [X.cluster_centers_ for X in k_means_var]
```

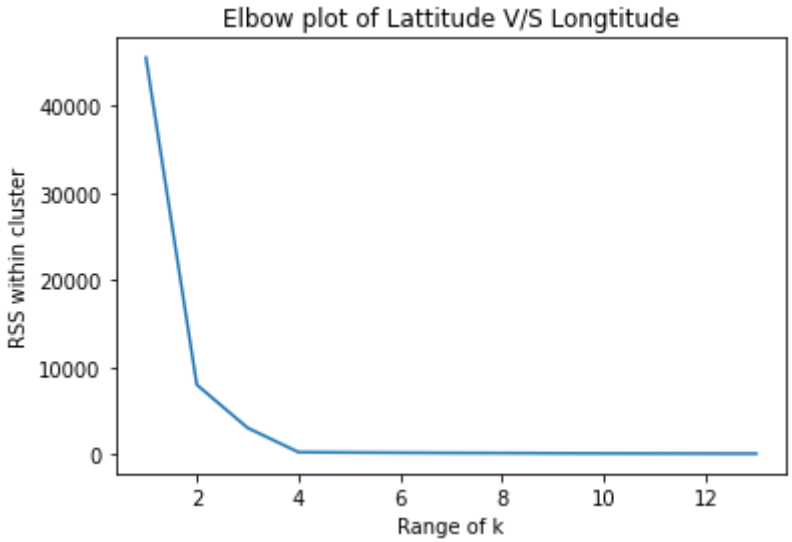
```
k_euclid = [cdist(df, cent, 'euclidean') for cent in centroids]
dist = [np.min(ke, axis=1) for ke in k_euclid]

# Calculate within-cluster sum of squares
wcss = [sum(d**2) for d in dist]

# Visualize the elbow method for determining k
import matplotlib.pyplot as plt
plt.plot(k_range, wcss)
plt.xlabel('Range of k')
plt.ylabel('RSS within cluster')
plt.title('Elbow plot of Lattitude V/S Longtitude')
plt.savefig('Elbow_5')
plt.show()
```

```
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/sklear
n/utils/validation.py:1858: FutureWarning: Feature names only support names
that are all strings. Got feature names with dtypes: ['int', 'str']. An err
or will be raised in 1.2.
  warnings.warn(
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/sklear
n/utils/validation.py:1858: FutureWarning: Feature names only support names
that are all strings. Got feature names with dtypes: ['int', 'str']. An err
or will be raised in 1.2.
  warnings.warn(
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/sklear
n/utils/validation.py:1858: FutureWarning: Feature names only support names
that are all strings. Got feature names with dtypes: ['int', 'str']. An err
or will be raised in 1.2.
  warnings.warn(
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/sklear
n/utils/validation.py:1858: FutureWarning: Feature names only support names
that are all strings. Got feature names with dtypes: ['int', 'str']. An err
or will be raised in 1.2.
  warnings.warn(
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/sklear
n/utils/validation.py:1858: FutureWarning: Feature names only support names
that are all strings. Got feature names with dtypes: ['int', 'str']. An err
or will be raised in 1.2.
  warnings.warn(
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/sklear
n/utils/validation.py:1858: FutureWarning: Feature names only support names
that are all strings. Got feature names with dtypes: ['int', 'str']. An err
or will be raised in 1.2.
  warnings.warn(
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/sklear
n/utils/validation.py:1858: FutureWarning: Feature names only support names
that are all strings. Got feature names with dtypes: ['int', 'str']. An err
or will be raised in 1.2.
  warnings.warn(
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/sklear
n/utils/validation.py:1858: FutureWarning: Feature names only support names
that are all strings. Got feature names with dtypes: ['int', 'str']. An err
or will be raised in 1.2.
  warnings.warn(
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/sklear
n/utils/validation.py:1858: FutureWarning: Feature names only support names
that are all strings. Got feature names with dtypes: ['int', 'str']. An err
or will be raised in 1.2.
  warnings.warn(
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/sklear
n/utils/validation.py:1858: FutureWarning: Feature names only support names
that are all strings. Got feature names with dtypes: ['int', 'str']. An err
or will be raised in 1.2.
  warnings.warn(
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/sklear
n/utils/validation.py:1858: FutureWarning: Feature names only support names
that are all strings. Got feature names with dtypes: ['int', 'str']. An err
or will be raised in 1.2.
  warnings.warn(
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/sklear
n/utils/validation.py:1858: FutureWarning: Feature names only support names
that are all strings. Got feature names with dtypes: ['int', 'str']. An err
or will be raised in 1.2.
  warnings.warn(
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/sklear
n/utils/validation.py:1858: FutureWarning: Feature names only support names
that are all strings. Got feature names with dtypes: ['int', 'str']. An err
```

```
or will be raised in 1.2.
  warnings.warn(
```

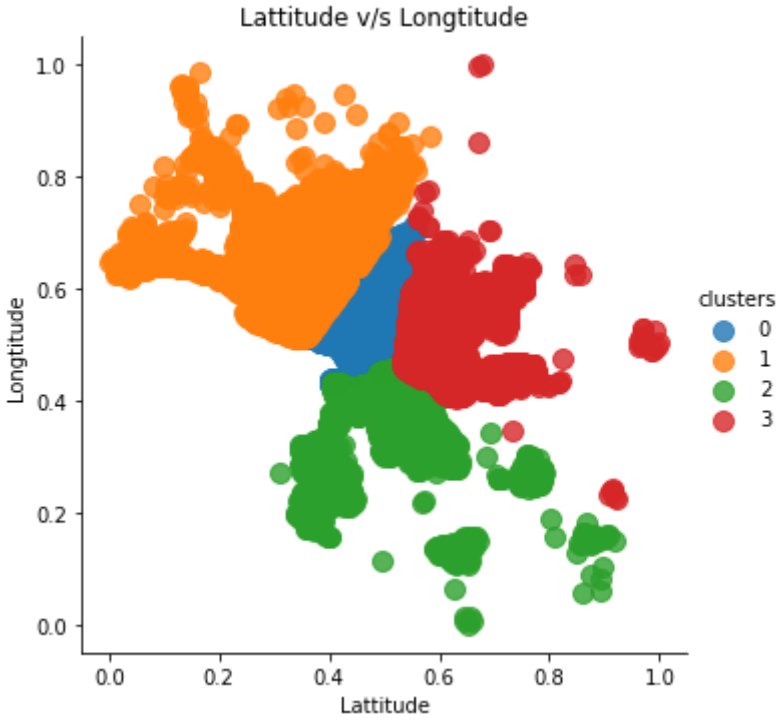


In [113…
```
kmeans = KMeans(n_clusters=4, random_state=0).fit(df)
labels = kmeans.labels_
#Glue back to originaal data
df['clusters'] = labels
df2 = df.rename(columns = {0 : 'Lattitude', 1: 'Longtitude'})
#Add the column into our list
```

```
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/sklear
n/utils/validation.py:1858: FutureWarning: Feature names only support names
that are all strings. Got feature names with dtypes: ['int', 'str']. An err
or will be raised in 1.2.
  warnings.warn(
```

In [140…
```
sns.lmplot('Lattitude', 'Longtitude', data = df2, fit_reg=False,hue="cluste
plt.title('Lattitude v/s Longtitude')
plt.xlabel('Lattitude')
plt.ylabel('Longtitude')
plt.savefig('cluster_5.png')
plt.show()
```

```
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/seabor
n/_decorators.py:36: FutureWarning: Pass the following variables as keyword
args: x, y. 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 misinterpretation.
  warnings.warn(
```



Inference

In the above housing data set I have taken Latitude and Longitude as the class labels for clustering the data and found out that K value as 4 would be the optimum cluster based on the inference provided by the elbow graph

Based on the above graph the houses can be clustered among the 4 clusters

In [ ]: