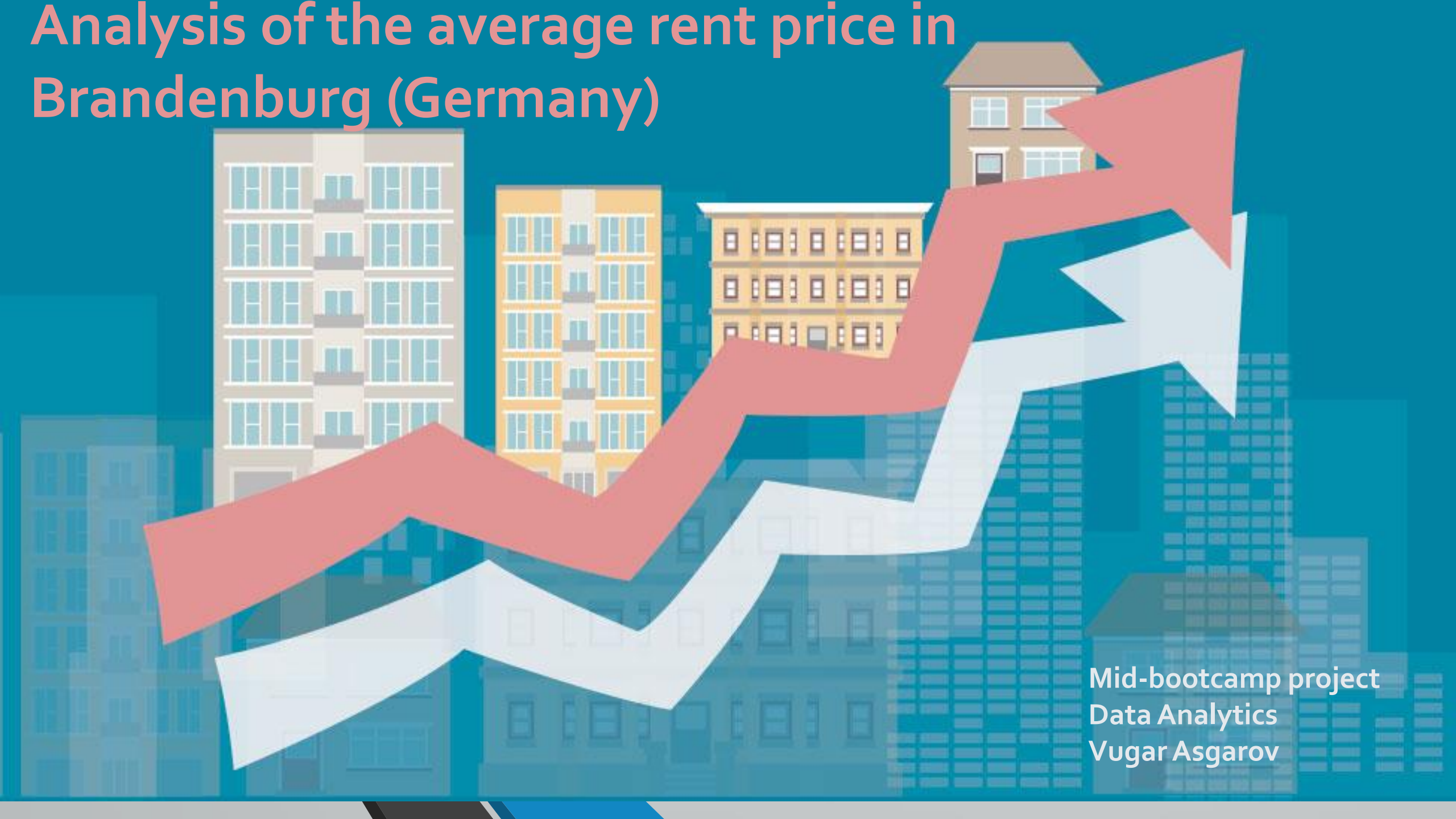
# Analysis of the average rent price in Brandenburg (Germany)



Mid-bootcamp project
Data Analytics
Vugar Asgarov

# Brainstorming and finding an appropiate Data set

Raw data set


Working with Data

# Research Focus:

- Rent Distribution in land Brandenburg (Germany)
- Which city has the most expensive and cheapest rent in Land Brandenburg?
- What are the factors (such as rooms, balcony, parking space) which can influence on rent price in the most expensive city of Brandenburg?
- Comparing KNN and Linear regression models in order to predict rent prices

# 1. Data Cleaning and Exploration

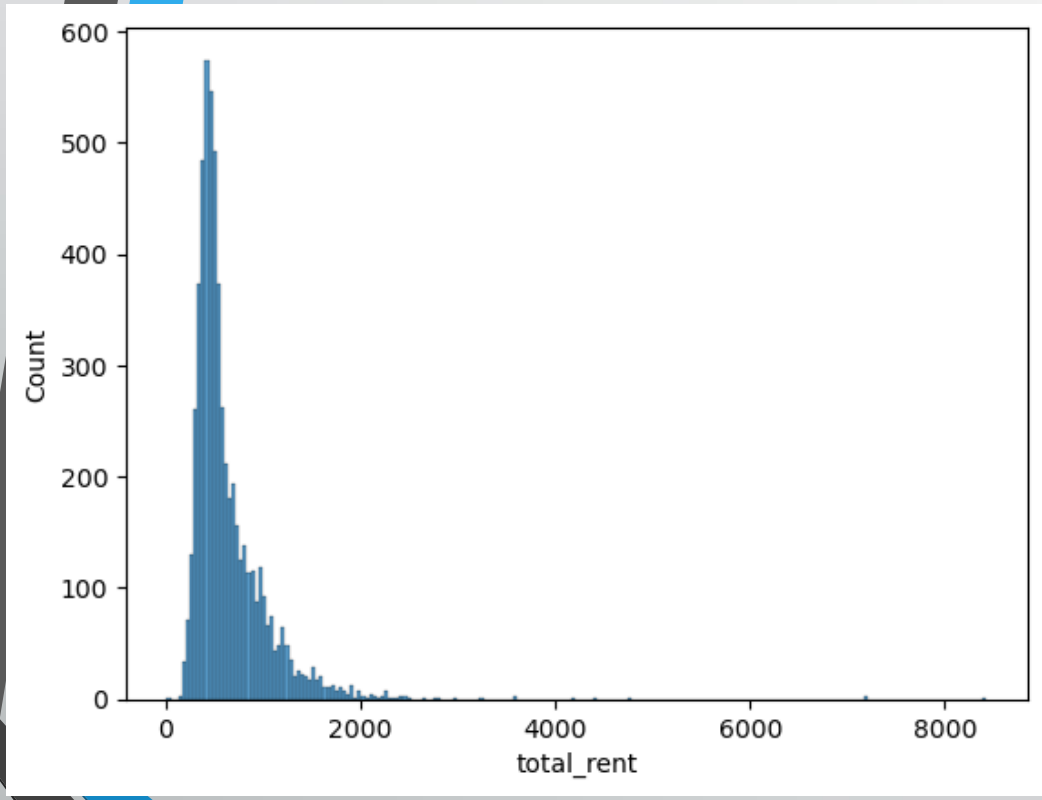
- Restricting data to special region(Brandenburg) as dataset is really big
- Renaming and Standartize the columns
- Removing all duplicates
- Checking missing values (dropping if needed and replacing them )

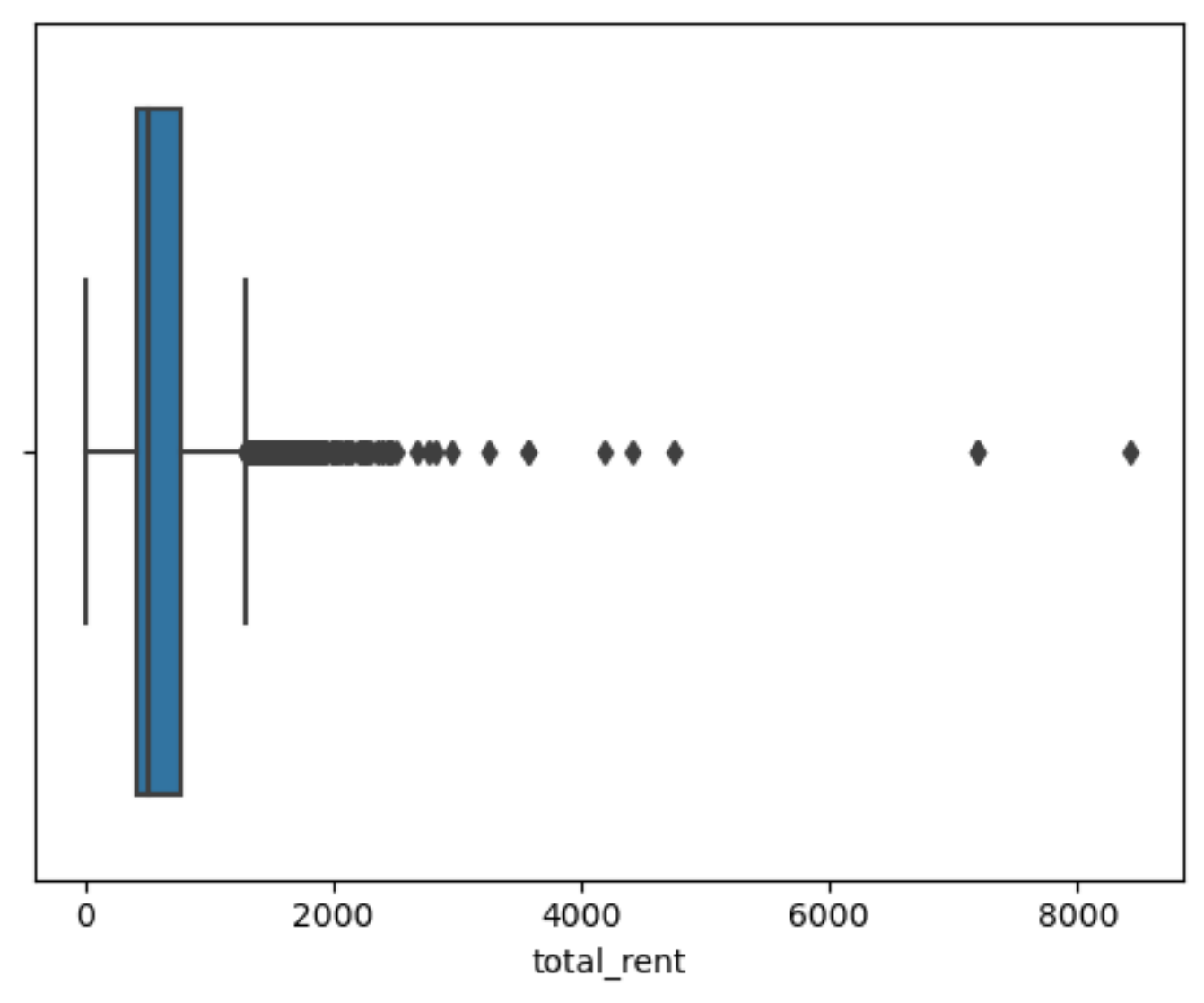
In [8]:
```
brandenburg_df = brandenburg_df.loc[:,['regio1','regio2',
'balcony', 'livingSpace', 'totalRent', 'serviceCharge',
'baseRent','lift','petsAllowed','floor','noRooms','garden','cellar','noParkSpaces']]
brandenburg_df.head()
```

Out[8]:

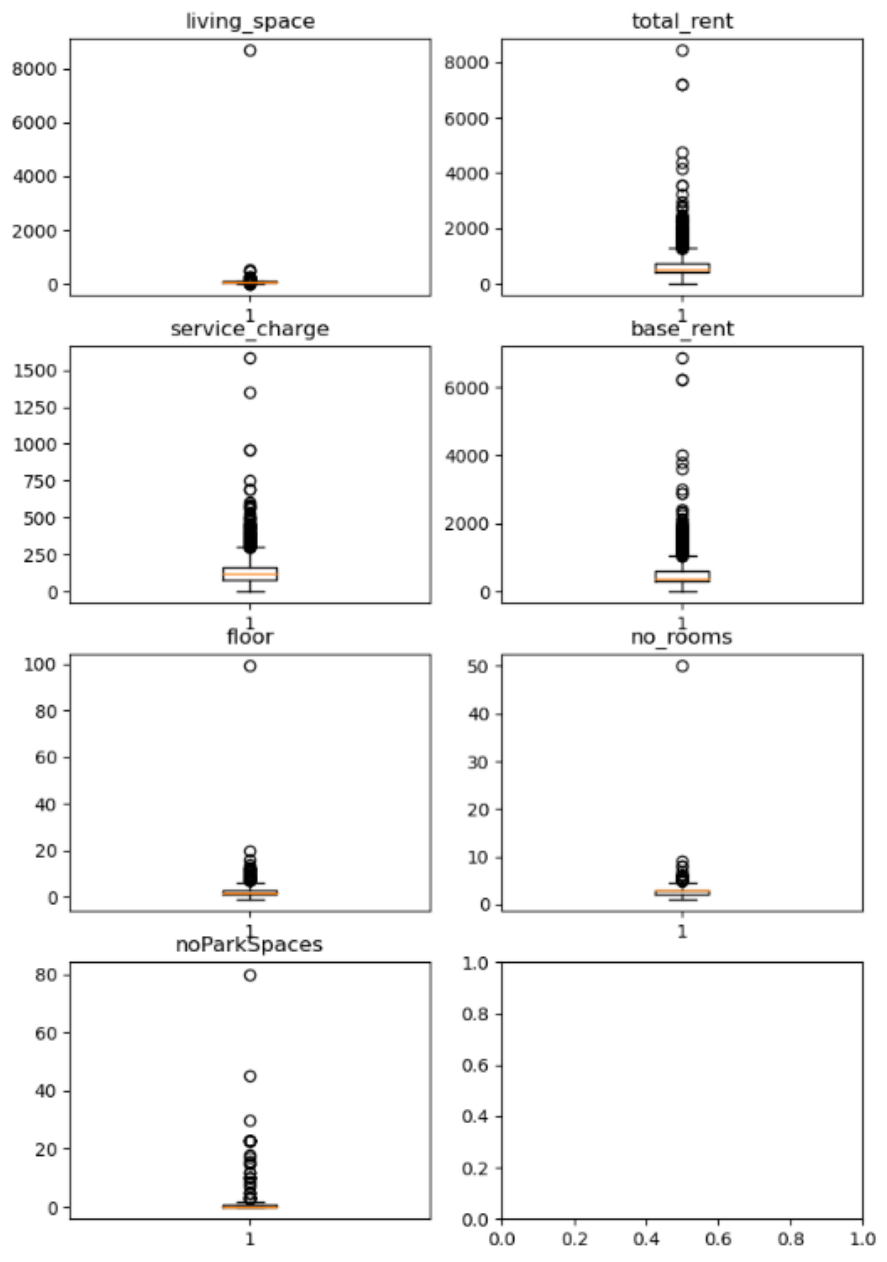
| | regio1 | regio2 | balcony | livingSpace | totalRent | serviceCharge | baseRent | lift | petsAllowed | floor | noRooms | garden | cellar | noP |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 50 | Brandenburg | Oder_Spree_Kreis | True | 36.29 | 377.0 | 50.0 | 285.0 | True | negotiable | 1.0 | 1.0 | False | True | |
| 114 | Brandenburg | Märkisch_Oderland_Kreis | True | 60.71 | 435.0 | 65.0 | 300.0 | False | NaN | 0.0 | 2.0 | False | True | |
| 136 | Brandenburg | Brandenburg_an_der_Havel | True | 67.80 | 590.0 | 170.0 | 420.0 | True | yes | 0.0 | 3.0 | False | True | |
| 232 | Brandenburg | Potsdam | True | 88.00 | 1300.0 | 300.0 | 1000.0 | True | negotiable | 2.0 | 3.0 | True | True | |
| 331 | Brandenburg | Dahme_Spreewald_Kreis | True | 69.00 | 533.0 | 150.0 | 370.0 | False | negotiable | 2.0 | 2.0 | True | True | |

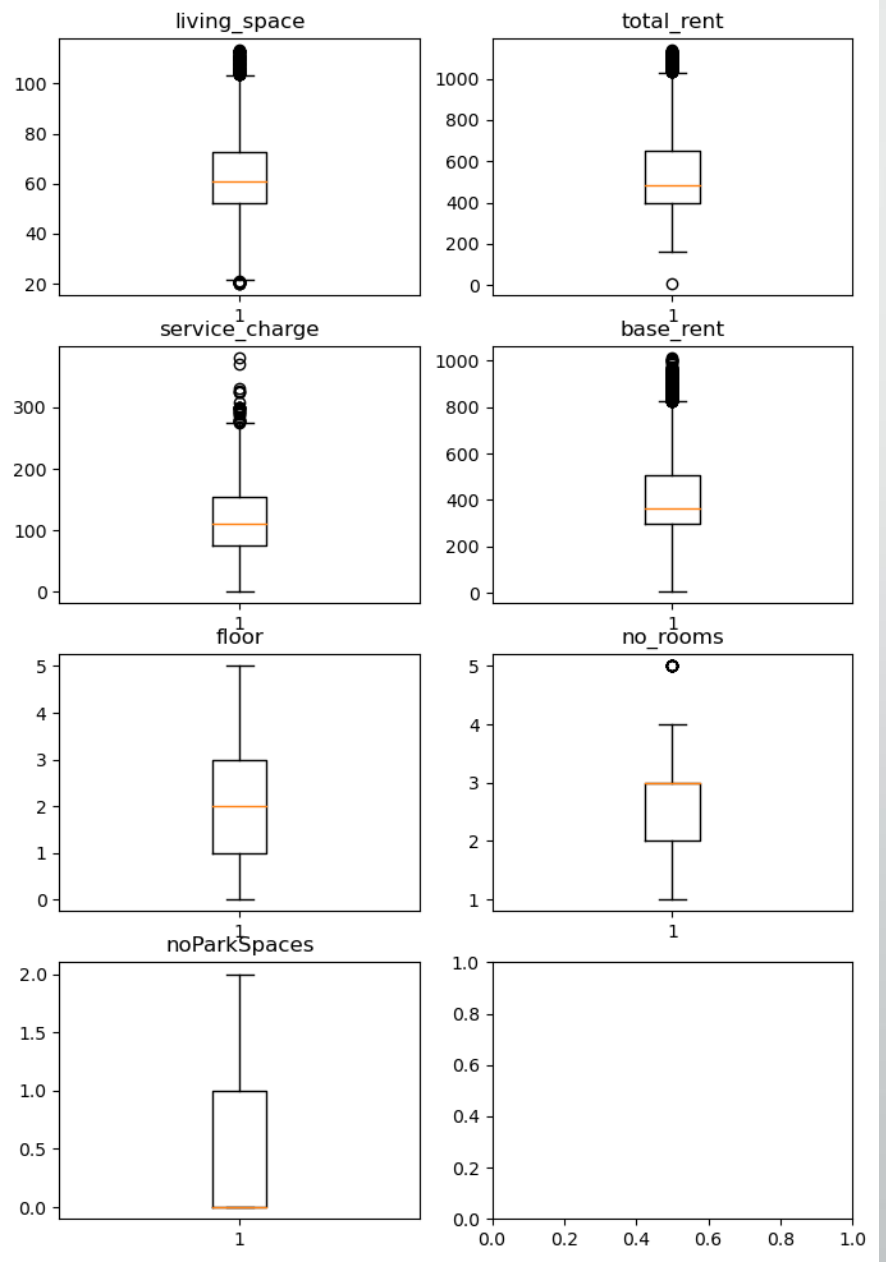
# Rent price distribution in Land Brandenburg

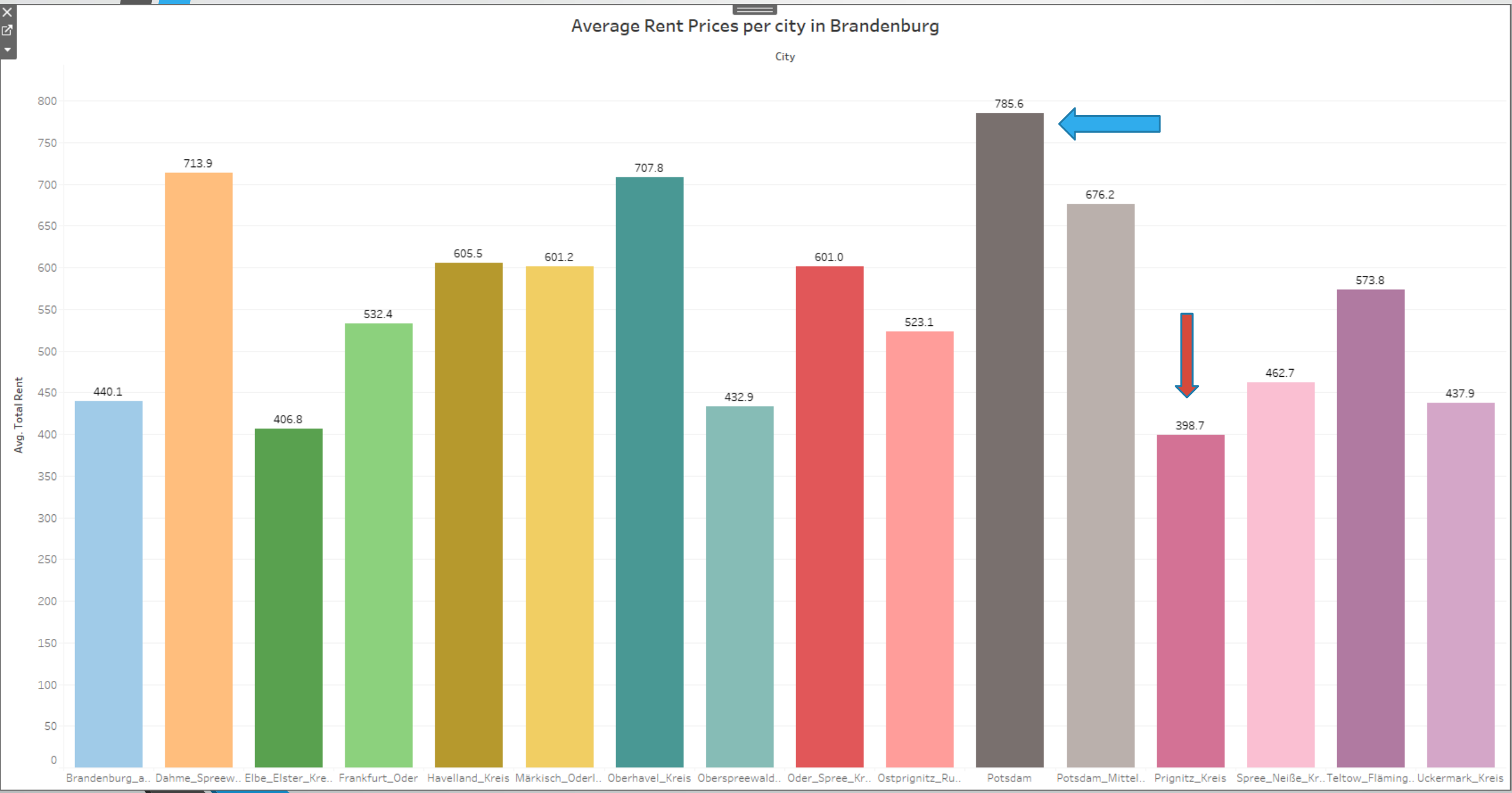


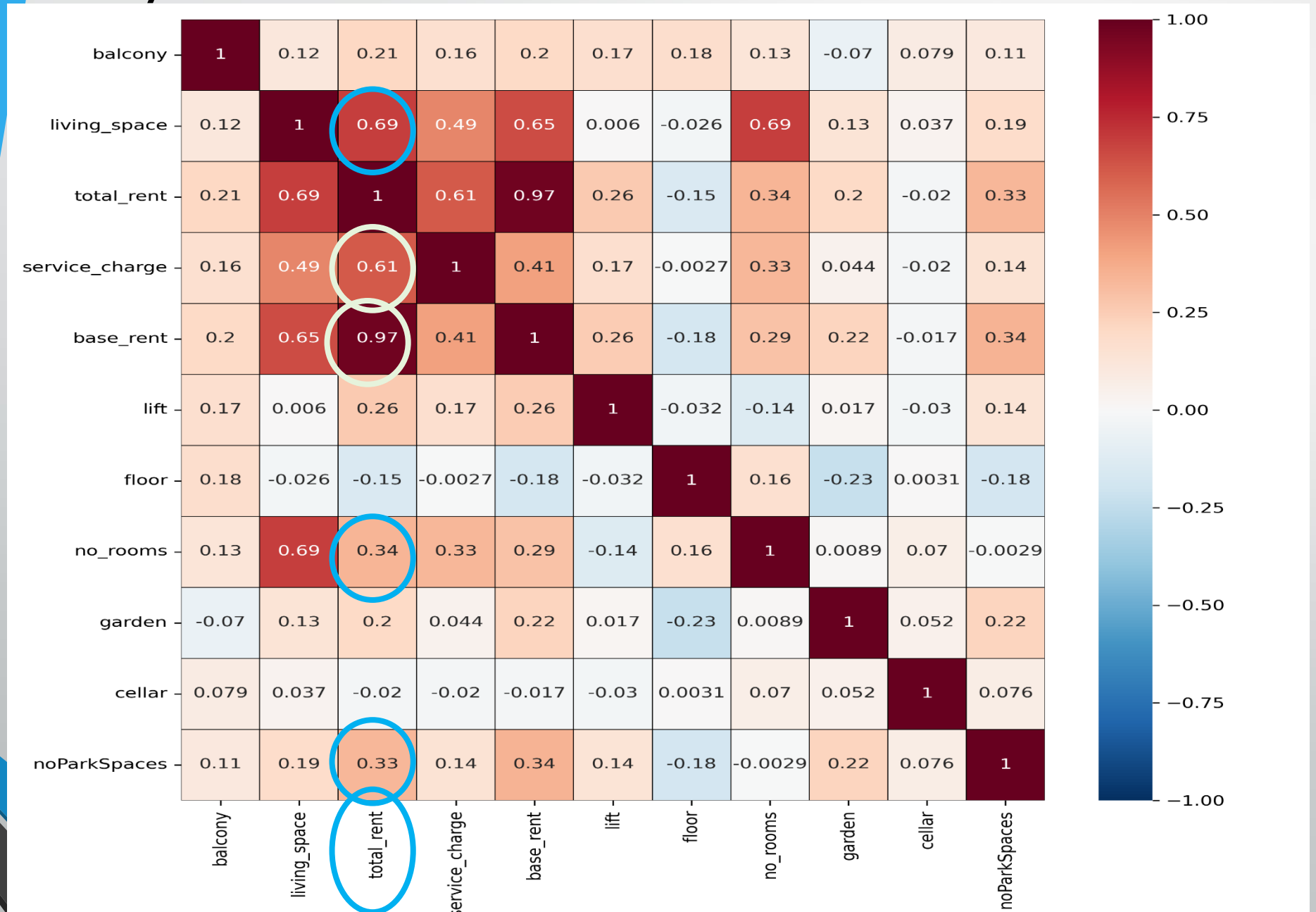
# 2. Dealing with outliers



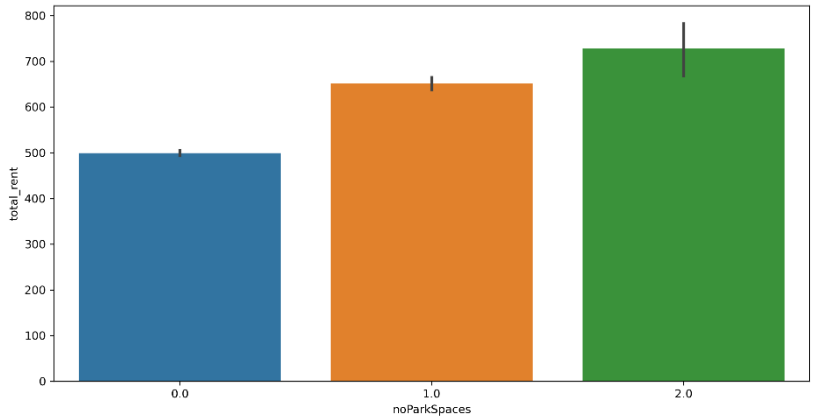



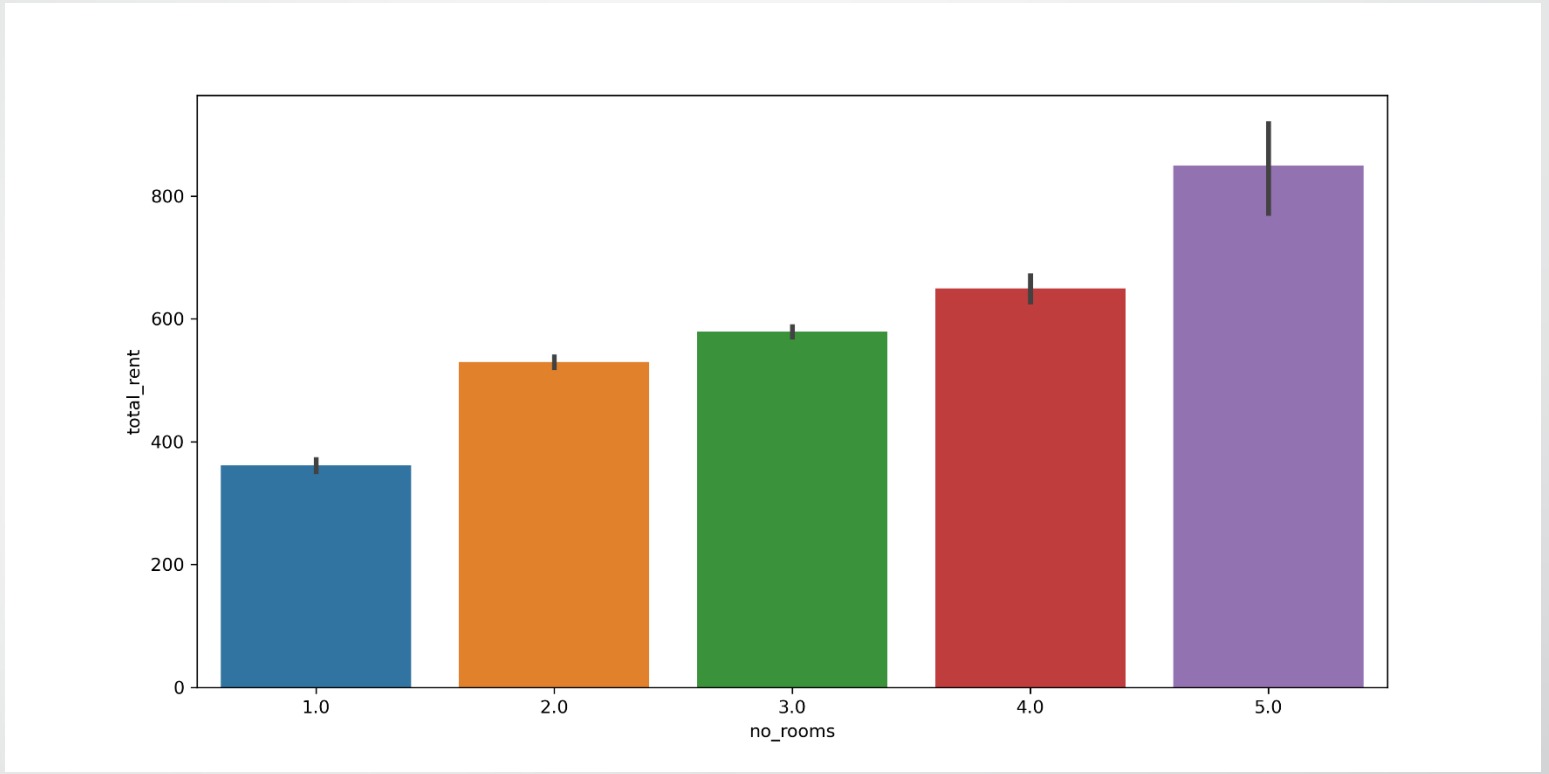

Average Rent Prices per city in Brandenburg
City
Avg. Total Rent
800
750
700
650
600
550
500
450
400
350
300
250
200
150
100
50
0
440.1
713.9
406.8
532.4
605.5
601.2
707.8
432.9
601.0
523.1
785.6
676.2
398.7
462.7
573.8
437.9
Brandenburg_a..
Dahme_Spreew..
Elbe_Elster_Kre..
Frankfurt_Oder
Havelland_Kreis
Märkisch_Oderl..
Oberhavel_Kreis
Oberspreewald..
Oder_Spree_Kr..
Ostprignitz_Ru..
Potsdam
Potsdam_Mittel..
Prignitz_Kreis
Spree_Neiße_Kr..
Teltow_Fläming..
Uckermark_Kreis

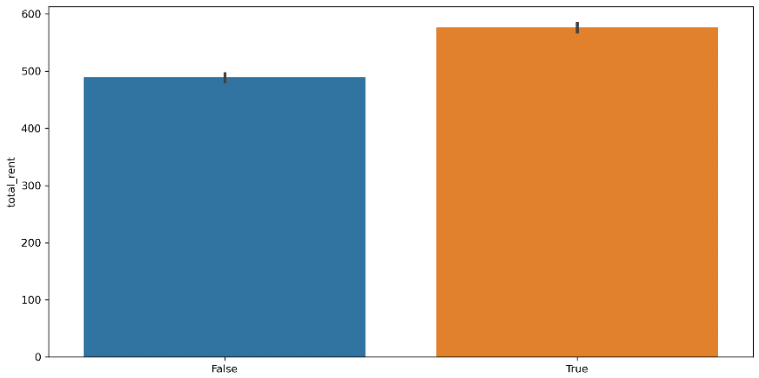
# 3. Analysis of Data (correlation)

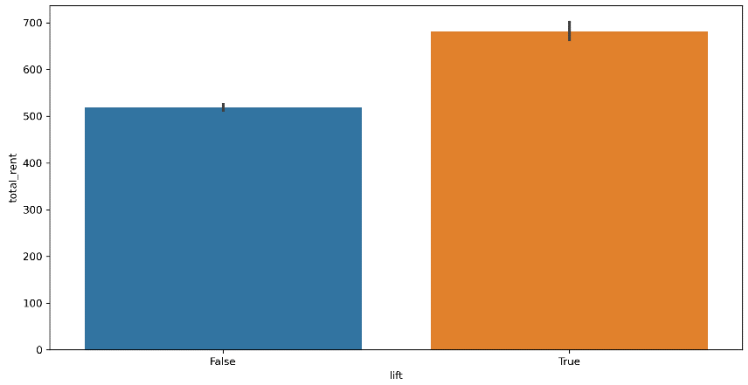


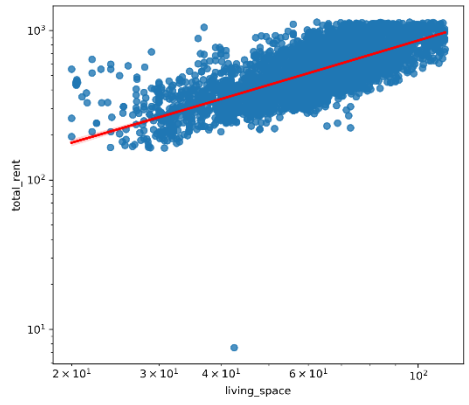
# 4. Main Factors which has an impact on total rent price



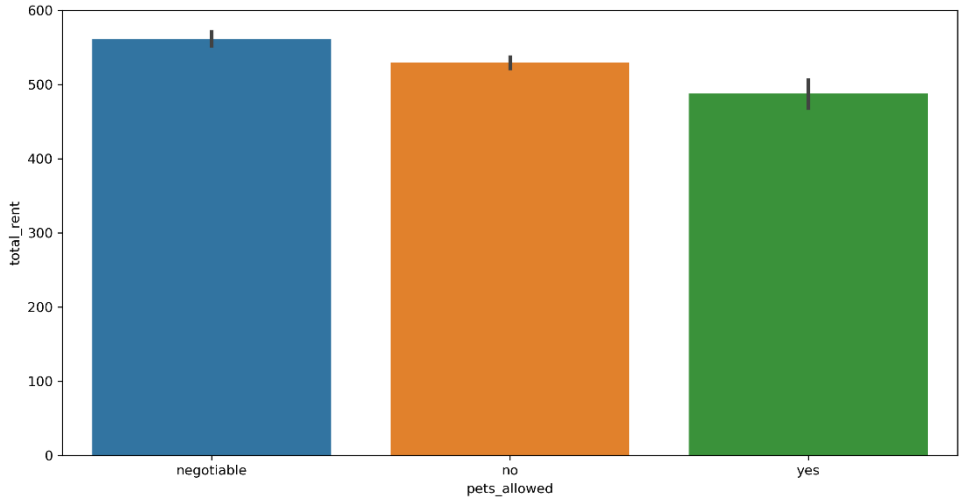



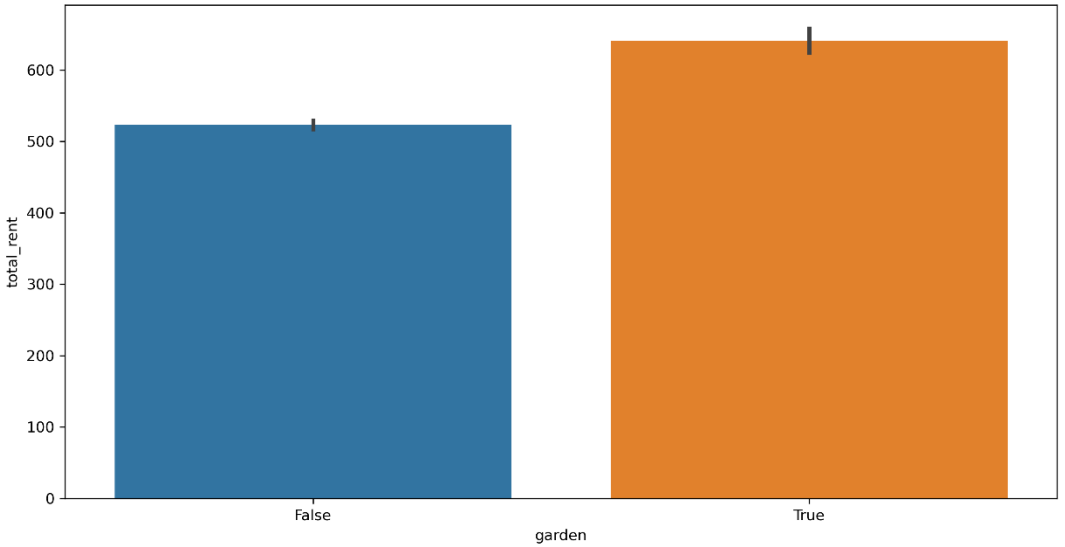



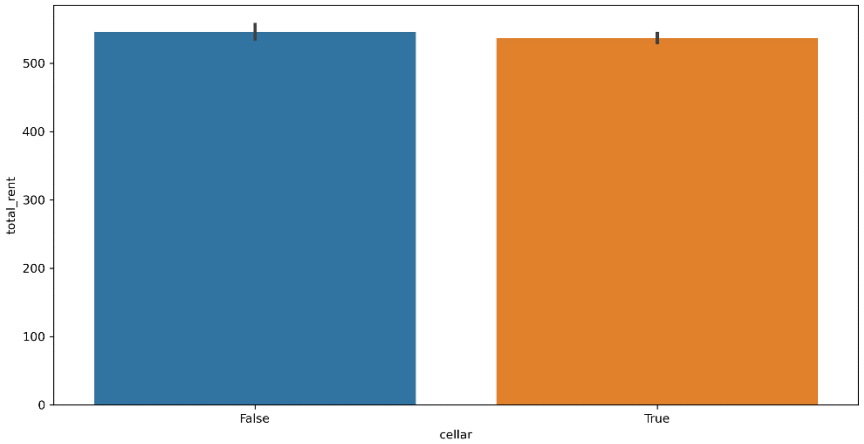



# Less impact on total rent price



# 5. Comparing Linier and KNN regression models in order to predict the total price

- Splitting the data
- Scaling and Transforming columns
- Appyling the KNN and Linier Regression Models

**Linear Regression**

In [14]:
```
#Apply linear regression
from sklearn.linear_model import LinearRegression as LinReg

linreg=LinReg()    # model
linreg.fit(X_train_f, y_train)   # model train
y_test_pred_linreg=linreg.predict(X_test_f)   # model prediction
y_train_pred_linreg=linreg.predict(X_train_f)   # model prediction
```

In [15]:
```
#stoing linear
#with open("../Models/Linear.pkl", "wb") as file:
 #   pickle.dump(linreg, file, pickle.HIGHEST_PROTOCOL)
```

**K-NN**

In [16]:
```
# help(KNeighborsRegressor)
```

In [17]:
```
from sklearn.neighbors import KNeighborsRegressor
regressor = KNeighborsRegressor(n_neighbors=15,weights='distance')
regressor.fit(X_train_f, y_train)
```

Out[17]:
```
KNeighborsRegressor(n_neighbors=15, weights='distance')
```

# Model Validation

In [23]:
```
print ('Linear: train R2: {} -- test R2: {}'.format(linreg.score(X_train_f, y_train),
                                                    linreg.score(X_test_f, y_test)))
print ('KNN: train R2: {} -- test R2: {}'.format(regressor.score(X_train_f, y_train),
                                                 regressor.score(X_test_f, y_test)))
```

```
Linear: train R2: 0.702711868203153 -- test R2: 0.6711008808731851
KNN: train R2: 0.9826771109528131 -- test R2: 0.6718832275006406
```

In [25]:
```
from sklearn.metrics import mean_absolute_error as mae

train_mae=mae(linreg.predict(X_train_f), y_train)
test_mae=mae(linreg.predict(X_test_f), y_test)

print ('Linear: train MAE: {} -- test MAE: {}'.format(train_mae, test_mae))

train_mae=mae(regressor.predict(X_train_f), y_train)
test_mae=mae(regressor.predict(X_test_f), y_test)

print ('KNN: train MAE: {} -- test MAE: {}'.format(train_mae, test_mae))
```

```
Linear: train MAE: 81.57288338180335 -- test MAE: 86.01133954172518
KNN: train MAE: 8.512774966975657 -- test MAE: 80.77195489568828
```

**Result:When we look at "R2"score both model has almost same performance on the test set. However Linier has a bit high "mae" performance.**

# Thank you for your Attention