In [150…

```python
import pandas as pd
import seaborn as sns
import numpy as np
```

In [151…

```python
# Description of Dataset
# The dataset contains data Dataset includes info about real estate objects in Moscow.
#The following are a discription of the data
# The data is from www.kaggle.com, the url is https://www.kaggle.com/timmofeyy/realestate

# "metro": The nearest metro stationto to the apartment.
# "price": The rent price for the apartment.
#"way": A way to reach metro station (on foot or by public transport)
#"views": The number of views for each apartment.
#"provider": A person or agency who is renting apartment.
#"fee_percent": Fee percent of an agency or realtor
#"storey": tThe storey, where the apartment located.
#"minutes": Minutes quantity to reach metro station
#"storey": The total number of storeys in a building.
```

In [152…

```python
data = pd.read_csv('move.csv')
```

In [153…

```python
#Exploratory Data Analysis
#There is no noll value in the data
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1446 entries, 0 to 1445
Data columns (total 13 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   Unnamed: 0    1446 non-null   int64 
 1   metro         1446 non-null   object
 2   price         1446 non-null   int64 
 3   way           1446 non-null   object
 4   views         1446 non-null   int64 
 5   provider      1446 non-null   object
 6   fee_percent   1446 non-null   int64 
 7   storey        1446 non-null   int64 
 8   minutes       1446 non-null   int64 
 9   storeys       1446 non-null   int64 
 10  living_area   1446 non-null   int64 
 11  kitchen_area  1446 non-null   int64 
 12  total_area    1446 non-null   int64 
dtypes: int64(10), object(3)
memory usage: 147.0+ KB
```

In [154…

```python
# The following are plans for data exploration
#1. A visual exploration
#2. Checking for null values
#3. Checking and dealing with outliers
#4. Checking and fixing typographical errors and also transalation issues

#5.
data.head()
```

Out[154…

Unnamed: 0	metro	price	way	views	provider	fee_percent	storey	minutes	storeys	liv

	Unnamed: 0	metro	price	way	views	provider	fee_percent	storey	minutes	storeys	liv
0	0	Planernaia	45000	walk	513	realtor	50	7	10	12	
1	1	VDNKh	50000	walk	389	realtor	50	16	10	16	
2	2	Alekseevskaia	50000	walk	483	realtor	50	5	3	12	
3	3	Sviblovo	38000	walk	414	realtor	50	3	15	5	
4	4	Rimskaia	55999	walk	360	realtor	99	6	7	17	

In [155…

```
data_raw = data.copy()
# This removes the unnamed column
data.drop(data.columns[data.columns.str.contains('Unnamed', case = False)], axis = 1, inp
data.head()
```

Out[155…

	metro	price	way	views	provider	fee_percent	storey	minutes	storeys	living_area	k
0	Planernaia	45000	walk	513	realtor	50	7	10	12	19	
1	VDNKh	50000	walk	389	realtor	50	16	10	16	18	
2	Alekseevskaia	50000	walk	483	realtor	50	5	3	12	19	
3	Sviblovo	38000	walk	414	realtor	50	3	15	5	37	
4	Rimskaia	55999	walk	360	realtor	99	6	7	17	21	

In [157…

```
# A description of the data to look for outliers
data.describe()
```

Out[157…

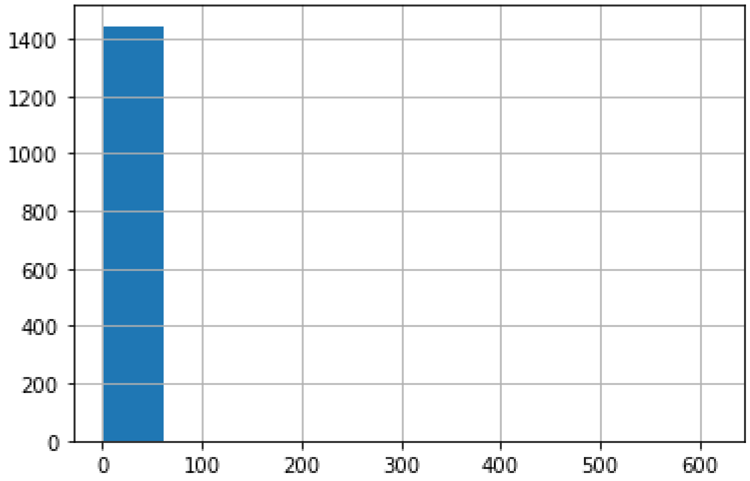
	price	views	fee_percent	storey	minutes	storeys	living_are
count	1446.000000	1446.000000	1446.000000	1446.000000	1446.000000	1446.000000	1446.00000
mean	43770.738589	417.917012	37.949516	7.089903	8.753804	22.545643	20.58575
std	33232.151532	936.532913	26.893347	16.511552	4.710759	347.279854	5.60899
min	14000.000000	4.000000	0.000000	1.000000	0.000000	1.000000	6.00000
25%	29000.000000	38.000000	0.000000	4.000000	5.000000	9.000000	18.00000
50%	38000.000000	103.000000	50.000000	6.000000	7.000000	12.000000	20.00000
75%	45000.000000	414.000000	50.000000	9.000000	12.000000	16.000000	21.00000
max	500000.000000	5174.000000	100.000000	613.000000	47.000000	13217.000000	37.00000

In [156…

```
# A summary statistics may reveal some outliers. There are outliers in storey and storeys
#Any storey or storeys value greater than 35  will be dropped
# Let us plot a histogram for storey and storeys
data['storey'].hist()
```

Out[156…

```
<AxesSubplot:>
```



In [ ]:

In [114…

```
data = data.loc[data['storey'] <= 35,:]
data = data.loc[data['storeys'] <= 35,:]
#Two rows of data were removed
```

In [158…

```
# Let me check the unique values while looking for error in spellings

data.nunique()
```

Out[158…

```
metro            119
price             62
way                2
views             93
provider           7
fee_percent       16
storey            29
minutes           27
storeys           36
living_area       25
kitchen_area      18
total_area        36
dtype: int64
```

In [116…

```
# For each column, I will then look at the unique values
data['metro'].unique()
```

Out[116…

```
array([' Planernaia ', ' VDNKh ', ' Alekseevskaia ', ' Sviblovo ',
       ' Rimskaia ', ' Perovo ', ' Nekrasovka ', ' Riazanskii prospekt ',
       ' Medvedkovo ', ' Khovrino ', ' Okskaia ', ' Vystavochnaia ',
       ' Otradnoe ', ' Kuntcevskaia ', ' Shabolovskaia ',
       ' Dobryninskaia ', ' Paveletckaia ', ' Altufevo ', ' Tcaritcyno ',
       ' Shchelkovskaia ', ' Skhodnenskaia ', ' Solntcevo ',
       ' Ulitca Starokachalovskaia ', ' Zhulebino ',
       ' Preobrazhenskaia ploshchad ', ' Rasskazovka ',
       ' Buninskaia Alleia ', ' Fili ', ' Kommunarka ',
       ' Lukhmanovskaia ', ' Teplyi Stan ', ' Prazhskaia ',
       ' Filatov Lug ', ' Annino ', ' Beliaevo ', ' Liublino ',
       ' Kuzminki ', ' Novye Cheremushki ', ' Marino ', ' Strogino ',
       ' Salarevo ', ' Piatnitckoe shosse ', ' Izmailovskaia ',
       ' Petrovsko-Razumovskaia ', ' Tekstilshchiki ', ' Novokosino ',
       ' Ulitca Dmitrievskogo ', ' Nagornaia ', ' Dubrovka ',
       ' Partizanskaia ', ' Bulvar Rokossovskogo ', ' Petrovskii park ',
Loading [MathJax]/extensions/Safe.js e Opolchenie ', ' Mitino ', ' Studencheskaia ',
       ' Bulvar Admirala Ushakova ', ' Krasnogvardeiskaia ',
       ' Kantemirovskaia ', ' Vodnyi stadion ', ' Kurskaia ',
       ' Borisovo ', ' Tcvetnoi bulvar ', ' Elektrozavodskaia ',
       ' Seligerskaia ', ' Rechnoi vokzal ', ' Prospekt Mira ',
       ' Belorusskaia ', ' Prospekt Vernadskogo ', ' Bratislavskaia ',
       ' Volzhskaia ', ' Kotelniki ', ' Okruzhnaia ', ' Krasnye vorota ',
       ' Belomorskaia ', ' Nakhimovskii prospekt ', ' Spartak ',
       ' Govorovo ', ' Iasenevo ', ' Tulskaia ', ' Krasnoselskaia ',
       ' Vladykino ', ' Shelepikha ', ' Aviamotornaia ',
       ' Marina Roshcha ', ' Proletarskaia ', ' Ploshchad Ilicha ',
       ' Okhotnyi riad ', ' Ulitca 1905 goda ', ' Sukharevskaia ',
       ' Taganskaia ', ' Botanicheskii sad ', ' Dmitrovskaia ',
       ' Ozernaia ', ' Baumanskaia ', ' Dinamo ', ' Polianka ',
       ' Sokolniki ', ' Lefortovo ', ' Akademicheskaia ', ' Pechatniki ',
       ' Minskaia ', ' Universitet ', ' Butyrskaia ', ' Ramenki ',
       ' Arbatskaia ', ' Bagrationovskaia ', ' Oktiabrskaia ',
       ' Iugo-Zapadnaia ', ' Oktiabrskoe pole ', ' Chertanovskaia ',
       ' Ziablikovo ', ' Novoperedelkino ', ' Kaluzhskaia ',
       ' Timiriazevskaia ', ' Kievskaia '], dtype=object)
```

In [159…

```
#Everthing seems fine, apart from a value named --No Data--, this will be deleted
data.drop(data.index[(data["metro"] == "No data")],axis=0,inplace=True)
```

In [160…

```
data['provider'].unique()
# These reveals some typo and also some russian spellings
```

Out[160…

```
array(['realtor                \xa0 \xa0 ', 'owner                 ',
       'realtor                 ',
       'agency                \xa0 \xa0\xa0 \xa0 ',
       'agency                 ', 'agency                \xa0 \xa0 ',
       'Застройщик                 '], dtype=object)
```

In [161…

```
data['provider']= data.provider.str.replace('realtor                \xa0 \xa0','relator')
data['provider']= data.provider.str.replace('realtor                 ','relator')
data['provider']= data.provider.str.replace('agency                \xa0 \xa0\xa0 \xa0 ','
data['provider']= data.provider.str.replace('agency                 ','agency')
data['provider']= data.provider.str.replace('Застройщик                 ','developer')
data['provider']= data.provider.str.replace('relator ','relator')
data['provider']= data.provider.str.replace('agency                \xa0 \xa0 ','agency')
data['provider']= data.provider.str.replace('owner                 ','owner')
data['provider'].unique()
```

Out[161…

```
array(['relator', 'owner', 'agency', 'developer'], dtype=object)
```

In [162…

```
# A visual inspection for typo in way column, everything looks fine
data['way'].unique()
```

Out[162…

```
array(['walk', 'transport'], dtype=object)
```

In [163…

```
# Separate features from target
price = data['price'].copy()
var_data = data.drop(columns=['price'])
var_data.head()
```

Out[163…

	metro	way	views	provider	fee_percent	storey	minutes	storeys	living_area	kitchen_a
0	Planernaia	walk	513	relator	50	7	10	12	19	
1	VDNKh	walk	389	relator	50	16	10	16	18	

Loading [MathJax]/extensions/Safe.js

	metro	way	views	provider	fee_percent	storey	minutes	storeys	living_area	kitchen_a
2	Alekseevskaia	walk	483	relator	50	5	3	12	19	
3	Sviblovo	walk	414	relator	50	3	15	5	37	
4	Rimskaia	walk	360	relator	99	6	7	17	21	

In [164…

```
# For feature engineering
#1. I adjusted the value for skew, that is using log to transform them into a normal dist
#2. I aslo converted non categorical variable to variables by using dummy variables

# Checking for skew
num_data = var_data.select_dtypes('number').columns
skew_limit = 0.75
skew_vals = var_data[num_data].skew()
skew_vals
```

Out[164…

```
views            4.060967
fee_percent      0.053198
storey          33.988690
minutes          1.161365
storeys         37.518854
living_area      1.749263
kitchen_area     2.592403
total_area      -0.644455
dtype: float64
```

In [165…

```
#Filter out skew columns
skew_cols = skew_vals[abs(skew_vals)>skew_limit].sort_values(ascending=True)
ss = skew_cols
```

In [166…

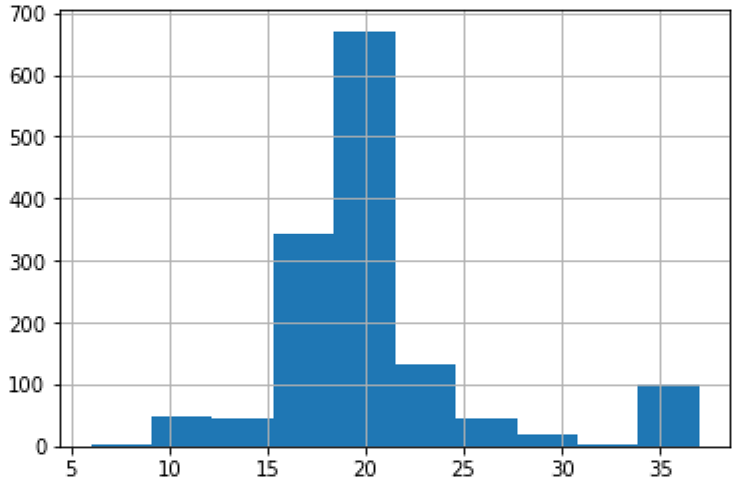
```
#Applying log transformation
tran_var_data = var_data
for col in ss.index.values:
    tran_var_data[col] = tran_var_data[col].apply(np.log1p)
```

In [169…

```
# Let us look at living_area before and after transformation
# Before transformation
data['living_area'].hist()
```

Out[169…

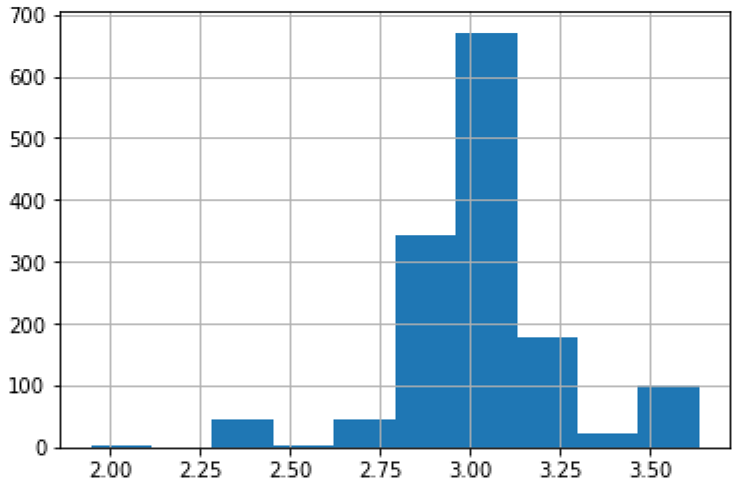
```
<AxesSubplot:>
```



Loading [MathJax]/extensions/Safe.js

In [170…
```
#After transformation
tran_var_data['living_area'].hist()
```

Out[170…
```
<AxesSubplot:>
```



In [ ]:
```

```

In [125…
```
# Creating dummy variables for the non numeric colum
metro_dummy = pd.get_dummies(data.metro)
way_dummy = pd.get_dummies(data.way)
provider_dummy= pd.get_dummies(data.provider)

#Merging dummy variables with transformed data

merged = pd.concat([tran_var_data,metro_dummy,way_dummy,provider_dummy], axis = 'columns'
```

In [134…
```
merged.head()
# A visual view of the data
```

Out[134…

	metro	way	views	provider	fee_percent	storey	minutes	storeys	living_area	kit
0	Planernaia	walk	6.242223	relator	50	2.079442	2.397895	2.564949	2.995732	
1	VDNKh	walk	5.966147	relator	50	2.833213	2.397895	2.833213	2.944439	
2	Alekseevskaia	walk	6.182085	relator	50	1.791759	1.386294	2.564949	2.995732	
3	Sviblovo	walk	6.028279	relator	50	1.386294	2.772589	1.791759	3.637586	
4	Rimskaia	walk	5.888878	relator	99	1.945910	2.079442	2.890372	3.091042	

5 rows × 132 columns

In [142…
```
#Dropping categorical data
aa = merged
final = aa.drop(['metro','way','provider'], axis = 'columns')
```

In [171…
```
# Formulating three hypothesis for this data
#1. The price of a house is normally distributed
#2. There is a correlation between the price of a house and the space of the living area
#3. There is a 50% chance that the provider is a relator

# Building test hypothesis 1
```

Loading [MathJax]/extensions/Safe.js

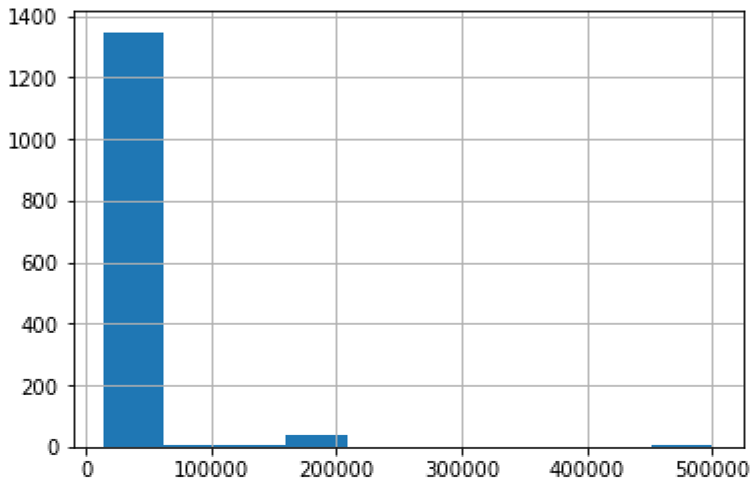
```
#First let me start with a plot
```

In [147…

```
price.hist()
```

Out[147…

```
<AxesSubplot:>
```



In [149…

```
# A plot shows that the distribution is not normal. For futher testing I will do a normal

from scipy.stats import normaltest
stat, p = normaltest(price)
print('stat = %.10f, p = %.10f' %(stat,p))

# The cut off is 0.05
if p > 0.05:
    print ('Normal')
else:
    print ('Not normally distrubted')
```

```
stat = 1712.4456333485, p = 0.0000000000
Not normally distrubted
```

In [ ]:

```
# Suggestion for next steps
#1. A ploynomial transformation could be checked and applied to the variables
#2. The data could be split, one for training and the other for testing

#Summary of the quality of the data
# The data is relatively small, it looks like some form of cleaning has been done. There
# Additional data could be provided by collecting more recent data and also getting data
```

Loading [MathJax]/extensions/Safe.js