# RentWise

## Predicting Rental Property Prices in US

Made by: Kartik

# Index

- Problem Statement
- My Solution
- Dataset overview
- Data Cleaning steps
- EDA
- Data Preprocessing
- Baseline Modelling
- Future steps

# Problem Statement

In the face of a booming rental market and soaring prices over the past decade, the increasing preference for renting over buying has resulted in a surge in rental costs, leaving many, including myself, grappling with the challenge of finding affordable rental properties.

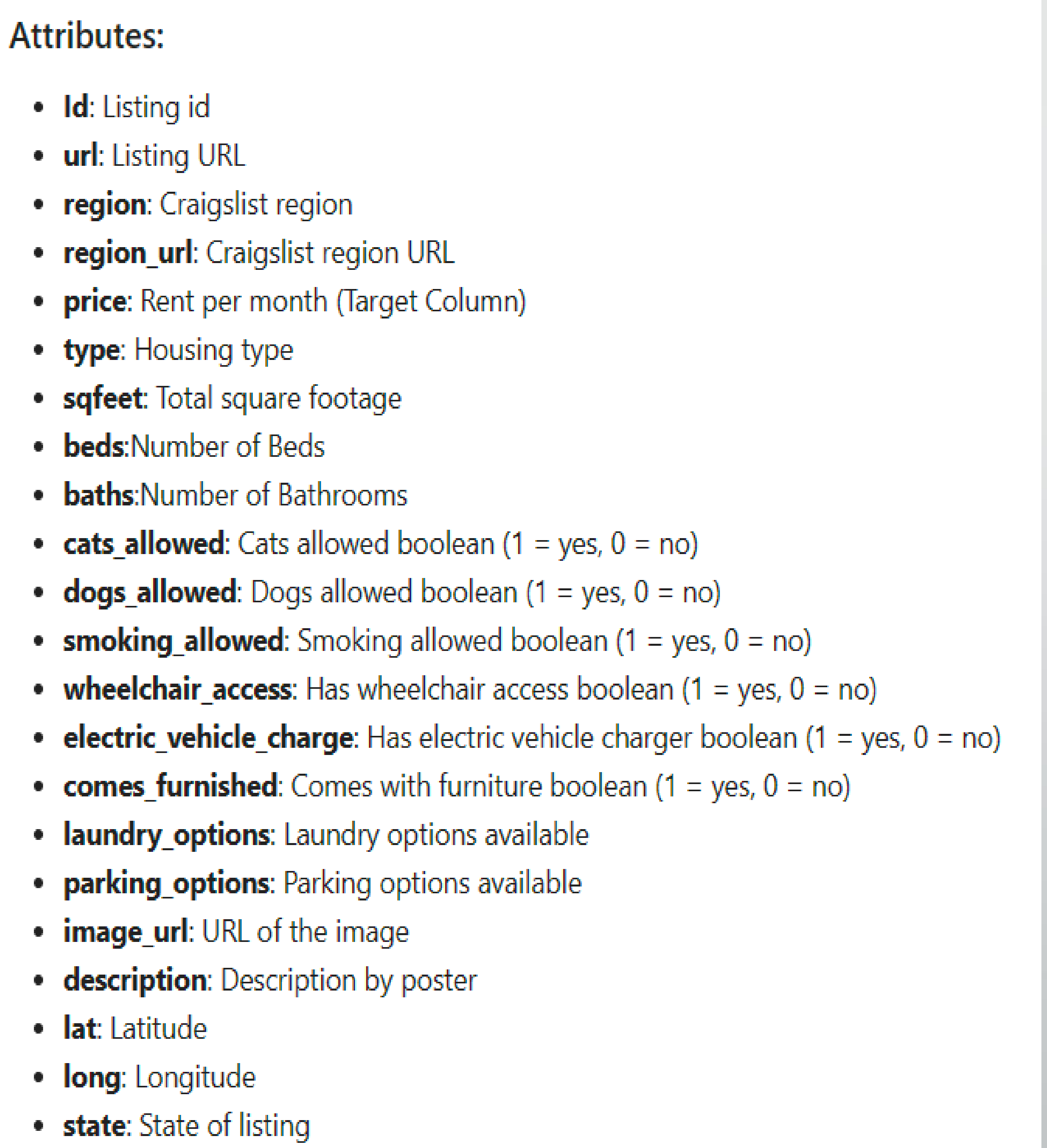
# My solution: RentWise

I aim to leverage my skills as a Data Scientist to develop a robust predictive model that can accurately estimate these rental prices, offering some insights for:

- renters
- landlords
- the ever-evolving real estate industry

# Dataset Introduction

- **Source:** Kaggle ([www.kaggle.com](http://www.kaggle.com))
- **No. of rows:** 384,977
- Each row depicting a **unique data entry** (rental property).
- **No Duplicated** rows
- **No. of columns:** 22
- **Different types of datatypes**: int64, float64 and objects.
- Had a lot of missing values in:
  1. **'laundry_options':** More than 20% (79,026)
  2. **'parking_options':** More than 36.5% (140,687)
  3. **'lat':** 4.9% (1,918)
  4. **'long':** 4.9% (1,918)

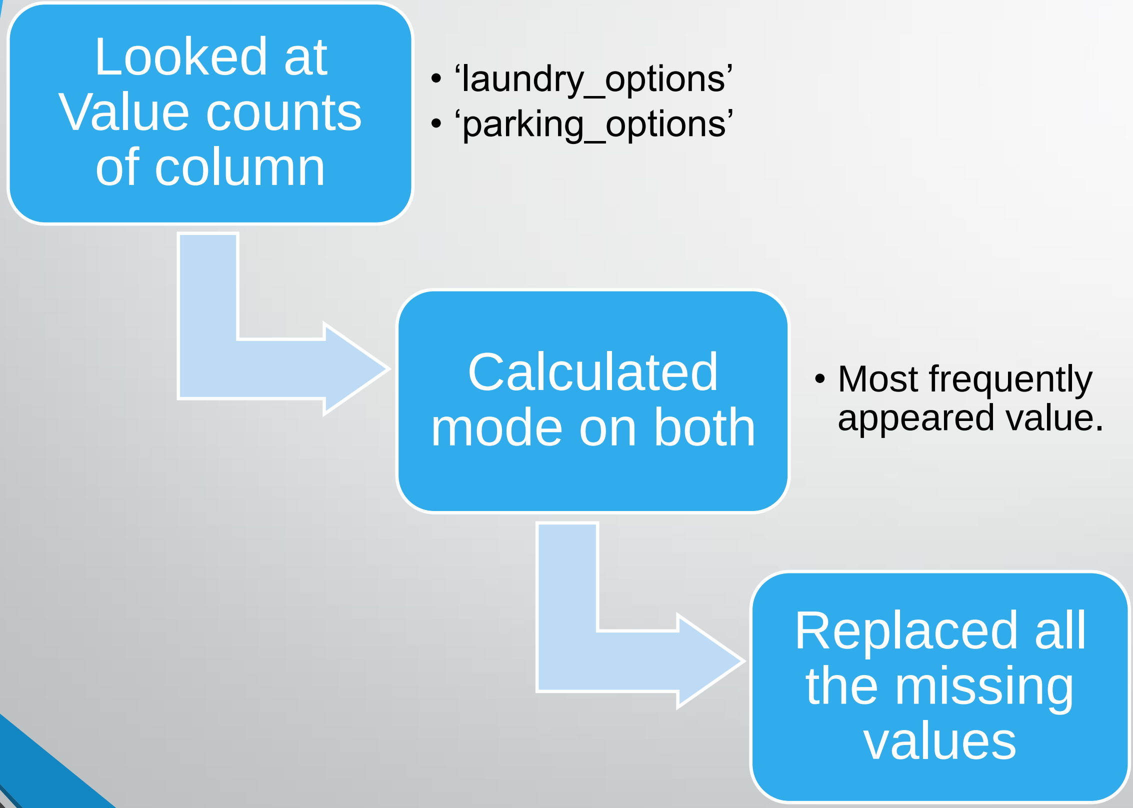


## Attributes:

- **Id**: Listing id
- **url**: Listing URL
- **region**: Craigslist region
- **region_url**: Craigslist region URL
- **price**: Rent per month (Target Column)
- **type**: Housing type
- **sqfeet**: Total square footage
- **beds**:Number of Beds
- **baths**:Number of Bathrooms
- **cats_allowed**: Cats allowed boolean (1 = yes, 0 = no)
- **dogs_allowed**: Dogs allowed boolean (1 = yes, 0 = no)
- **smoking_allowed**: Smoking allowed boolean (1 = yes, 0 = no)
- **wheelchair_access**: Has wheelchair access boolean (1 = yes, 0 = no)
- **electric_vehicle_charge**: Has electric vehicle charger boolean (1 = yes, 0 = no)
- **comes_furnished**: Comes with furniture boolean (1 = yes, 0 = no)
- **laundry_options**: Laundry options available
- **parking_options**: Parking options available
- **image_url**: URL of the image
- **description**: Description by poster
- **lat**: Latitude
- **long**: Longitude
- **state**: State of listing

# Data Cleaning Steps

## Dealing with Missing values



Looked at Value counts of column

- ‘laundry_options’
- ‘parking_options’

Calculated mode on both

- Most frequently appeared value.

Replaced all the missing values

- Checked and make sure

## Dropping columns

1. **‘lat’ and ‘long’:** Because they have such low percentage of missing values, we could safely drop them.

2. **‘id’:** was a unique identifier for each data point, providing no real value.

3. **‘url’, ‘region_url’ and ‘image_url’:** They all included image links, but a lot of them were similar, so we dropped them.

# Exploratory Data Analysis (EDA)

## With Outliers

### Distribution of ‘price’

min value: 0

Max value: 2,768,307,249.00

Mean: 8,897.79

### Distribution of ‘sqfeet’

Min value: 0

Max value: 8,388,607.00

Mean: 1,062.35

## Without Outliers

### Redistribution of ‘price’

Min value: 100

Max value: 5000

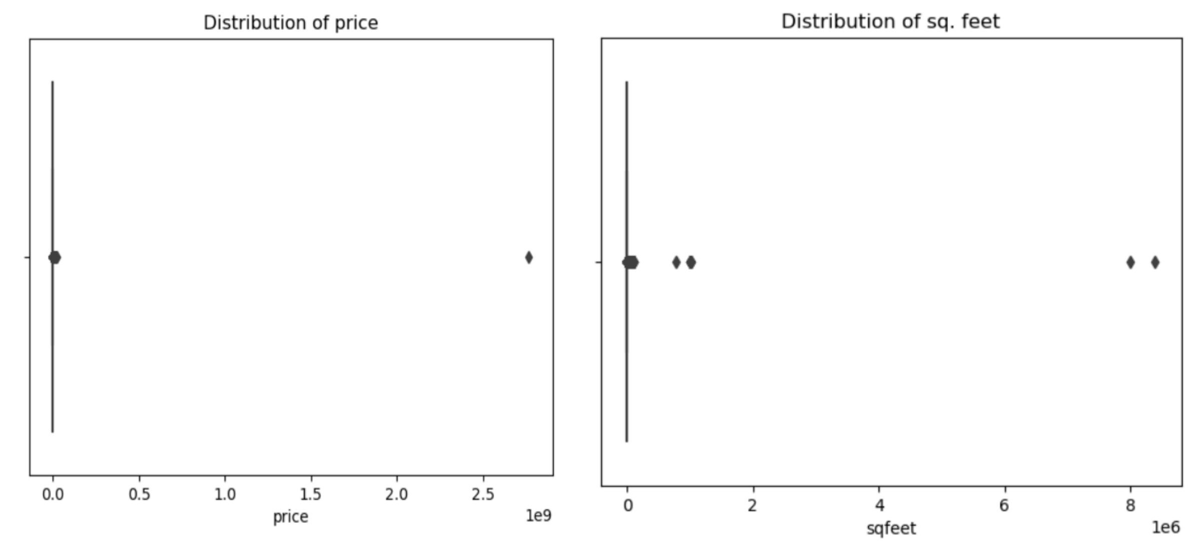
Mean: 1177.46

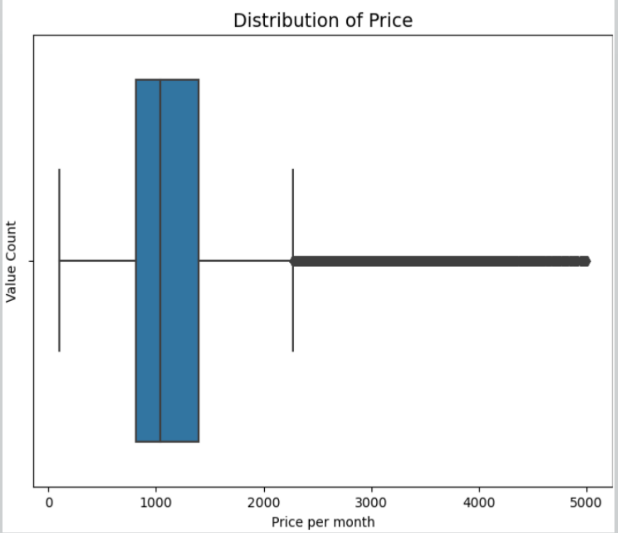
### Redistribution of ‘sqfeet’

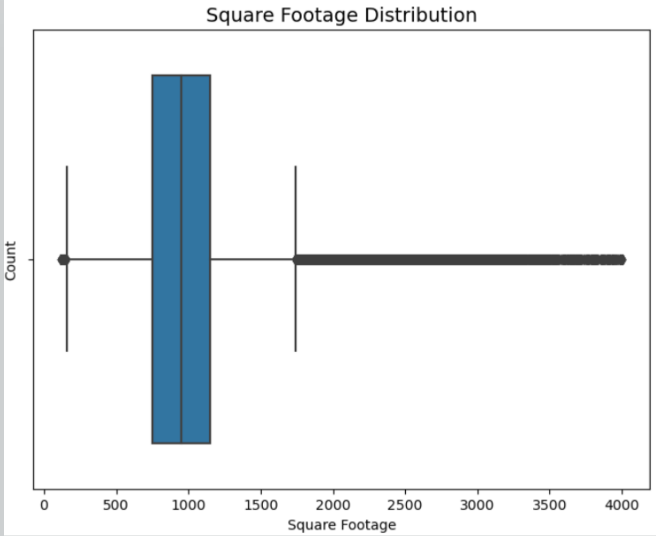
Min value: 120

Max value: 4000

Mean: 989.75



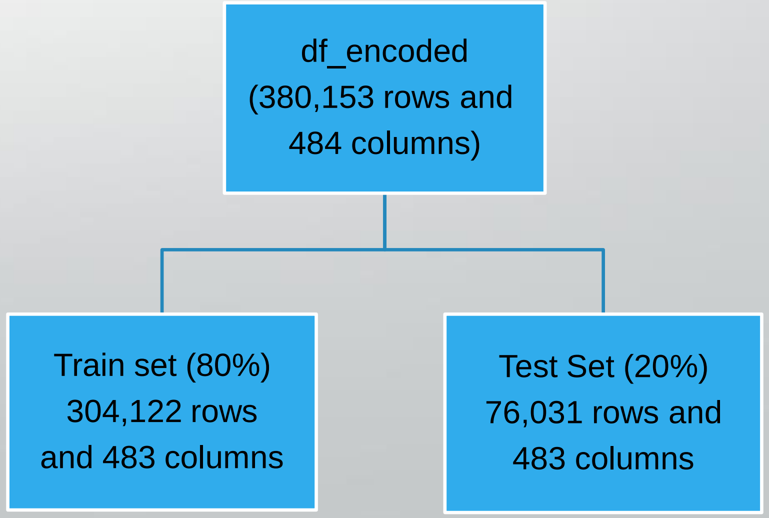




# Data Preprocessing

1. **Understanding Categorical Columns:**
   - Explored and analyzed categorical features in the dataset.
   - 'region', type', 'laundry_options', 'parking_options', and 'state'
2. **One-Hot Encoding:**
   - Applied one-hot encoding to transform categorical variables into a numerical format.
3. **Column Removal:**
   - Dropped the original categorical columns post one-hot encoding to enhance model readiness.
4. **Numerical Data Integration:**
   - Combined all numerical columns in a consolidated dataset named 'df_encoded.'.
5. **Train-Test Split:**
   - Segregated the dataset into training (80%) and testing (20%) subsets for model development and evaluation.

| Columns | Unique Values |
|---|---|
| ‘region’ | 404 |
| ‘type’ | 12 |
| ‘laundry_options’ | 5 |
| ‘parking_options’ | 7 |
| ‘state’ | 51 |

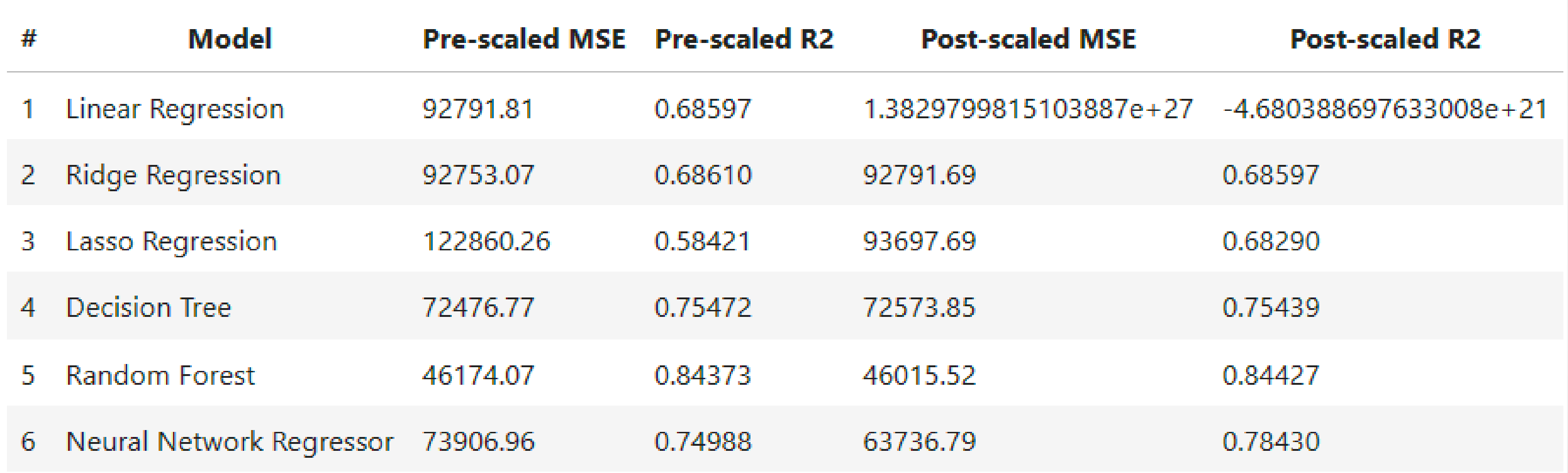


df_encoded (380,153 rows and 484 columns)

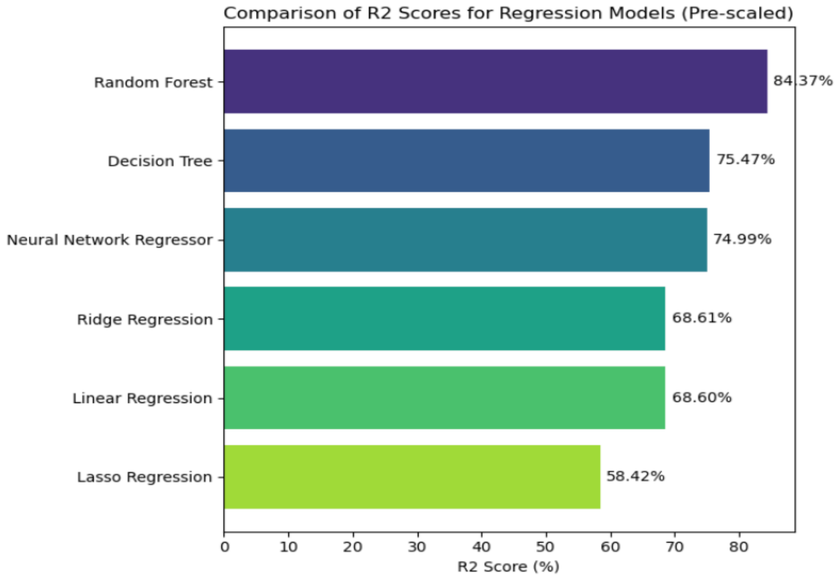
Train set (80%) 304,122 rows and 483 columns

Test Set (20%) 76,031 rows and 483 columns

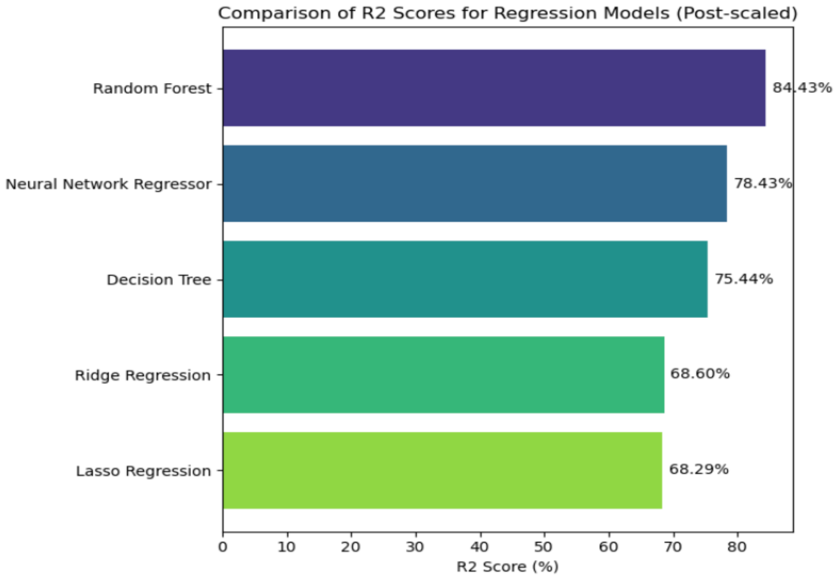
# Baseline Modelling



| # | Model | Pre-scaled MSE | Pre-scaled R2 | Post-scaled MSE | Post-scaled R2 |
|---|---|---|---|---|---|
| 1 | Linear Regression | 92791.81 | 0.68597 | 1.3829799815103887e+27 | -4.680388697633008e+21 |
| 2 | Ridge Regression | 92753.07 | 0.68610 | 92791.69 | 0.68597 |
| 3 | Lasso Regression | 122860.26 | 0.58421 | 93697.69 | 0.68290 |
| 4 | Decision Tree | 72476.77 | 0.75472 | 72573.85 | 0.75439 |
| 5 | Random Forest | 46174.07 | 0.84373 | 46015.52 | 0.84427 |
| 6 | Neural Network Regressor | 73906.96 | 0.74988 | 63736.79 | 0.78430 |



Comparison of R2 Scores for Regression Models (Pre-scaled)



Comparison of R2 Scores for Regression Models (Post-scaled)

**Top 3 models:**

1. Random Forest
2. Neural Network Regressor
3. Decision Tree

# Next steps

- **Best model selection:** My next steps involve choosing the best model.
- **Hyperparameter Tuning:** Utilizing techniques like grid search, cross-validation and Principal Component Analysis to find the best hyperparameter values and optimize my model’s performance.

Thank you !!!