United States House Rent Prediction

Team 12

Prashanti Salunkhe
Pratik Randad

Background

The main goal of the research is to create a prediction model using Machine Learning to predict the rental costs of houses across the United States based on numerous variables defining the houses' attributes. This dataset contains several features that characterize the entire nature of the house and its dynamics, as well as ‘Price’ which is to be predicted.

Motivation

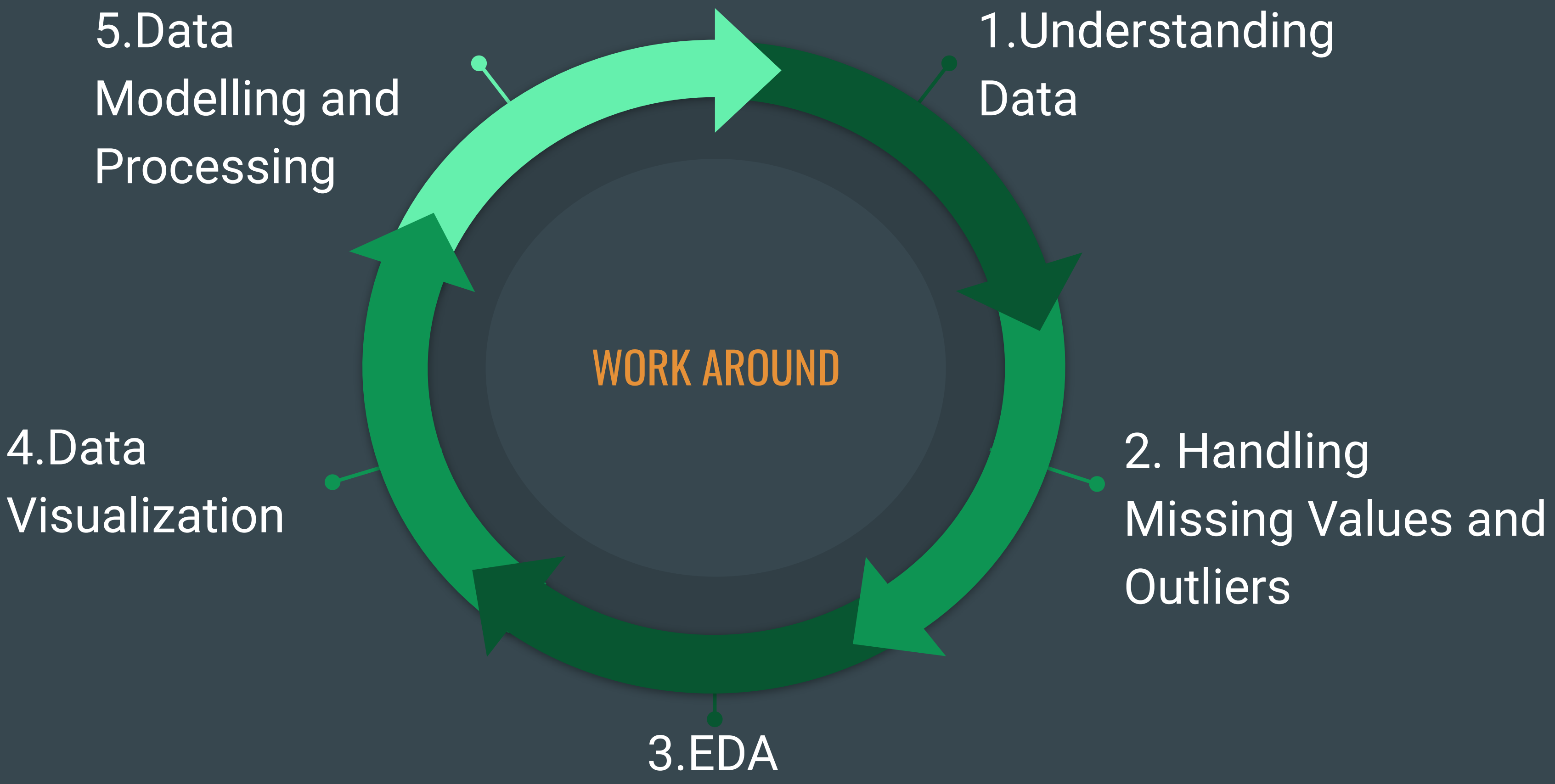
The house rent is an important deciding factor for us to manage the finances. With the growing number of real estates, a rent prediction study only serves to assist investors in determining the earning potential of a specific property in each region with specific qualities, thereby boosting the efficiency of real estate investment in the market. The goal of this project is to assist both landlords and tenants in pricing their rental properties appropriately.

Goals

A Model to predict Apartment Rent/Price

Reduce the disparity between the real rent and the rent anticipated

Evaluate performance of multiple models and select the best


WORK AROUND
1.Understanding Data
2. Handling Missing Values and Outliers
3.EDA
4.Data Visualization
5.Data Modelling and Processing

Specifications of Dataset

This dataset has been taken from Kaggle:

https://www.kaggle.com/datasets/rkb0023/houserentpredictiondataset

Description:

- Dataset size: 380.3MB
- Number of rows: 265,190
- Number of columns: 22

Column Description

- id
- url
- region
- region_url
- price
- type
- sqfeet
- beds
- baths
- cats_allowed
- dogs_allowed
- smoking_allowed
- wheelchair_access
- electric_vehicle_charge
- comes_furnished
- laundary_options
- parking_options
- image_url
- description
- lat
- long
- state

Data Info

```
In [6]: src_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 265190 entries, 0 to 265189
Data columns (total 22 columns):
 #   Column                   Non-Null Count   Dtype  
---  ------                   --------------   -----  
 0   id                       265190 non-null  int64  
 1   url                      265190 non-null  object 
 2   region                   265190 non-null  object 
 3   region_url               265190 non-null  object 
 4   price                    265190 non-null  int64  
 5   type                     265190 non-null  object 
 6   sqfeet                   265190 non-null  int64  
 7   beds                     265190 non-null  int64  
 8   baths                    265190 non-null  float64
 9   cats_allowed             265190 non-null  int64  
 10  dogs_allowed             265190 non-null  int64  
 11  smoking_allowed          265190 non-null  int64  
 12  wheelchair_access        265190 non-null  int64  
 13  electric_vehicle_charge  265190 non-null  int64  
 14  comes_furnished          265190 non-null  int64  
 15  laundry_options          210879 non-null  object 
 16  parking_options          170055 non-null  object 
 17  image_url                265190 non-null  object 
 18  description              265188 non-null  object 
 19  lat                      263771 non-null  float64
 20  long                     263771 non-null  float64
 21  state                    265189 non-null  object 
dtypes: float64(3), int64(10), object(9)
memory usage: 44.5+ MB
```

Data Cleaning

1.Dropping Unwanted Columns

We can remove the columns id, url, region url, image url, and description from our dataset because we know they aren't needed for our current situation.

```
]: data = src_data.drop(columns = ['id', 'url', 'region_url', 'image_url', 'description'])
```

```
]: data.head()
```

]:

	region	price	type	sqfeet	beds	baths	cats_allowed	dogs_allowed	smoking_allowed	wheelchair_access	electric_vehicle_charge	comes_furnished	laundry_options	parking_options	lat	long	state
0	birmingham	1195	apartment	1908	3	2.0	1	1	1	0	0	0	laundry on site	street parking	33.4226	-86.7065	al
1	birmingham	1120	apartment	1319	3	2.0	1	1	1	0	0	0	laundry on site	off-street parking	33.3755	-86.8045	al
2	birmingham	825	apartment	1133	1	1.5	1	1	1	0	0	0	laundry on site	street parking	33.4226	-86.7065	al
3	birmingham	800	apartment	927	1	1.0	1	1	1	0	0	0	laundry on site	street parking	33.4226	-86.7065	al
4	birmingham	785	apartment	1047	2	1.0	1	1	1	0	0	0	laundry on site	street parking	33.4226	-86.7065	al

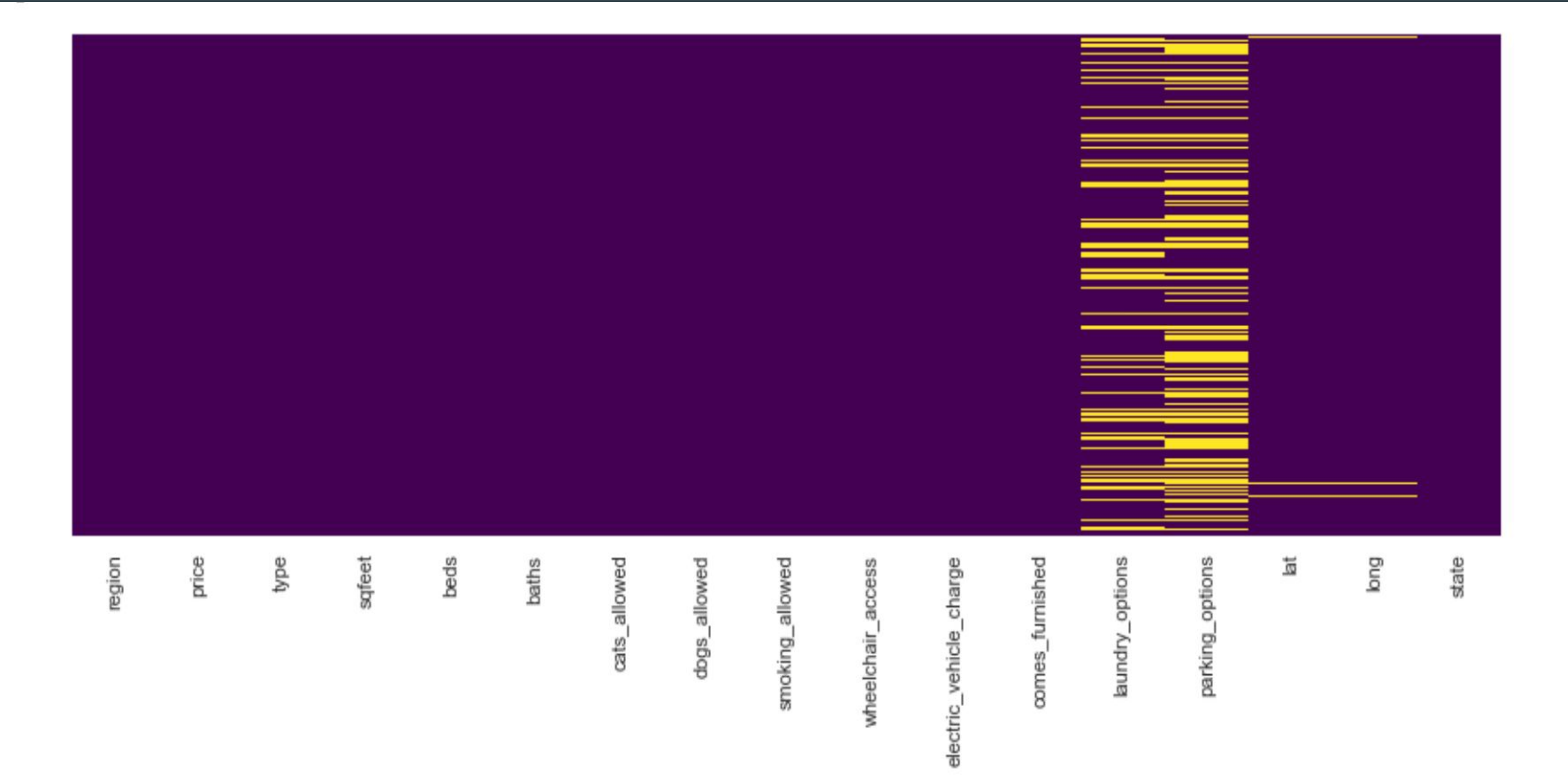
Analyzing Missing Values

```
In [12]: data.isna().sum()

Out[12]: region                         0
         price                          0
         type                           0
         sqfeet                         0
         beds                           0
         baths                          0
         cats_allowed                   0
         dogs_allowed                   0
         smoking_allowed                0
         wheelchair_access              0
         electric_vehicle_charge        0
         comes_furnished                0
         laundry_options            54311
         parking_options            95135
         lat                         1419
         long                        1419
         state                          1
         dtype: int64
```

We see that the 'lat' and 'long' columns have less null values, so we eliminate them.

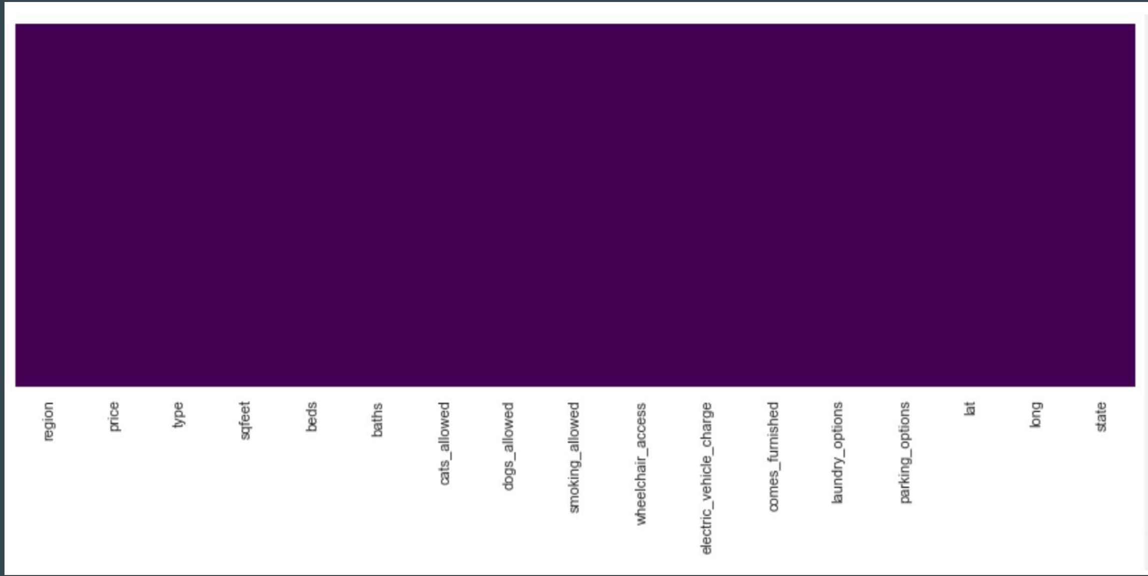
Visualizing Missing Values



Filling Missing Values

Used the most frequent values in each column to fill in the missing values in each columns, thereby eliminating null values

```
data=data.fillna(data.mode().iloc[0])
```



Identifying and Handling Outliers

```
data['type'].value_counts()
```

```
apartment          217090
house               23400
townhouse           10295
condo                4841
duplex               3436
manufactured         3004
cottage/cabin         697
loft                  510
flat                  349
in-law                144
land                    4
assisted living         1
Name: type, dtype: int64
```

```
In [26]: data['baths'].value_counts()

Out[26]: 1.0    135652
         2.0     93160
         1.5     18377
         2.5      7997
         3.0      3944
         0.0      2024
         4.0       988
         3.5       475
         4.5        77
         5.0        65
         5.5        21
         6.0         7
         8.0         1
         Name: baths, dtype: int64
```

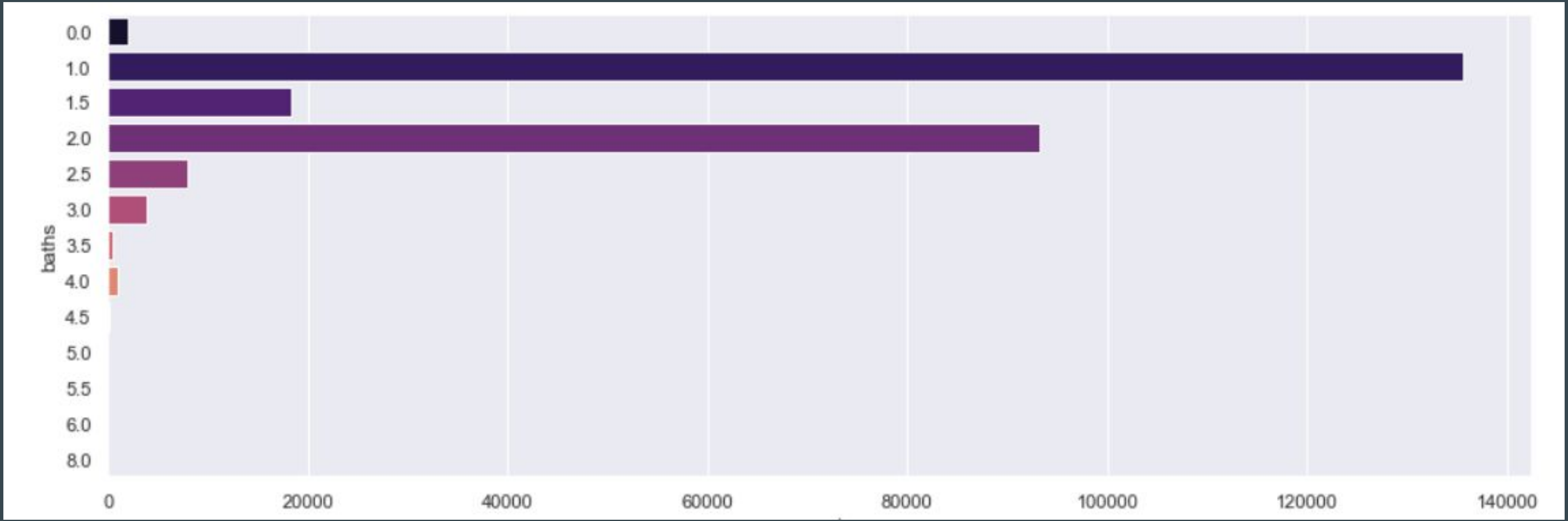
Column: 'Type'

Column:'Baths'

Identifying and Handling Outliers

Column: 'Type'

Identifying and Handling Outliers



Column: ‘Baths’

Identifying and Handling Outliers

```
In [19]: data=data[data['type']!='land']
         data=data[data['type']!='assisted living']
```

Column: ‘Type’

```
n [28]: data=data.loc[(data['baths']>0) & (data['baths']<7)].reset_index(drop=True)
```

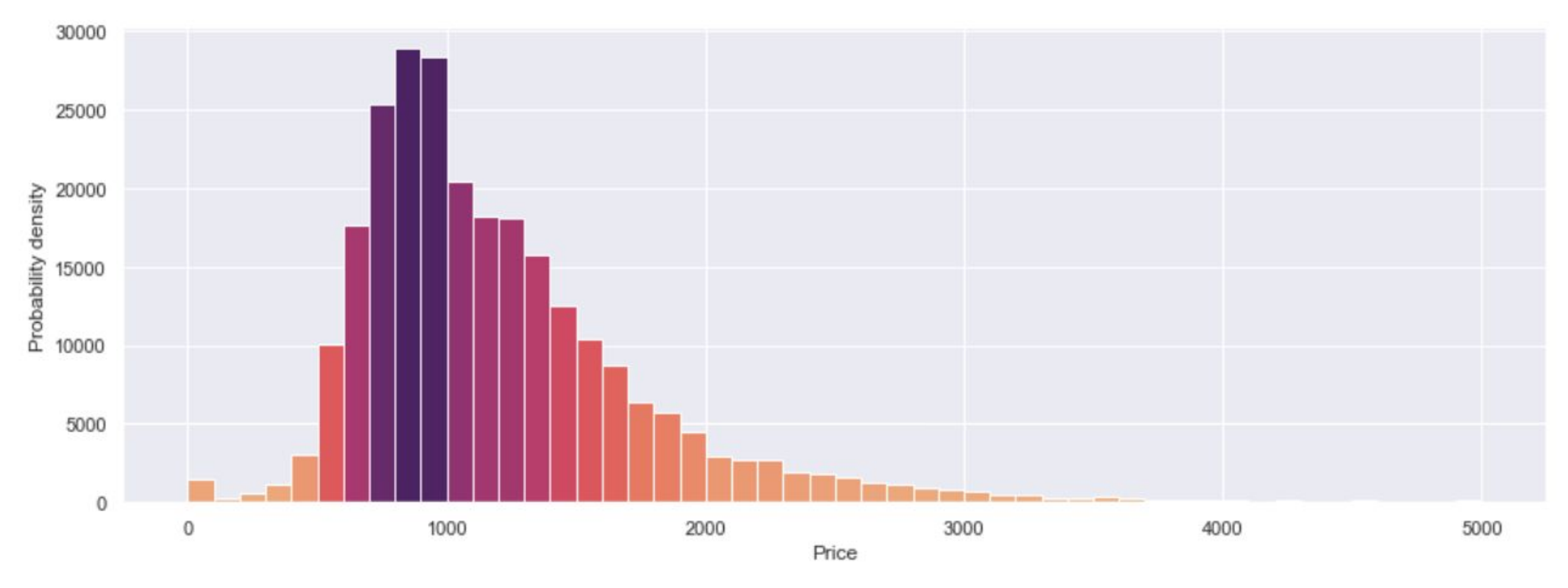
Column: ‘Baths’

Identifying and Handling Outliers

```
sns.boxplot( y=data["price"],palette=sns.color_palette("magma",10));
plt.show()
```

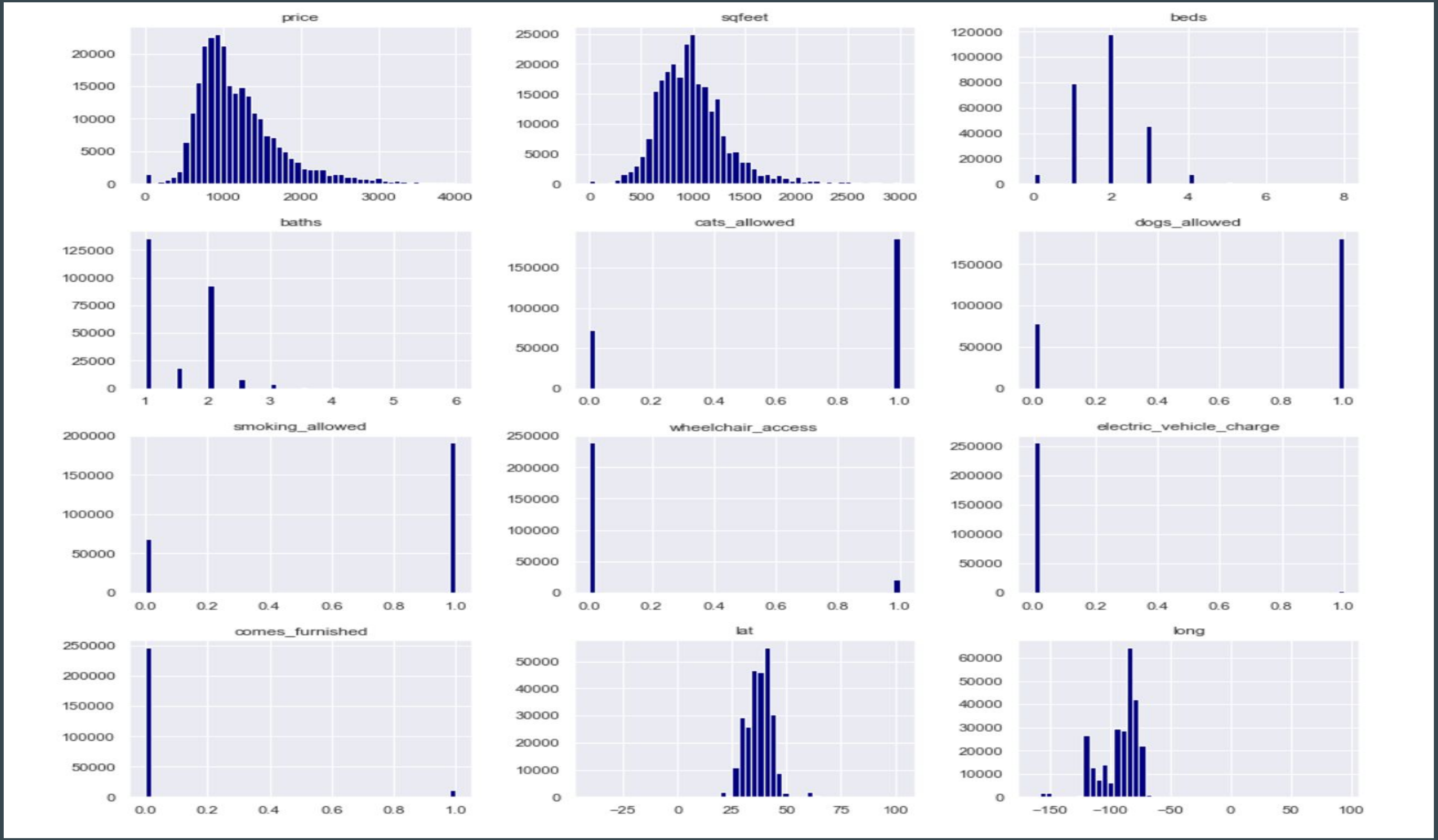
Column: ‘Price’

Identifying and Handling Outliers



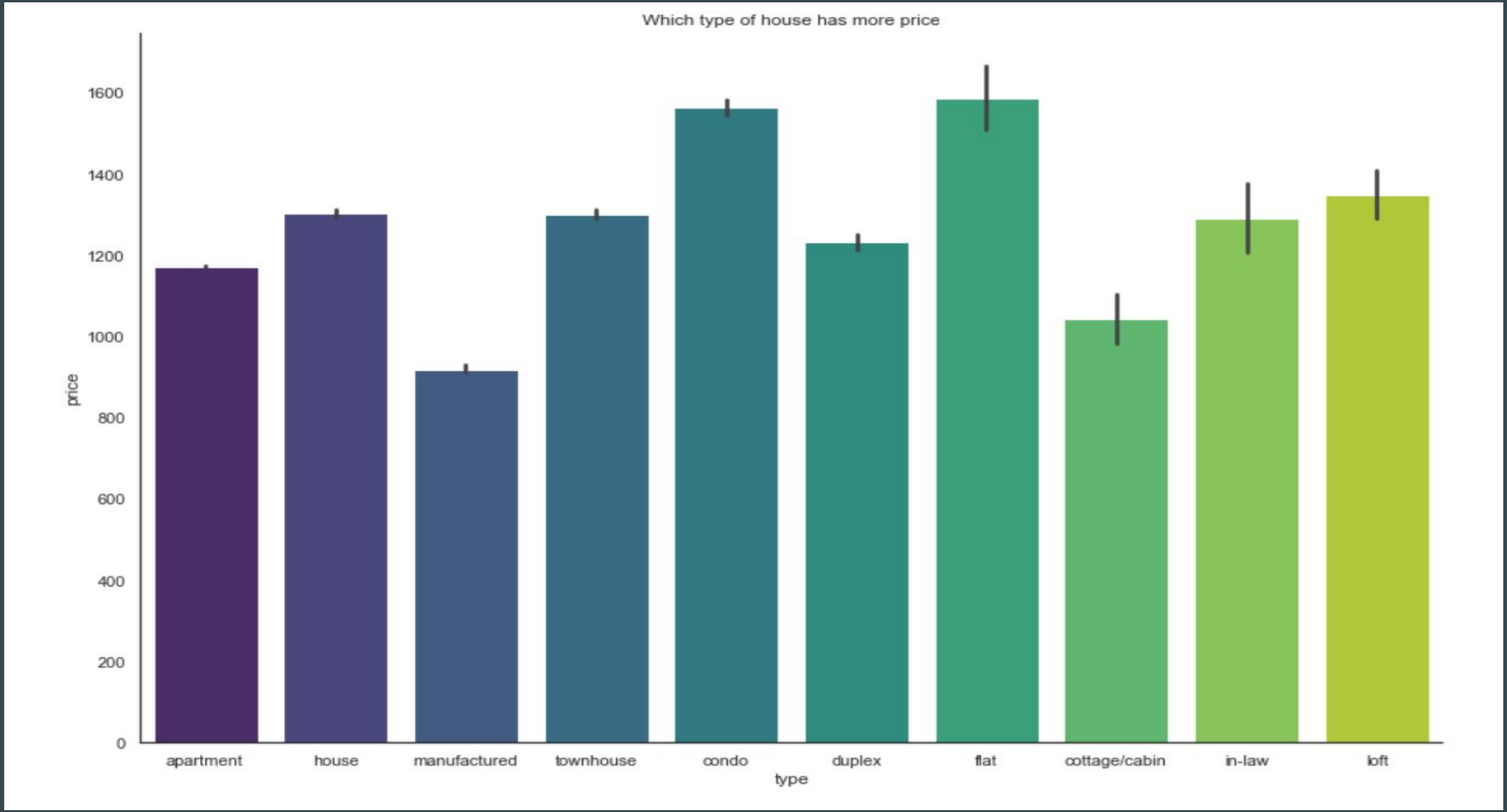
Column: ‘Price’

Data Visualization : Target vs Attribute

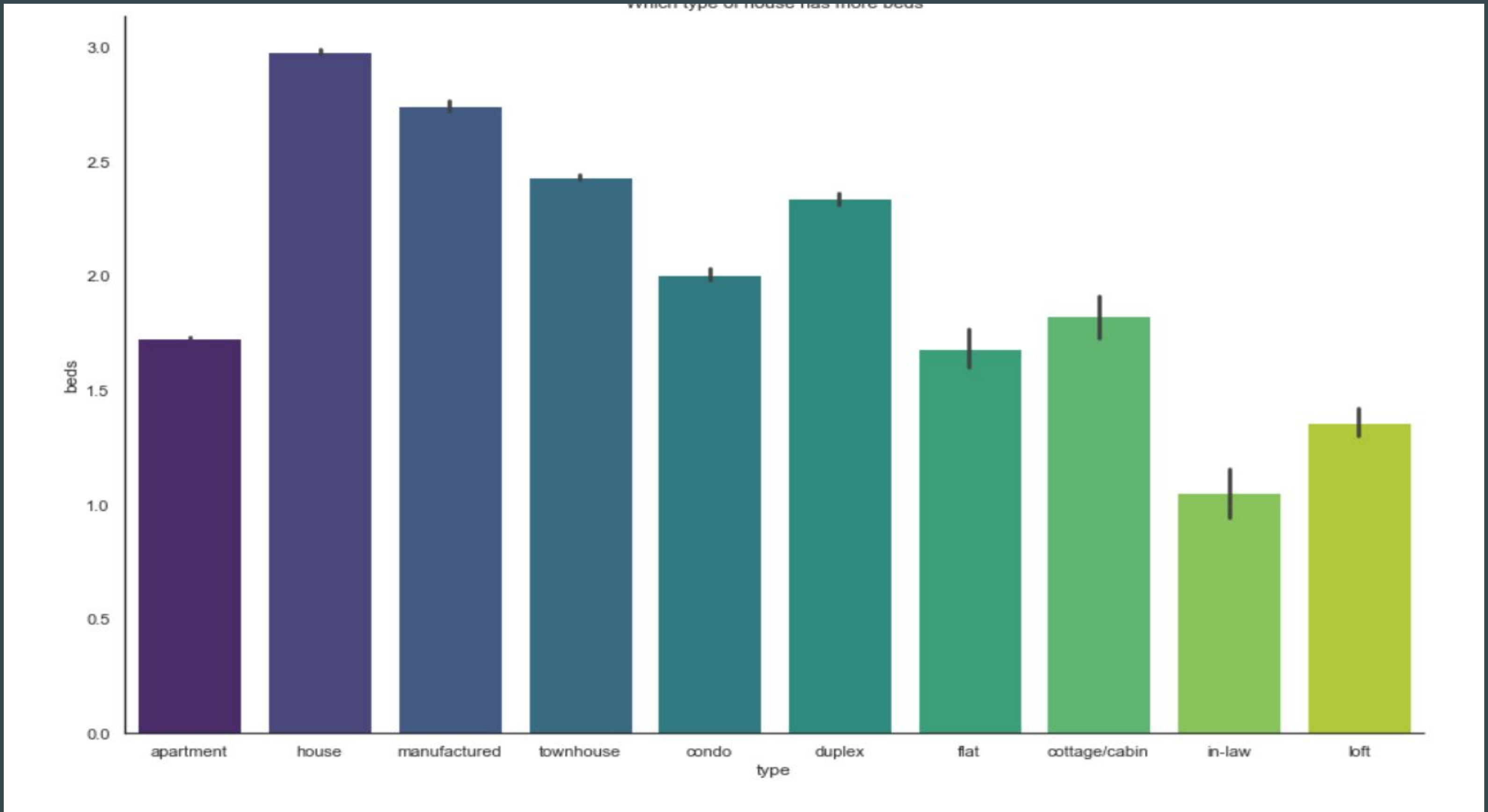


Data Visualization : Correlation between attributes

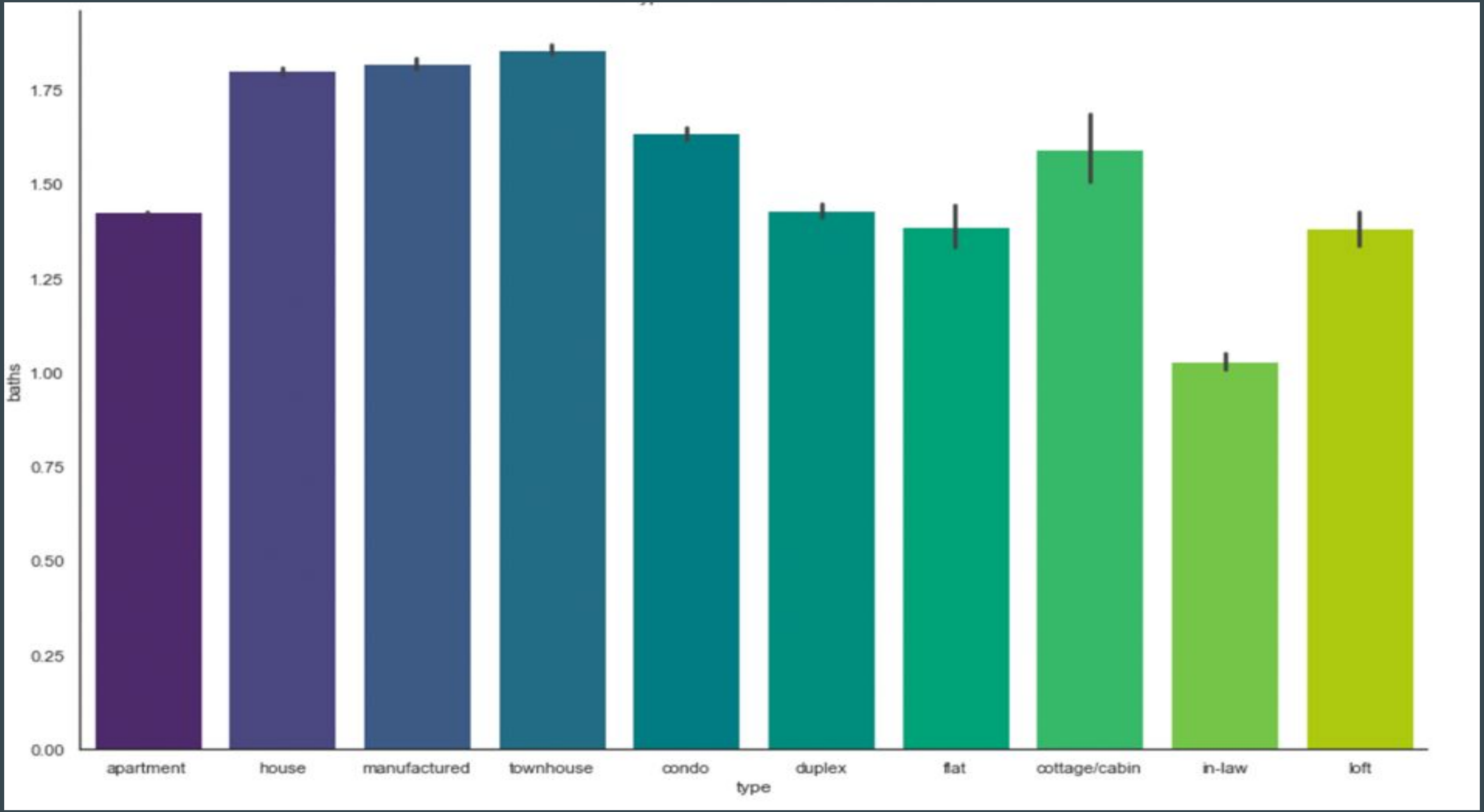
Data Visualization : Which type of house has more price



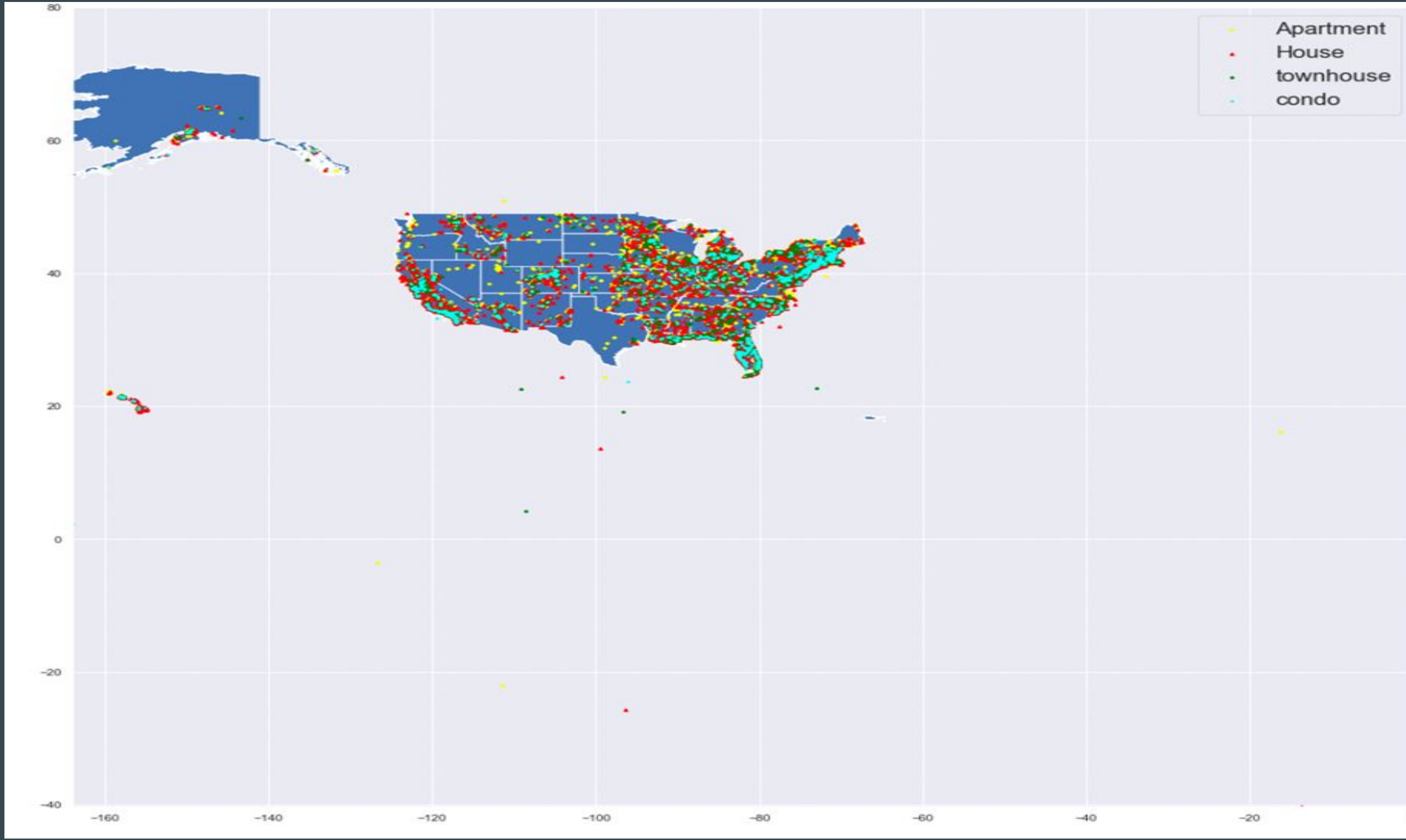
Data Visualization : Which type of house has more beds



Data Visualization : Which type of house has more baths



Data Visualization : With Geo Pandas



Data Processing : Label Encoding

In [64]:
```python
from sklearn.preprocessing import LabelEncoder
label = LabelEncoder()
```

In [65]:
```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 259319 entries, 0 to 259318
Data columns (total 18 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
 0   region                   259319 non-null  object
 1   price                    259319 non-null  int64
 2   type                     259319 non-null  object
 3   sqfeet                   259319 non-null  int64
 4   beds                     259319 non-null  int64
 5   baths                    259319 non-null  float64
 6   cats_allowed             259319 non-null  int64
 7   dogs_allowed             259319 non-null  int64
 8   smoking_allowed          259319 non-null  int64
 9   wheelchair_access        259319 non-null  int64
 10  electric_vehicle_charge  259319 non-null  int64
 11  comes_furnished          259319 non-null  int64
 12  laundry_options          259319 non-null  object
 13  parking_options          259319 non-null  object
 14  lat                      259319 non-null  float64
 15  long                     259319 non-null  float64
 16  state                    259319 non-null  object
 17  geometry                 259319 non-null  geometry
dtypes: float64(3), geometry(1), int64(9), object(5)
memory usage: 35.6+ MB
```

In [66]:
```python
data["region"]=label.fit_transform(data["region"])
data["type"]=label.fit_transform(data["type"])
data["laundry_options"]=label.fit_transform(data["laundry_options"])
data["parking_options"]=label.fit_transform(data["parking_options"])
data["state"]=label.fit_transform(data["state"])
```

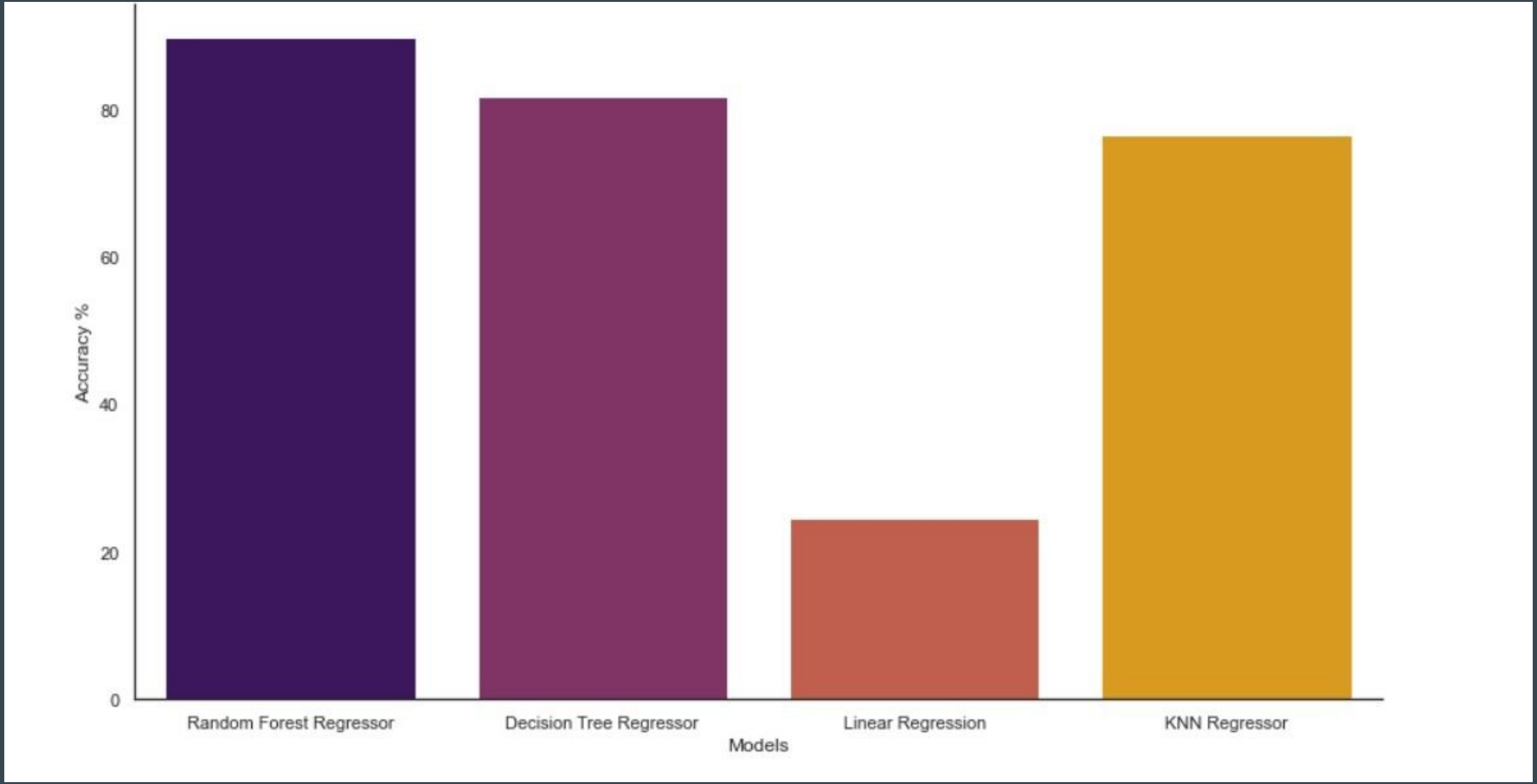
Models Used

- Random Forest Regressor
- Decision Tree Regressor
- Linear Regression
- KNN Regressor

Predicted Scores

Linear Regressor	Decision Tree Regressor	Random Forest Regressor	KNN Regressor
24.56%	81.67%	89.61%	76.55%

Model Accuracies



Conclusion and Future Scope

- Future Scope: To improve this models efficiency ahead, we can use ‘Description’ column to extract keywords that would give a bit more understanding about the attributes of the property, and add these as a column to use in the model.
- We've completed all the essential data processing and calculations, and Random Forest has the greatest accuracy score for Rent Prediction of all the algorithms employed.

- As Decision tree algorithm works for both the classification and regression problems, here also we got good accuracy for decision tree regression algorithm.
- If we do further feature selection that may increase the efficiency even more.We tried multiple values for neighbors in KNN regression model and we got the highest accuracy for neighbors=2.
- Random forest uses ensemble learning techniques for regression and classification tasks. Since it uses average value predicted from multiple random trees it gives the highest accuracy.

References

- https://medium.com/analytics-vidhya/fastest-way-to-install-geopandas-in-jupyter-notebook-on-windows-8f734e11fa2b
- https://www.statista.com/statistics/456925/median-size-of-single-family-home-usa/
- https://hersanyagci.medium.com/detecting-and-handling-outliers-with-pandas-7adbfcd5cad8
- https://catalog.data.gov/dataset/tiger-line-shapefile-2017-nation-u-s-current-state-and-equivalent-national
- https://seaborn.pydata.org/tutorial/color_palettes.html
- https://www.analyticsvidhya.com/blog/2018/08/k-nearest-neighbor-introduction-regression-python/
- https://stackoverflow.com/questions/49780491/plotting-histogram-for-all-columns-in-a-data-frame

Thank you!