test

February 19, 2024

[50]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

[67]:
```
df = pd.read_csv('../store/melb_data.csv')
```

0.0.1 1: Display information of the data

[14]:
```
# Size
print("Data Size: ", df.size)
```

```
Data Size:  285180
```

[15]:
```
# Shape
print("Data Shape: ", df.shape)
```

```
Data Shape:  (13580, 21)
```

[16]:
```
# Number of Dimensions
print("Number of Dimensions: ", df.ndim)
```

```
Number of Dimensions:  2
```

[53]:
```
# Overview information
print("Information: ", df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13580 entries, 0 to 13579
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Suburb         13580 non-null  object
 1   Address        13580 non-null  object
 2   Rooms          13580 non-null  int64
 3   Type           13580 non-null  object
 4   Price          13580 non-null  float64
 5   Method         13580 non-null  object
 6   SellerG        13580 non-null  object
```

```
 7   Date           13580 non-null  object
 8   Distance       13580 non-null  float64
 9   Postcode       13580 non-null  float64
 10  Bedroom2       13580 non-null  float64
 11  Bathroom       13580 non-null  float64
 12  Car            13518 non-null  float64
 13  Landsize       13580 non-null  float64
 14  BuildingArea   7130 non-null   float64
 15  YearBuilt      8205 non-null   float64
 16  CouncilArea    12211 non-null  object
 17  Lattitude      13580 non-null  float64
 18  Longtitude     13580 non-null  float64
 19  Regionname     13580 non-null  object
 20  Propertycount  13580 non-null  float64
dtypes: float64(12), int64(1), object(8)
memory usage: 2.2+ MB
Information:  None
```

0.0.2 2. Display Statistics

[20]:
```
# All attributes
df.describe()
```

[20]:

	Rooms	Price	Distance	Postcode	Bedroom2	\
count	13580.000000	1.358000e+04	13580.000000	13580.000000	13580.000000	
mean	2.937997	1.075684e+06	10.137776	3105.301915	2.914728	
std	0.955748	6.393107e+05	5.868725	90.676964	0.965921	
min	1.000000	8.500000e+04	0.000000	3000.000000	0.000000	
25%	2.000000	6.500000e+05	6.100000	3044.000000	2.000000	
50%	3.000000	9.030000e+05	9.200000	3084.000000	3.000000	
75%	3.000000	1.330000e+06	13.000000	3148.000000	3.000000	
max	10.000000	9.000000e+06	48.100000	3977.000000	20.000000	

	Bathroom	Car	Landsize	BuildingArea	YearBuilt	\
count	13580.000000	13518.000000	13580.000000	7130.000000	8205.000000	
mean	1.534242	1.610075	558.416127	151.967650	1964.684217	
std	0.691712	0.962634	3990.669241	541.014538	37.273762	
min	0.000000	0.000000	0.000000	0.000000	1196.000000	
25%	1.000000	1.000000	177.000000	93.000000	1940.000000	
50%	1.000000	2.000000	440.000000	126.000000	1970.000000	
75%	2.000000	2.000000	651.000000	174.000000	1999.000000	
max	8.000000	10.000000	433014.000000	44515.000000	2018.000000	

	Lattitude	Longtitude	Propertycount
count	13580.000000	13580.000000	13580.000000
mean	-37.809203	144.995216	7454.417378
std	0.079260	0.103916	4378.581772

```
min      -38.182550    144.431810     249.000000
25%      -37.856822    144.929600    4380.000000
50%      -37.802355    145.000100    6555.000000
75%      -37.756400    145.058305   10331.000000
max      -37.408530    145.526350   21650.000000
```

[34]:
```
# Select attribute: Price, Landsize, Propertycount
df_3col = df.loc[:, ['Price', 'Landsize', 'Propertycount']]
df_3col.describe()
```

[34]:
```
              Price       Landsize  Propertycount
count  1.358000e+04   13580.000000   13580.000000
mean   1.075684e+06     558.416127    7454.417378
std    6.393107e+05    3990.669241    4378.581772
min    8.500000e+04       0.000000     249.000000
25%    6.500000e+05     177.000000    4380.000000
50%    9.030000e+05     440.000000    6555.000000
75%    1.330000e+06     651.000000   10331.000000
max    9.000000e+06  433014.000000   21650.000000
```

[35]:
```
# Select attribute with Landsize > 500
df_cond = df.loc[(df['Landsize'] < 500)]
df_cond.describe()
```

[35]:
```
             Rooms         Price     Distance     Postcode     Bedroom2  \
count  7392.000000  7.392000e+03  7392.000000  7392.000000  7392.000000
mean      2.587798  9.484325e+05     8.131710  3098.894210     2.566558
std       0.822415  5.065054e+05     4.713157    79.210585     0.817484
min       1.000000  8.500000e+04     0.000000  3000.000000     0.000000
25%       2.000000  5.900000e+05     4.600000  3042.000000     2.000000
50%       3.000000  8.350000e+05     7.500000  3073.000000     3.000000
75%       3.000000  1.205125e+06    11.200000  3146.000000     3.000000
max      10.000000  5.700000e+06    47.300000  3977.000000    10.000000

          Bathroom          Car     Landsize  BuildingArea    YearBuilt  \
count  7392.000000  7342.000000  7392.000000   4064.000000  4741.000000
mean      1.429654     1.288069   197.216585    119.261818  1965.291500
std       0.597363     0.748220   155.069985     66.255755    41.101236
min       0.000000     0.000000     0.000000      0.000000  1850.000000
25%       1.000000     1.000000     0.000000     81.000000  1930.000000
50%       1.000000     1.000000   195.000000    109.000000  1970.000000
75%       2.000000     2.000000   318.000000    144.000000  2003.000000
max       7.000000     7.000000   499.000000   1561.000000  2018.000000

         Lattitude   Longtitude  Propertycount
count  7392.000000  7392.000000    7392.000000
mean    -37.810351   144.982682    7712.066694
```

```
std       0.065692     0.079523    4328.118170
min     -38.164390   144.568870     394.000000
25%     -37.850592   144.930575    4675.000000
50%     -37.808200   144.989105    6786.000000
75%     -37.766300   145.029315   10579.000000
max     -37.491750   145.453760   21650.000000
```

[37]:
```
# Select attribute with Bedroom2 = 2 AND Bathroom = 1 AND Car = 1
df_cond2 = df.loc[(df['Bedroom2'] == 2) & (df['Bathroom'] == 1) & (df['Car'] ==␣
 ↪1)]
df_cond2.describe()
```

[37]:
```
              Rooms         Price     Distance     Postcode  Bedroom2  \
count  2137.000000  2.137000e+03  2137.000000  2137.000000    2137.0
mean      2.026205  6.794727e+05     8.056575  3100.423023       2.0
std       0.194171  2.908005e+05     4.304758    67.580018       0.0
min       1.000000  1.450000e+05     0.000000  3000.000000       2.0
25%       2.000000  4.900000e+05     5.100000  3046.000000       2.0
50%       2.000000  6.110000e+05     7.700000  3081.000000       2.0
75%       2.000000  7.900000e+05    11.100000  3147.000000       2.0
max       4.000000  2.905000e+06    41.000000  3910.000000       2.0

       Bathroom     Car      Landsize  BuildingArea    YearBuilt    Lattitude  \
count    2137.0  2137.0   2137.000000   1142.000000  1416.000000  2137.000000
mean        1.0     1.0    348.408985     84.769562  1969.518362   -37.811354
std         0.0     0.0    963.376076     44.334332    31.775777     0.063619
min         1.0     1.0      0.000000      0.000000  1830.000000   -38.164390
25%         1.0     1.0      0.000000     69.000000  1960.000000   -37.854700
50%         1.0     1.0    132.000000     80.000000  1970.000000   -37.809150
75%         1.0     1.0    305.000000     94.000000  1995.000000   -37.765900
max         1.0     1.0  17200.000000   1143.000000  2016.000000   -37.570630

        Longtitude  Propertycount
count  2137.000000    2137.000000
mean    144.991409    7854.260178
std       0.072949    4576.352268
min     144.571590     438.000000
25%     144.943500    4605.000000
50%     144.995710    6938.000000
75%     145.037900   10412.000000
max     145.335010   21650.000000
```

[38]:
```
# Select attribuite with a specific condition (all conditions together):
df_allcond = df.loc[(df['Landsize'] < 500) & (df['Bedroom2'] == 2) &␣
 ↪(df['Bathroom'] == 1) & (df['Car'] == 1)]
df_allcond.describe()
```

[38]:
```
              Rooms         Price     Distance     Postcode  Bedroom2  \
count  1749.000000  1.749000e+03  1749.000000  1749.000000    1749.0
mean      2.026872  6.693860e+05     8.018754  3098.251001       2.0
std       0.202582  2.784590e+05     4.310617    69.396890       0.0
min       1.000000  1.450000e+05     0.000000  3000.000000       2.0
25%       2.000000  4.825000e+05     5.100000  3046.000000       2.0
50%       2.000000  6.100000e+05     7.700000  3078.000000       2.0
75%       2.000000  7.800000e+05    11.200000  3146.000000       2.0
max       4.000000  2.905000e+06    41.000000  3910.000000       2.0

       Bathroom     Car     Landsize  BuildingArea    YearBuilt    Lattitude  \
count    1749.0  1749.0  1749.000000    956.000000  1176.000000  1749.000000
mean        1.0     1.0   106.315609     84.622228  1969.389456   -37.809896
std         0.0     0.0   122.028427     44.034837    32.794572     0.063897
min         1.0     1.0     0.000000      0.000000  1880.000000   -38.164390
25%         1.0     1.0     0.000000     69.000000  1960.000000   -37.852580
50%         1.0     1.0    85.000000     80.000000  1970.000000   -37.806350
75%         1.0     1.0   177.000000     94.000000  1996.000000   -37.764300
max         1.0     1.0   499.000000   1143.000000  2016.000000   -37.570630

        Longtitude  Propertycount
count  1749.000000    1749.000000
mean    144.988234    7874.824471
std       0.072346    4619.813212
min     144.571590     438.000000
25%     144.940000    4675.000000
50%     144.993300    6938.000000
75%     145.034800   10412.000000
max     145.292840   21650.000000
```

0.0.3 3. Inspect if there are any missing values.

[54]:
```
# 3.1 Remove rows that contain missing value
row_remove = df.dropna()
print("Shape: ", row_remove.shape)
row_remove.info()
```

```
Shape:  (6196, 21)
<class 'pandas.core.frame.DataFrame'>
Index: 6196 entries, 1 to 12212
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Suburb         6196 non-null   object
 1   Address        6196 non-null   object
 2   Rooms          6196 non-null   int64
 3   Type           6196 non-null   object
```

```
 4   Price          6196 non-null   float64
 5   Method         6196 non-null   object
 6   SellerG        6196 non-null   object
 7   Date           6196 non-null   object
 8   Distance       6196 non-null   float64
 9   Postcode       6196 non-null   float64
 10  Bedroom2       6196 non-null   float64
 11  Bathroom       6196 non-null   float64
 12  Car            6196 non-null   float64
 13  Landsize       6196 non-null   float64
 14  BuildingArea   6196 non-null   float64
 15  YearBuilt      6196 non-null   float64
 16  CouncilArea    6196 non-null   object
 17  Lattitude      6196 non-null   float64
 18  Longtitude     6196 non-null   float64
 19  Regionname     6196 non-null   object
 20  Propertycount  6196 non-null   float64
dtypes: float64(12), int64(1), object(8)
memory usage: 1.0+ MB
```

[47]:
```
# 3.2 Replace missing value with zeros
fill_zero = df.fillna(0)
print("Shape: ", fill_zero.shape)
fill_zero.describe()
```

```
Shape:  (13580, 21)
```

[47]:

	Rooms	Price	Distance	Postcode	Bedroom2	\
count	13580.000000	1.358000e+04	13580.000000	13580.000000	13580.000000	
mean	2.937997	1.075684e+06	10.137776	3105.301915	2.914728	
std	0.955748	6.393107e+05	5.868725	90.676964	0.965921	
min	1.000000	8.500000e+04	0.000000	3000.000000	0.000000	
25%	2.000000	6.500000e+05	6.100000	3044.000000	2.000000	
50%	3.000000	9.030000e+05	9.200000	3084.000000	3.000000	
75%	3.000000	1.330000e+06	13.000000	3148.000000	3.000000	
max	10.000000	9.000000e+06	48.100000	3977.000000	20.000000	

	Bathroom	Car	Landsize	BuildingArea	YearBuilt	\
count	13580.000000	13580.000000	13580.000000	13580.000000	13580.000000	
mean	1.534242	1.602725	558.416127	79.788611	1187.056996	
std	0.691712	0.966548	3990.669241	399.281619	961.246692	
min	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	1.000000	177.000000	0.000000	0.000000	
50%	1.000000	2.000000	440.000000	51.000000	1925.000000	
75%	2.000000	2.000000	651.000000	129.940000	1975.000000	
max	8.000000	10.000000	433014.000000	44515.000000	2018.000000	

```
          Lattitude    Longtitude  Propertycount
count  13580.000000  13580.000000   13580.000000
mean     -37.809203    144.995216    7454.417378
std        0.079260      0.103916    4378.581772
min      -38.182550    144.431810     249.000000
25%      -37.856822    144.929600    4380.000000
50%      -37.802355    145.000100    6555.000000
75%      -37.756400    145.058305   10331.000000
max      -37.408530    145.526350   21650.000000
```

[51]:
```
# Calculate the mean of every attribute in fill_zero and df
fill_zero_mean = fill_zero.select_dtypes(include=np.number).mean()
df_mean = df.select_dtypes(include=np.number).mean()

mean_comparison = pd.DataFrame({'fill_zero': fill_zero_mean, 'ori_df': df_mean})
print(mean_comparison)
```

```
                fill_zero        ori_df
Rooms        2.937997e+00  2.937997e+00
Price        1.075684e+06  1.075684e+06
Distance     1.013778e+01  1.013778e+01
Postcode     3.105302e+03  3.105302e+03
Bedroom2     2.914728e+00  2.914728e+00
Bathroom     1.534242e+00  1.534242e+00
Car          1.602725e+00  1.610075e+00
Landsize     5.584161e+02  5.584161e+02
BuildingArea 7.978861e+01  1.519676e+02
YearBuilt    1.187057e+03  1.964684e+03
Lattitude   -3.780920e+01 -3.780920e+01
Longtitude   1.449952e+02  1.449952e+02
Propertycount 7.454417e+03  7.454417e+03
```

ANS: Filling Zero shoud not be use, since some data such as Average Building Area is largely differs than the original data. Thus, the meaning of average is not truly indicates the real value of it.

[77]:
```
# 3.3 Replace missing value on non-numeric column with zeros, and on numeric␣
 ↪column with mean.
data_imp = df

for column in data_imp.columns:
    if data_imp[column].dtype == np.number:
        data_imp[column] = data_imp[column].fillna(data_imp[column].mean())
    else:
        data_imp[column] = data_imp[column].fillna(0)

data_imp_mean = data_imp.select_dtypes(include=np.number).mean()
df_mean = df.select_dtypes(include=np.number).mean()
```

```
mean_comparison2 = pd.DataFrame({'data_imp': data_imp_mean, 'ori_df': df_mean})
print(mean_comparison2)
```

```
                 data_imp        ori_df
Rooms          2.937997e+00  2.937997e+00
Price          1.075684e+06  1.075684e+06
Distance       1.013778e+01  1.013778e+01
Postcode       3.105302e+03  3.105302e+03
Bedroom2       2.914728e+00  2.914728e+00
Bathroom       1.534242e+00  1.534242e+00
Car            1.610075e+00  1.610075e+00
Landsize       5.584161e+02  5.584161e+02
BuildingArea   1.519676e+02  1.519676e+02
YearBuilt      1.964684e+03  1.964684e+03
Lattitude     -3.780920e+01 -3.780920e+01
Longtitude     1.449952e+02  1.449952e+02
Propertycount  7.454417e+03  7.454417e+03
```

```
/var/folders/p1/5vynmg61091ch5xrnfnzcgmr0000gn/T/ipykernel_97862/2981099932.py:5
: DeprecationWarning: Converting `np.inexact` or `np.floating` to a dtype is
deprecated. The current result is `float64` which is not strictly correct.
  if data_imp[column].dtype == np.number:
```

[81]:
```
data_imp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13580 entries, 0 to 13579
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Suburb         13580 non-null  object
 1   Address        13580 non-null  object
 2   Rooms          13580 non-null  int64
 3   Type           13580 non-null  object
 4   Price          13580 non-null  float64
 5   Method         13580 non-null  object
 6   SellerG        13580 non-null  object
 7   Date           13580 non-null  object
 8   Distance       13580 non-null  float64
 9   Postcode       13580 non-null  float64
 10  Bedroom2       13580 non-null  float64
 11  Bathroom       13580 non-null  float64
 12  Car            13580 non-null  float64
 13  Landsize       13580 non-null  float64
 14  BuildingArea   13580 non-null  float64
 15  YearBuilt      13580 non-null  float64
 16  CouncilArea    13580 non-null  object
 17  Lattitude      13580 non-null  float64
```

```
 18  Longtitude     13580 non-null  float64
 19  Regionname     13580 non-null  object
 20  Propertycount  13580 non-null  float64
dtypes: float64(12), int64(1), object(8)
memory usage: 2.2+ MB
```

ANS: Data Imputation shoud be use, since every mean of data stay the same from the original data.

0.0.4 4. Format the datetime of the attribute to YYYY/MM/DD

[82]:
```
data_imp["Date"] = pd.to_datetime(data_imp["Date"], format='%d/%m/%Y')
data_imp["Date"] = data_imp['Date'].dt.strftime('%Y/%m/%d')

data_imp
```

[82]:

	Suburb	Address	Rooms	Type	Price	Method	\
0	Abbotsford	85 Turner St	2	h	1480000.0	S	
1	Abbotsford	25 Bloomburg St	2	h	1035000.0	S	
2	Abbotsford	5 Charles St	3	h	1465000.0	SP	
3	Abbotsford	40 Federation La	3	h	850000.0	PI	
4	Abbotsford	55a Park St	4	h	1600000.0	VB	
…	…	…	…	…	…	…	
13575	Wheelers Hill	12 Strada Cr	4	h	1245000.0	S	
13576	Williamstown	77 Merrett Dr	3	h	1031000.0	SP	
13577	Williamstown	83 Power St	3	h	1170000.0	S	
13578	Williamstown	96 Verdon St	4	h	2500000.0	PI	
13579	Yarraville	6 Agnes St	4	h	1285000.0	SP	

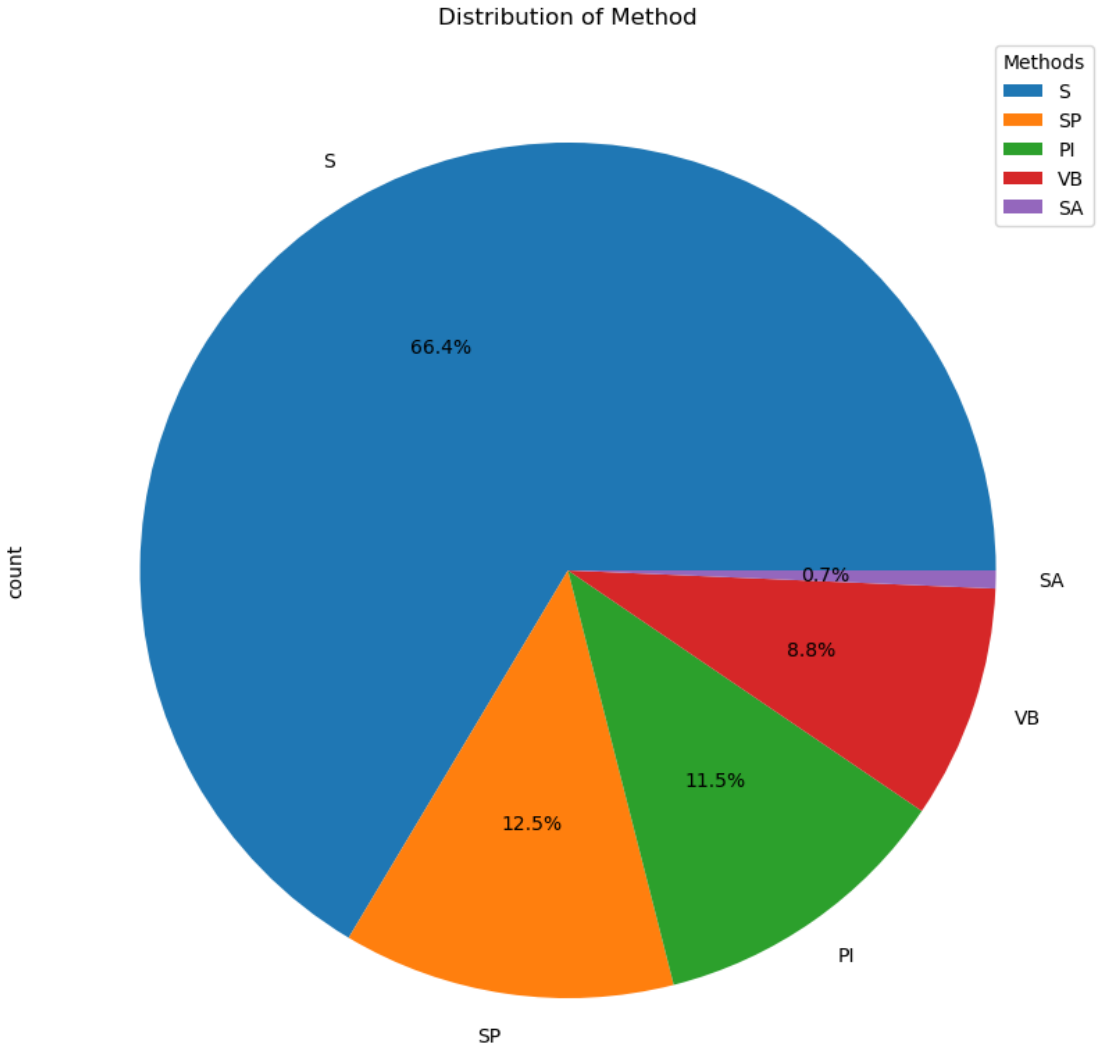
	SellerG	Date	Distance	Postcode	…	Bathroom	Car	Landsize	\
0	Biggin	2016/12/03	2.5	3067.0	…	1.0	1.0	202.0	
1	Biggin	2016/02/04	2.5	3067.0	…	1.0	0.0	156.0	
2	Biggin	2017/03/04	2.5	3067.0	…	2.0	0.0	134.0	
3	Biggin	2017/03/04	2.5	3067.0	…	2.0	1.0	94.0	
4	Nelson	2016/06/04	2.5	3067.0	…	1.0	2.0	120.0	
…	…	…	…	…	…	…	…	…	
13575	Barry	2017/08/26	16.7	3150.0	…	2.0	2.0	652.0	
13576	Williams	2017/08/26	6.8	3016.0	…	2.0	2.0	333.0	
13577	Raine	2017/08/26	6.8	3016.0	…	2.0	4.0	436.0	
13578	Sweeney	2017/08/26	6.8	3016.0	…	1.0	5.0	866.0	
13579	Village	2017/08/26	6.3	3013.0	…	1.0	1.0	362.0	

	BuildingArea	YearBuilt	CouncilArea	Lattitude	Longtitude	\
0	151.96765	1964.684217	Yarra	-37.79960	144.99840	
1	79.00000	1900.000000	Yarra	-37.80790	144.99340	
2	150.00000	1900.000000	Yarra	-37.80930	144.99440	
3	151.96765	1964.684217	Yarra	-37.79690	144.99690	
4	142.00000	2014.000000	Yarra	-37.80720	144.99410	

```
…              …           …           …        …          …
13575     151.96765  1981.000000           0 -37.90562   145.16761
13576     133.00000  1995.000000           0 -37.85927   144.87904
13577     151.96765  1997.000000           0 -37.85274   144.88738
13578     157.00000  1920.000000           0 -37.85908   144.89299
13579     112.00000  1920.000000           0 -37.81188   144.88449

                      Regionname Propertycount
0           Northern Metropolitan        4019.0
1           Northern Metropolitan        4019.0
2           Northern Metropolitan        4019.0
3           Northern Metropolitan        4019.0
4           Northern Metropolitan        4019.0
…                             …           …
13575  South-Eastern Metropolitan        7392.0
13576        Western Metropolitan        6380.0
13577        Western Metropolitan        6380.0
13578        Western Metropolitan        6380.0
13579        Western Metropolitan        6543.0

[13580 rows x 21 columns]
```

0.0.5 5. Create a pie chart to demonstrate unique values of the attribute Method.

[79]:
```
method_counts = data_imp['Method'].value_counts()

method_counts.plot.pie(autopct='%1.1f%%')
plt.title('Distribution of Method')
plt.legend(title="Methods", loc="best")
plt.show()
```


Distribution of Method
Methods
S
SP
PI
VB
SA
count
66.4%
12.5%
11.5%
8.8%
0.7%