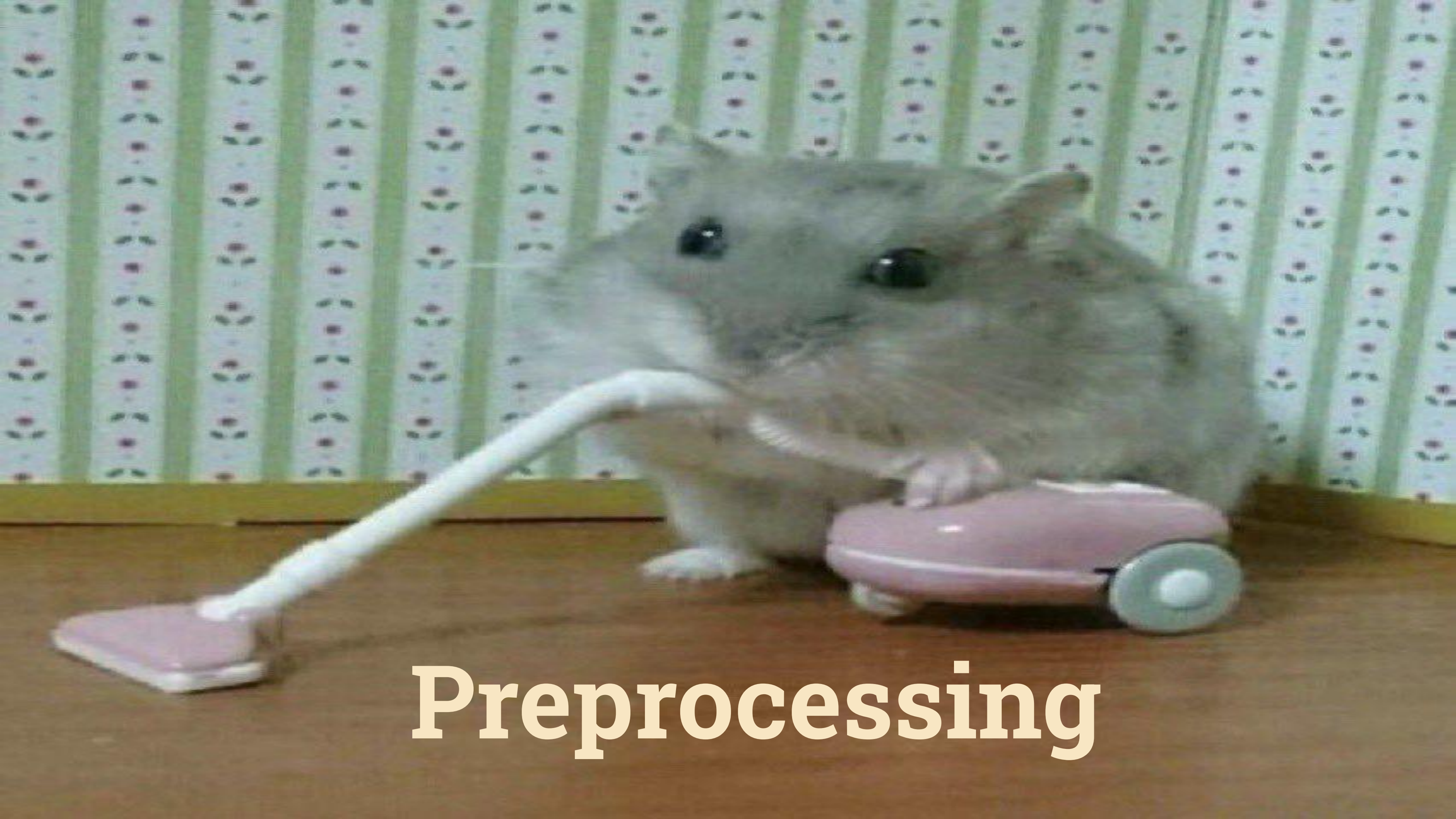
Machine Learning

Final presentation

Preprocessing



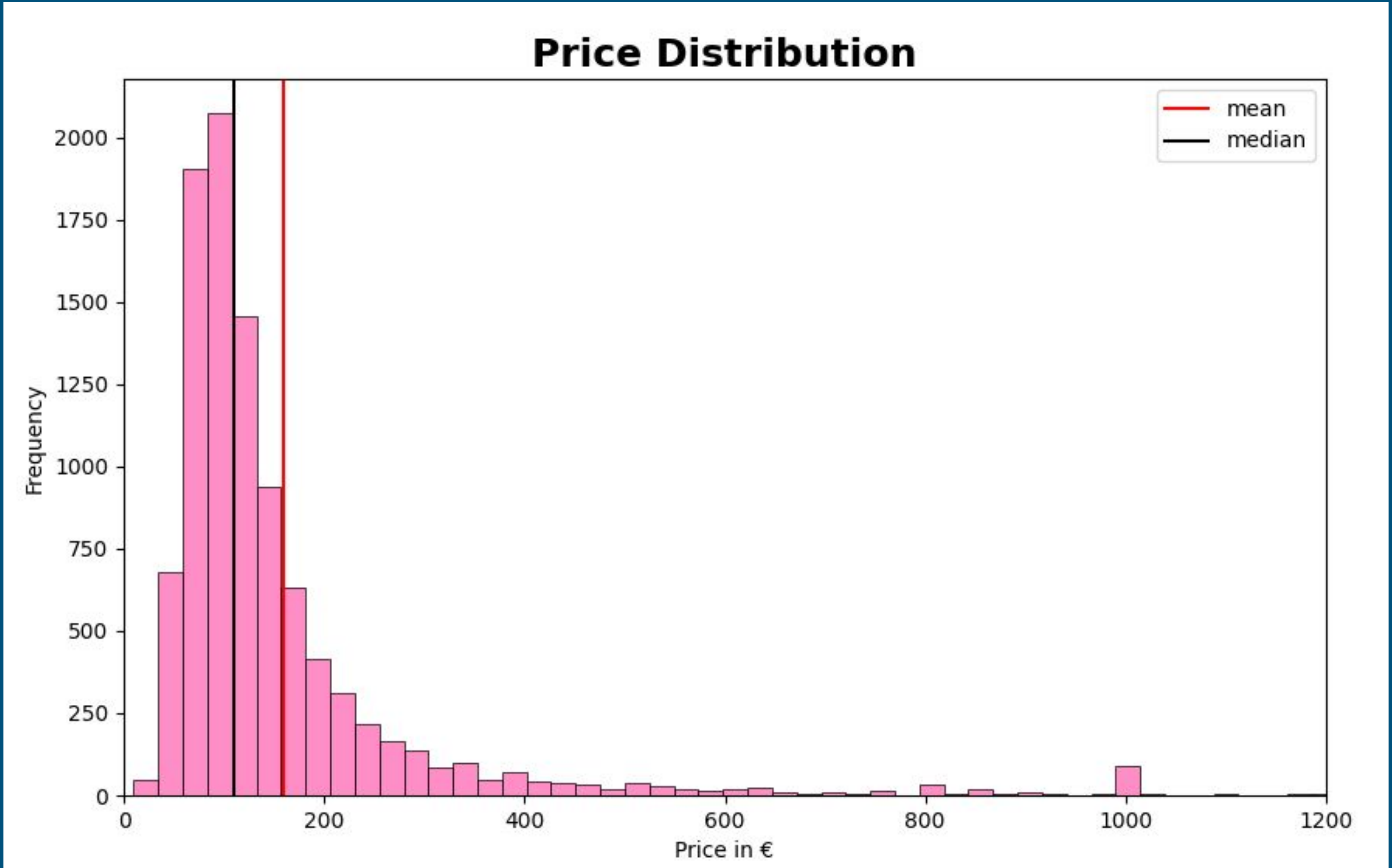
Data Cleaning steps

- remove columns we assume are unnecessary and have no effect on the price
- drop columns with high amounts of NaN
- reduce text columns to only numerical or boolean values
- use LabelEncoder() for numeric values
- remove outliers with a price above 2500€, empty URL links or unusual values
- turn amenities into a binary matrix and filter according to luxury keywords
- calculate the median price (calendar listings) and replace missing price values

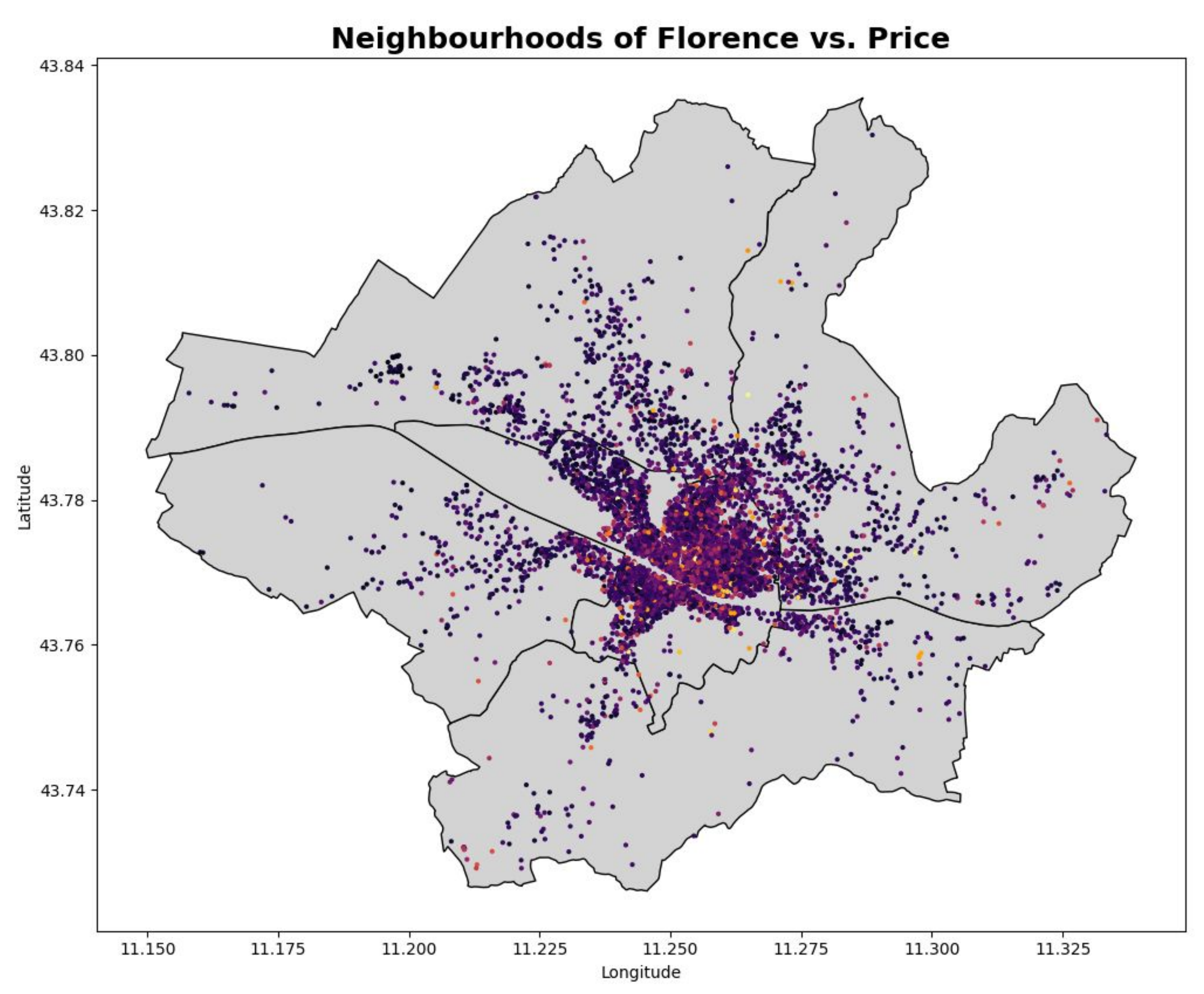
Data Engineering

- Sentiment score based on listings reviews
- calculate distance to city center

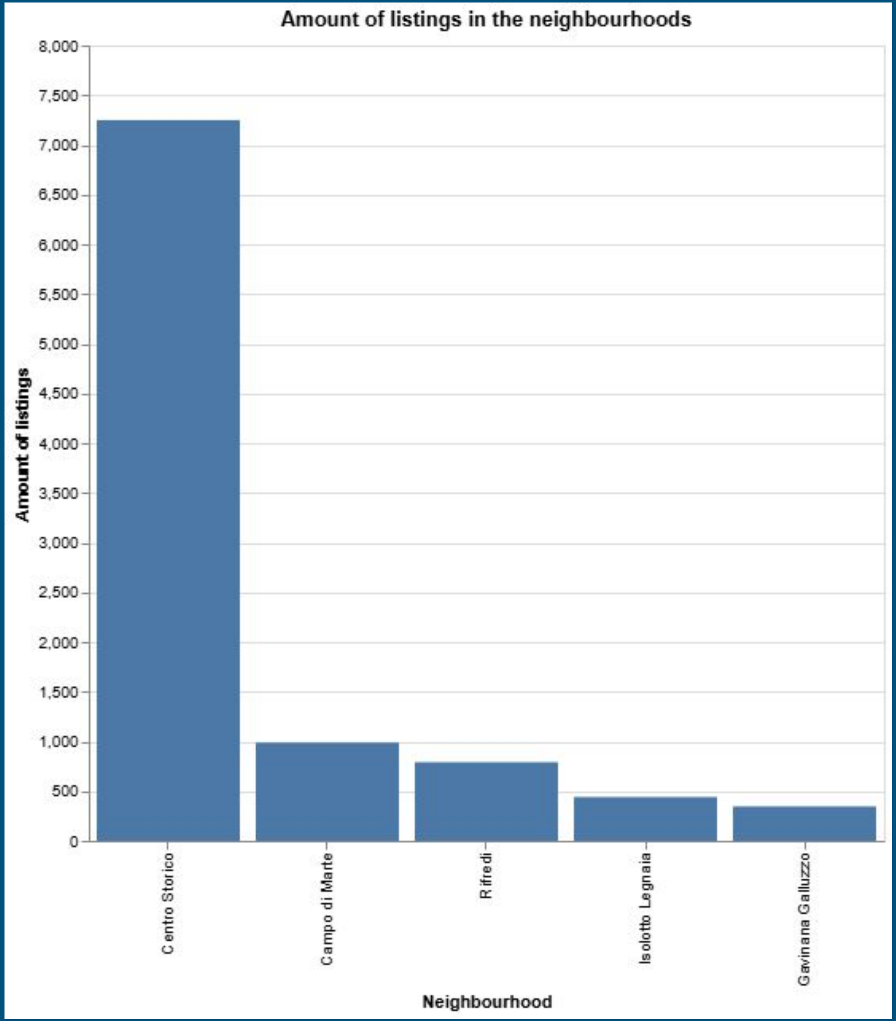
Visualization



Price Distribution
mean
median
Frequency
0
250
500
750
1000
1250
1500
1750
2000
0
200
400
600
800
1000
1200
Price in €

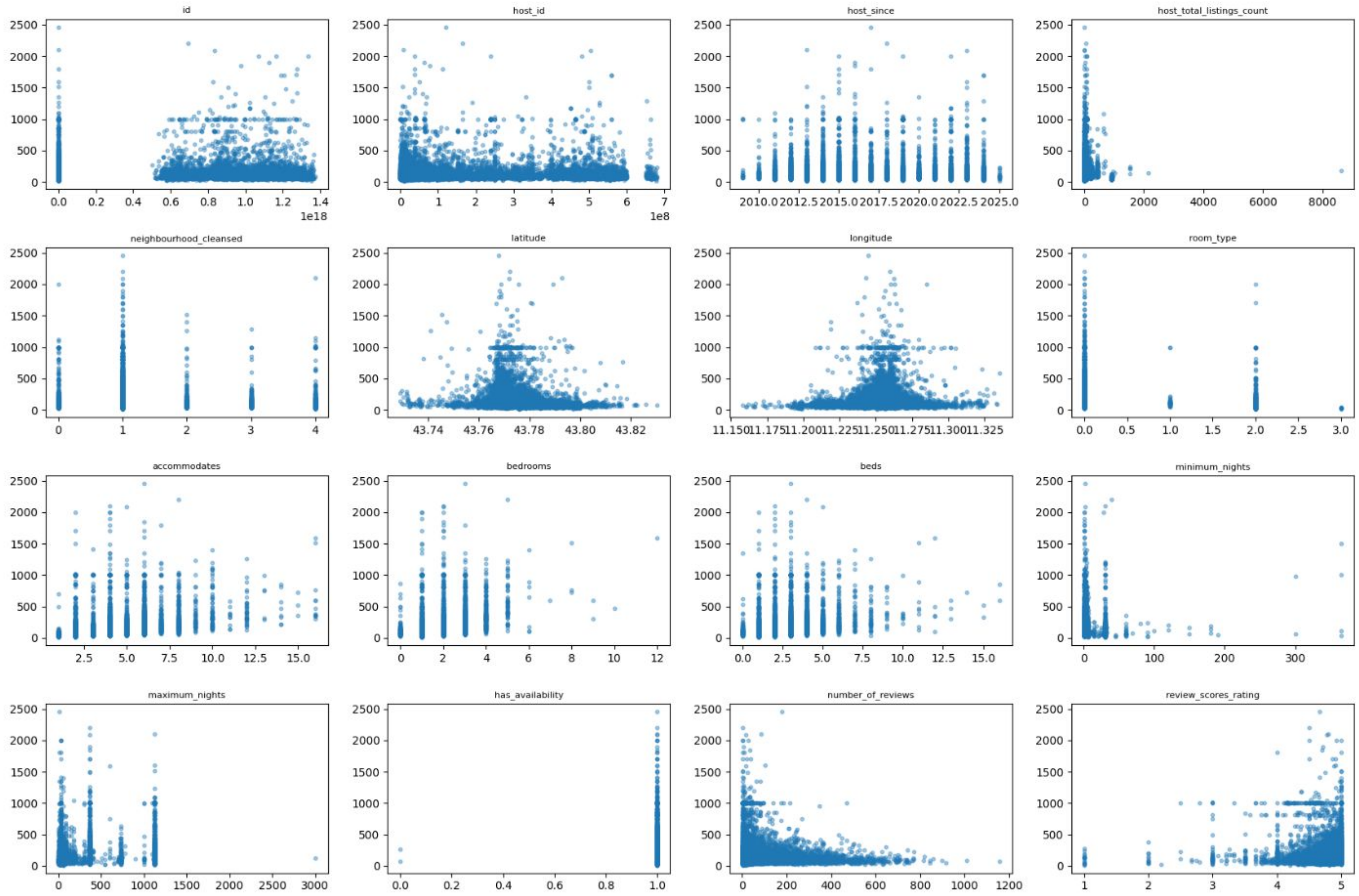


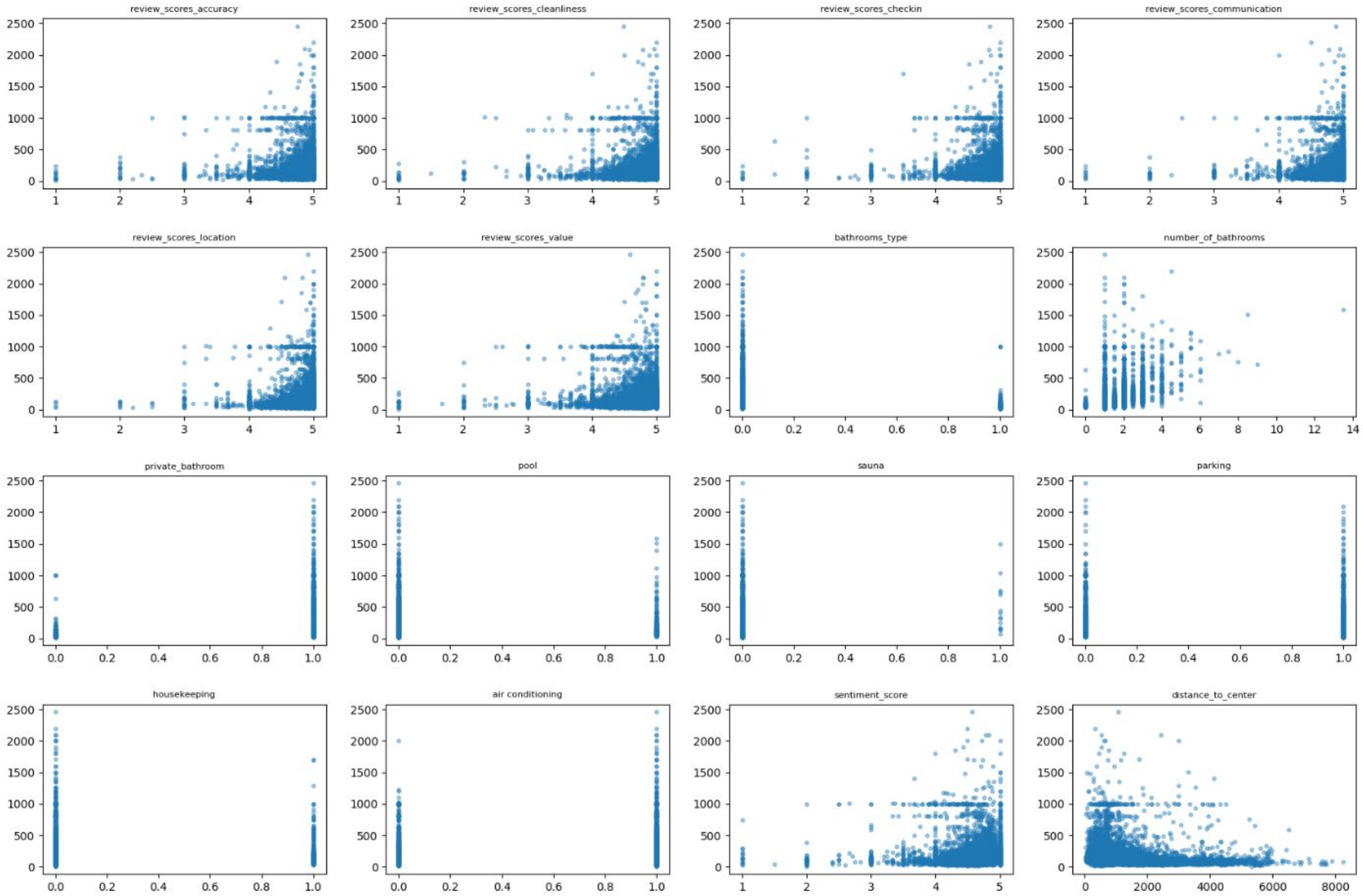
Neighbourhoods of Florence vs. Price
Latitude
Longitude
43.84
43.82
43.80
43.78
43.76
43.74
11.150
11.175
11.200
11.225
11.250
11.275
11.300
11.325

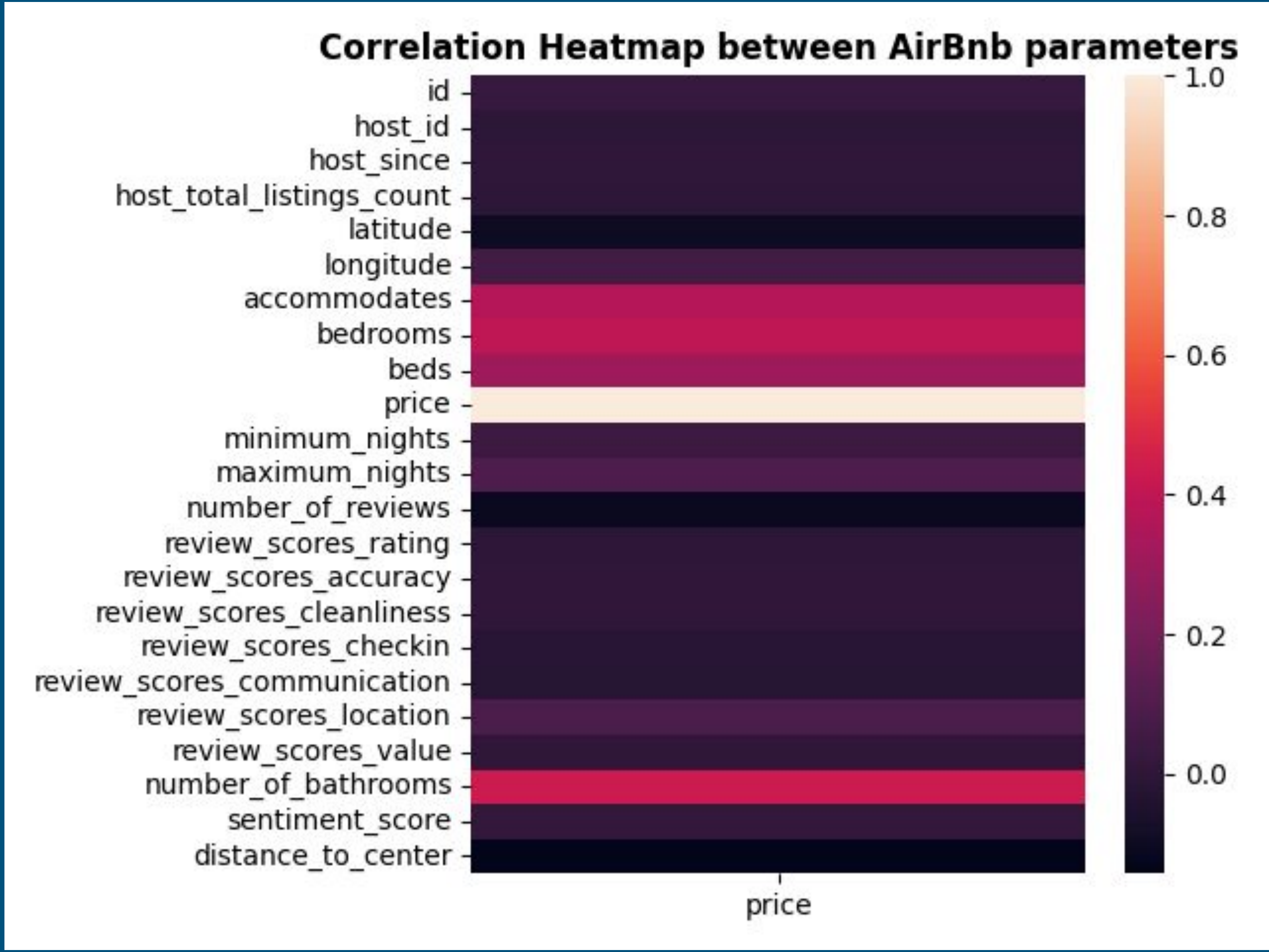


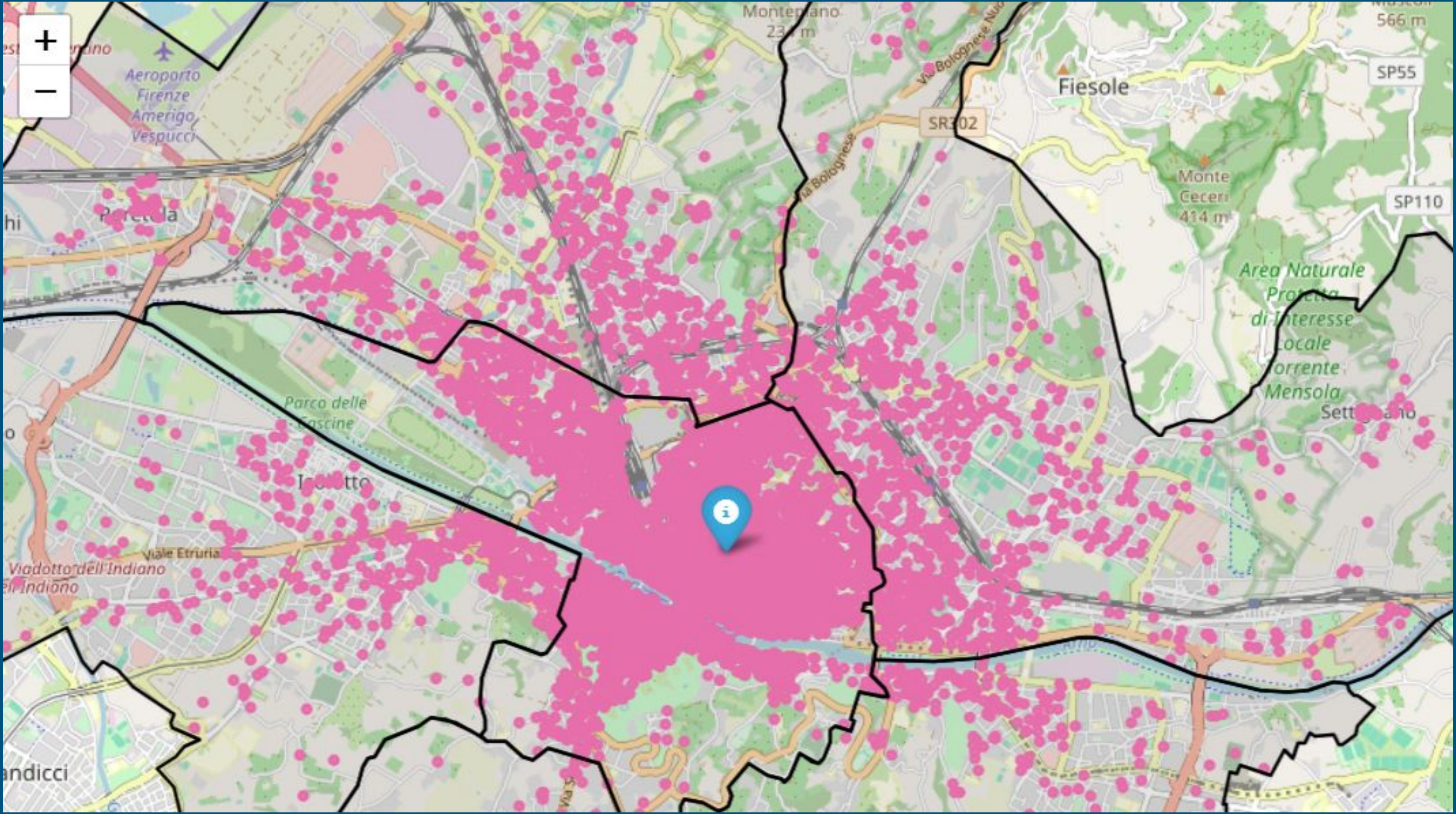
Amount of listings in the neighbourhoods
Amount of listings
8,000
7,500
7,000
6,500
6,000
5,500
5,000
4,500
4,000
3,500
3,000
2,500
2,000
1,500
1,000
500
0
Centro Storico
Campo di Marte
Rifredi
Isolotto Legnaia
Gavinana Galluzzo
Neighbourhood

AirBnB attributes vs. Price



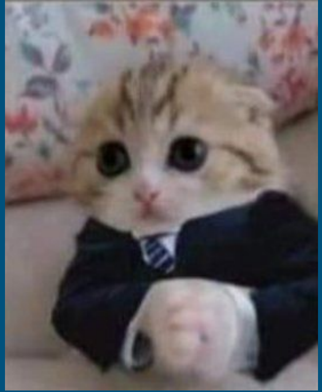

review_scores_accuracy
review_scores_cleanliness
review_scores_checkin
review_scores_communication
review_scores_location
review_scores_value
bathrooms_type
number_of_bathrooms
private_bathroom
pool
sauna
parking
housekeeping
air conditioning
sentiment_score
distance_to_center


Correlation Heatmap between AirBnb parameters
id
host_id
host_since
host_total_listings_count
latitude
longitude
accommodates
bedrooms
beds
price
minimum_nights
maximum_nights
number_of_reviews
review_scores_rating
review_scores_accuracy
review_scores_cleanliness
review_scores_checkin
review_scores_communication
review_scores_location
review_scores_value
number_of_bathrooms
sentiment_score
distance_to_center
price
1.0
0.8
0.6
0.4
0.2
0.0


Aeroporto
Firenze
Amerigo
Vespucci
Fiesole
SR302
SP55
SP110
Monte
Ceceri
414 m
566 m
Parco delle
Cascine
Viale Etruria
Viadotto dell'Indiano
Area Naturale
Protetta
di Interesse
Locale
Torrente
Mensola
Arno

Model training

Background Knowledge



Loss functions:

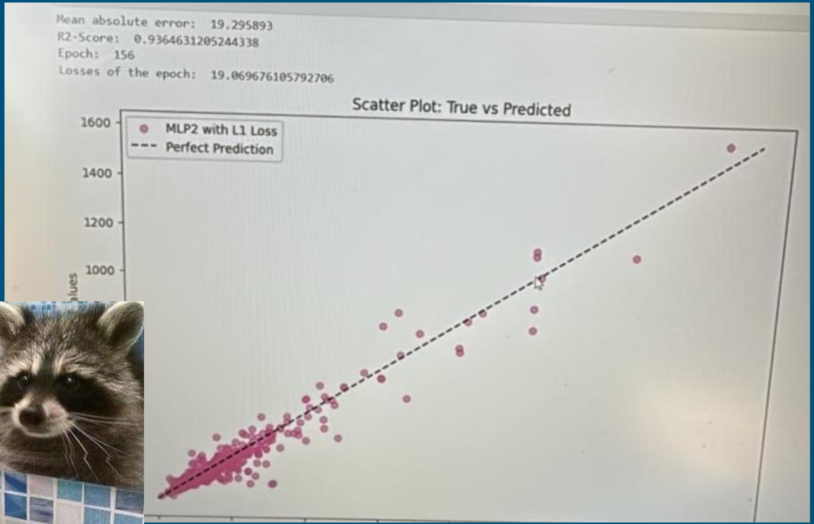
- **L1 Loss (MAE)**: average absolute difference between predicted and true values -> robust to outliers, harder to optimize in gradient based methods
- **L2 Loss (MSE)**: penalizes large errors more than small ones -> sensitive to outliers, for regression task
- **RMSE Loss**: same unit as targets -> easier to interpret, still sensitive to outliers
- **Huber Loss**: combines L1 and L2 Loss -> less sensitive to outliers than MSE, easier to optimize than MAE

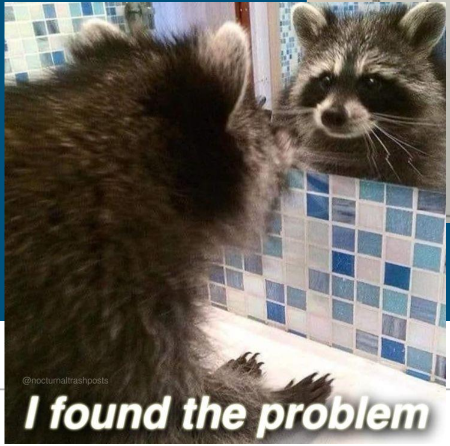
Parameters:

- **learning rate**: controls how fast the model learns,
  - too high -> overshooting, unstable, too low -> slow convergence
- **weight decay**: adds penalty on model weights to prevent overfitting

Data Leakage Problem

- We thought we had the perfect MLP
- R^2: 0.9+
- Checked everything
- normalization on the whole data frame
- Solution:





Ostrich algorithm

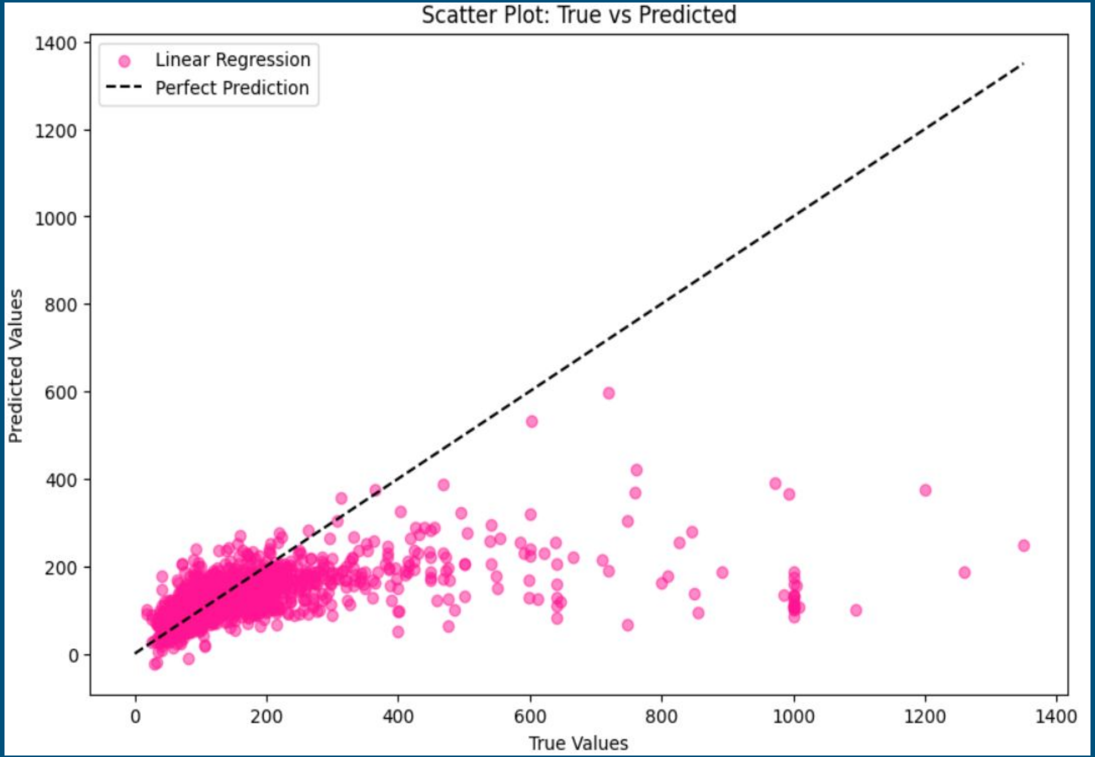
14 languages

Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

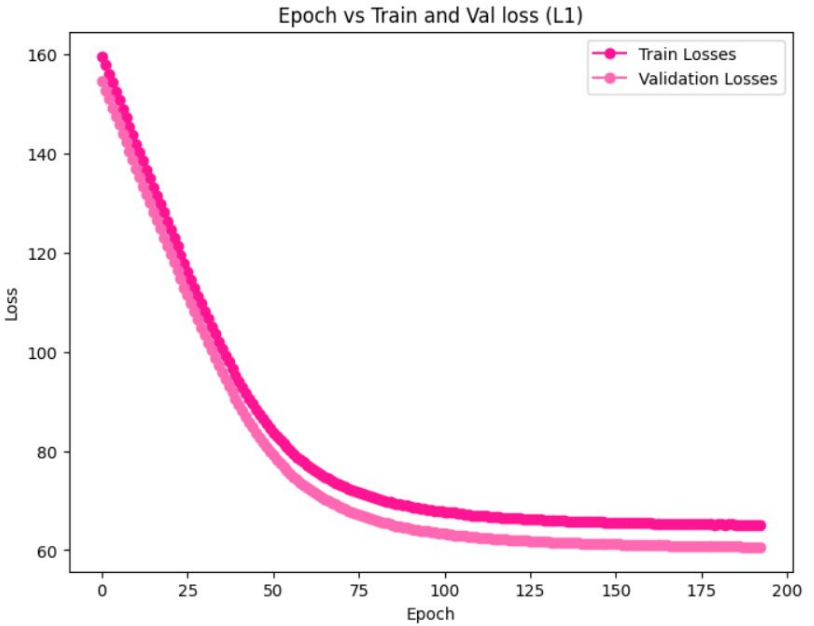
In computer science, the **ostrich algorithm** is a strategy of ignoring potential problems on the basis that they may be exceedingly rare. It is named after the ostrich effect which is defined as "to stick one's head in the sand and pretend there is no problem". It is used when it appears the situation may be more cost-effectively managed by allowing the problem to continue to occur rather than to attempt its prevention.

Linear Regression

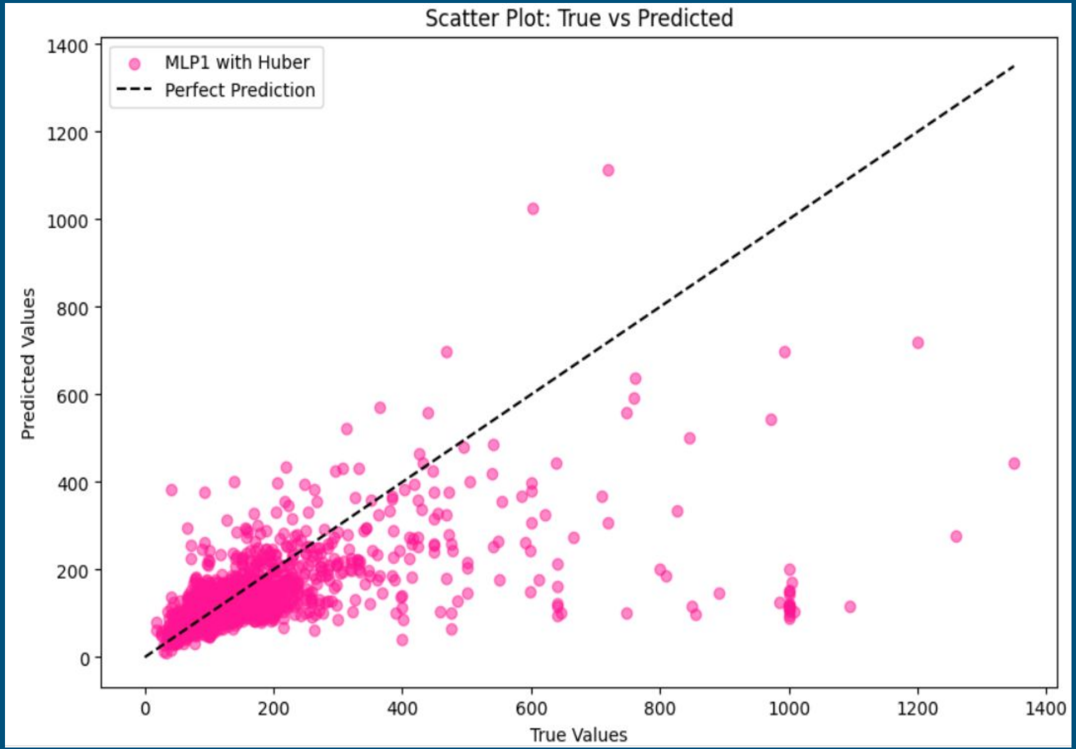


Hyperparameters

- Loos Function: L1 Loss
- Learning Rate: 0.008
- Optimizer: SGD

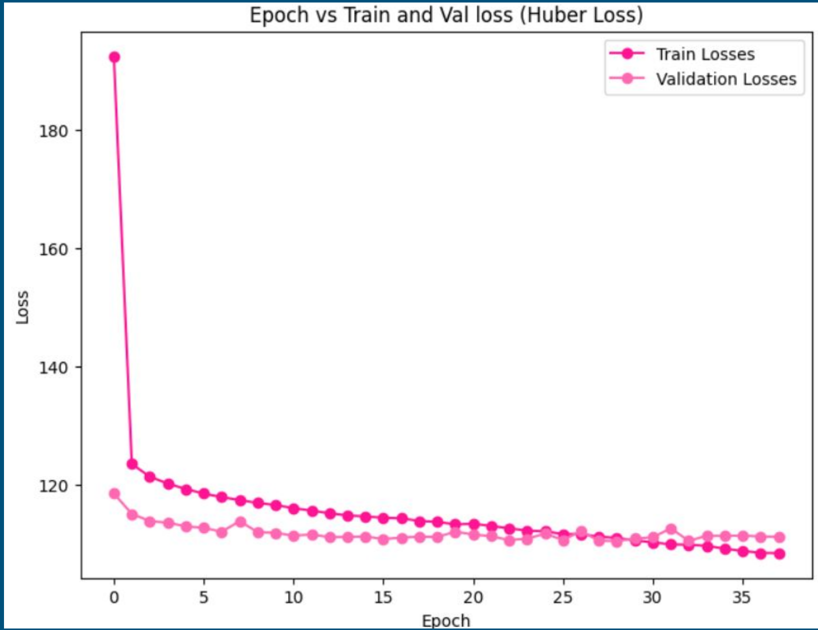


MLP1 - 4 Layer Perceptron

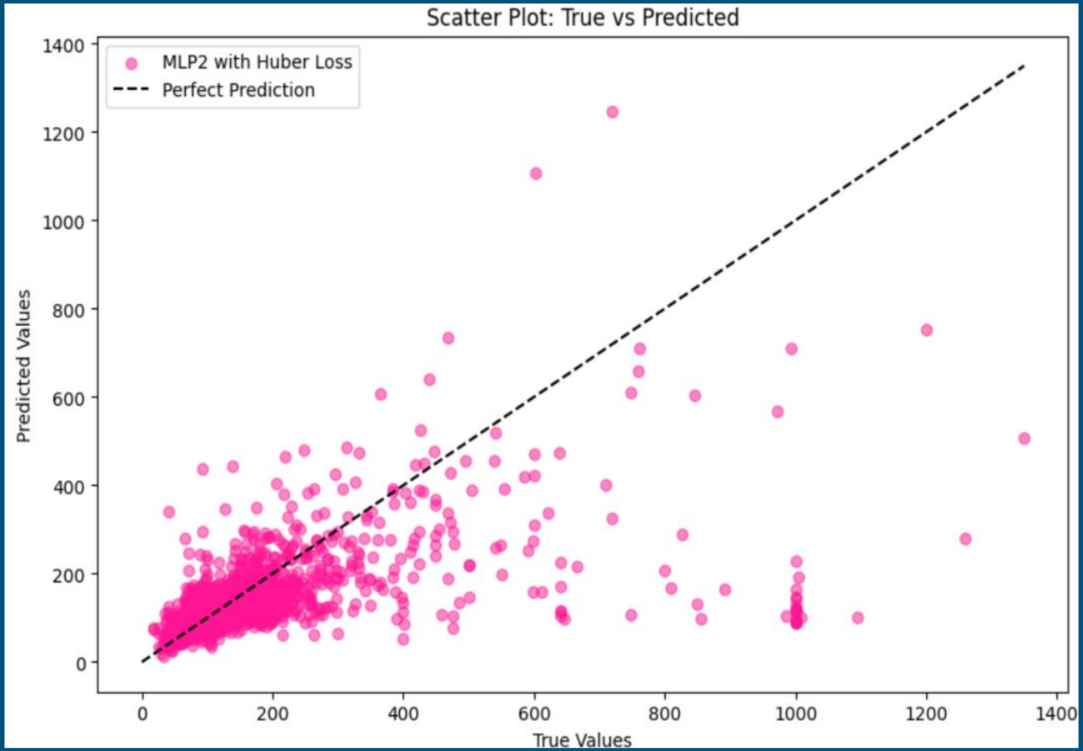


Hyperparameters

- Loss Function: Huber Loss
- Learning Rate: 0.001
- Optimizer: Adam
- weight decay: 0.0005

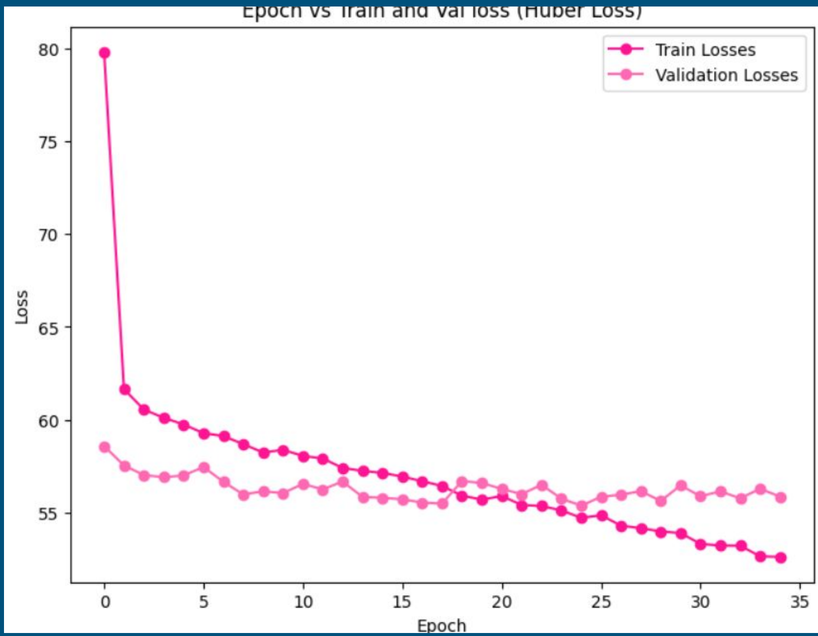


MLP2 - 5 Layer Perceptron

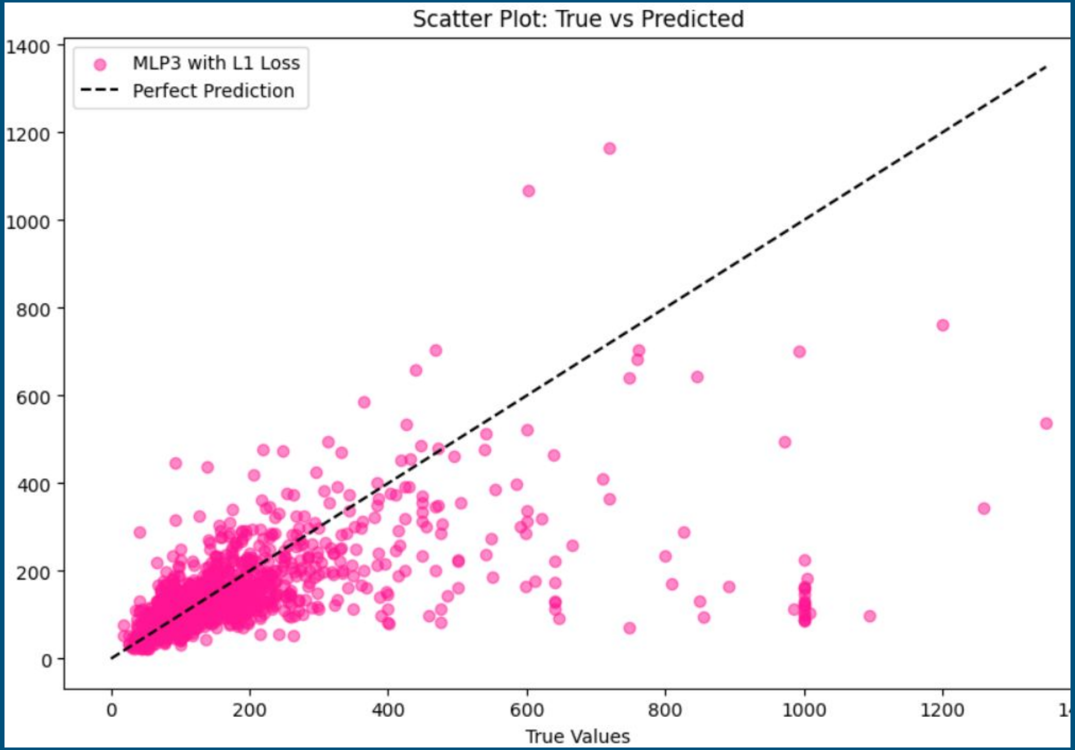


Hyperparameters

- Loos Function: L1 Loss
- Learning Rate: 0.001
- Optimizer: Adam
- weight decay: 0.0005

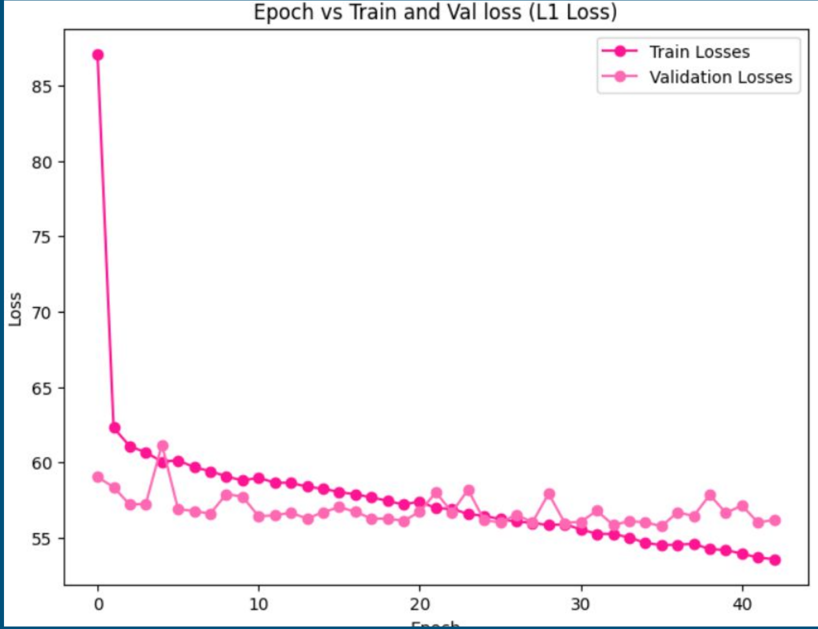


MLP3 - 6 Layer Perceptron

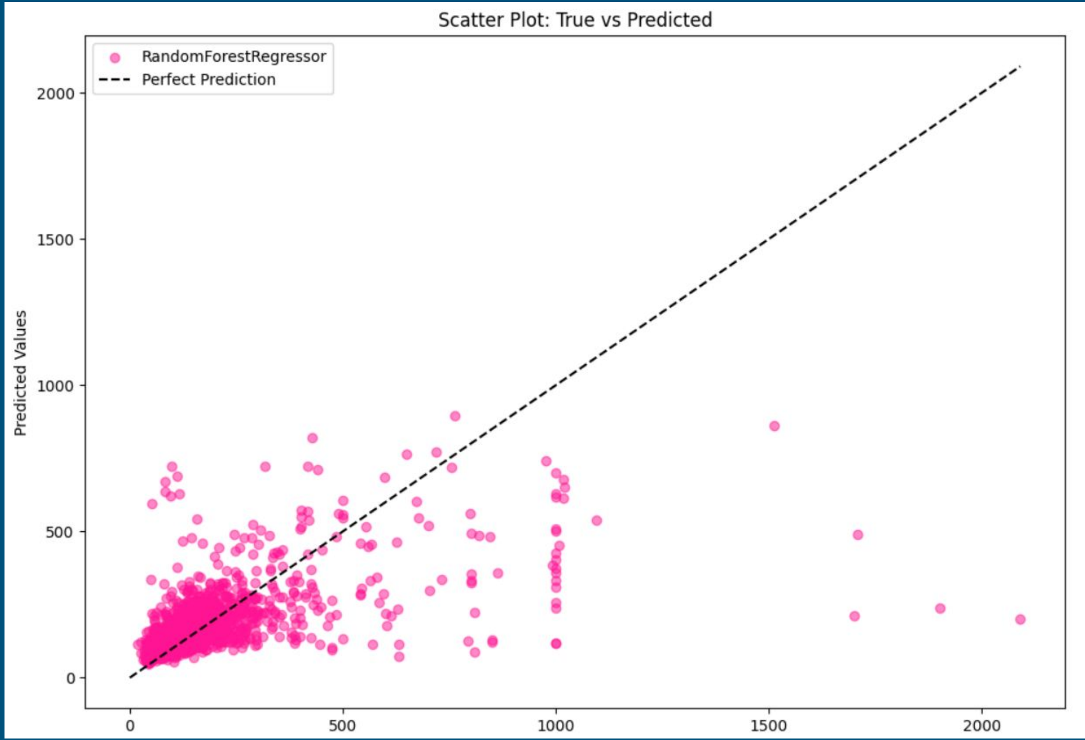


Hyperparameters

- Loos Function: L1 Loss
- Learning Rate: 0.001
- Optimizer: Adam
- weight decay: 0.005



RandomForestRegressor

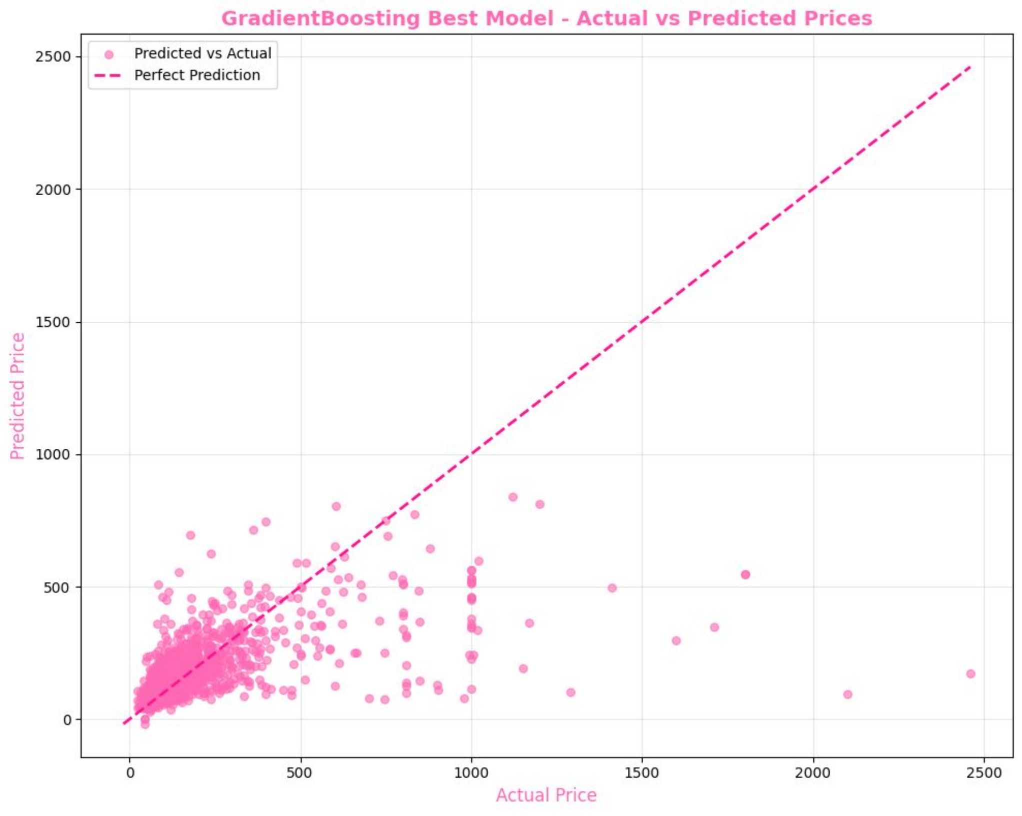


Hyperparameters:

- **max_depth = 10**
- **max_features = 30**
- **min_samples_leaf = 4**
- **n_estimators = 500**

→ found with GridSearchCV()

Gradientboosting Regressor

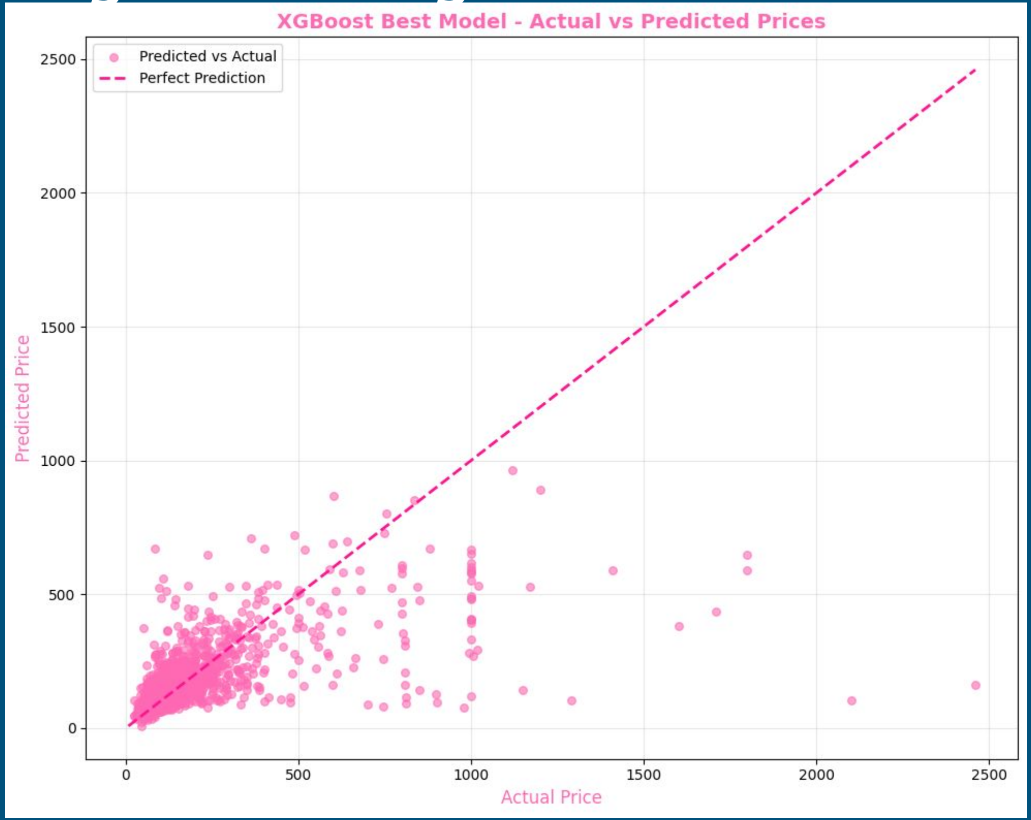


Hyperparameters:

- **n_estimators = 500**
- **min_samples_split = 10**
- **min_samples_leaf = 15**
- **max_depth = 4**
- **learning_rate = 0.05**
- **subsamples = 0.8**

→ found with RandomizedSearchCV()

XgboostRegressor



Hyperparameters:

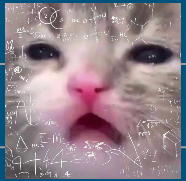
- **n_estimators = 500**
- **min_child_weight = 5**
- **max_depth = 5**
- **learning_rate = 0.03**
- **reg_alpha = 0.1**
- **reg_lambda = 10**

→ found with RandomizedSearchCV()

Scores Comparison

Comparison

Model Name	R^2	MAE	R^2 with PCA	MAE with PCA
Linear Regression	0.18	61.23	0.024	58.83
MLP1	0.27	56.81	0.26	64
MLP2	0.27	56.37	0.24	51.41
MLP3	0.29	56.31	0.24	51.1
RandomForestRegressor	0.37	62.9	-	-
GradientBoosting	0.36	65.38	-	-
XGBoost	0.40	62.30	-	-



→results from the latest model run

Evaluation

Evaluation

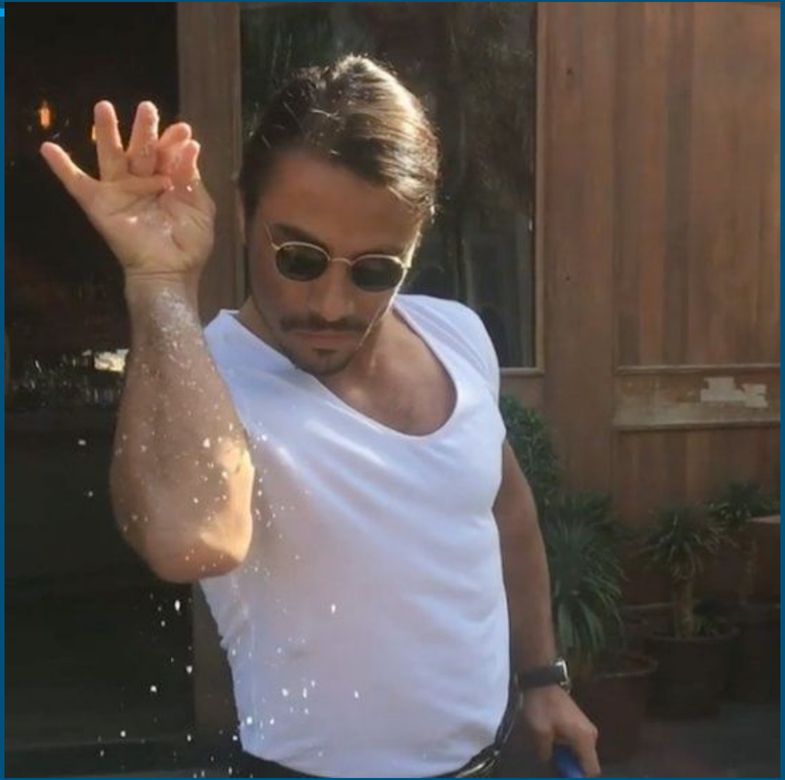
Problems we’ve encountered:

- normalized with the wrong mean and standard deviation
- skewed results due to specific outliers
- missing price values
- urls that lead to nothing
- the models tend to overfit → change loss function or early stopping patience
- after data cleaning we only had 9000 rows left

Our insights:

- small difference between using PCA or not, since we do not have a high dimensionality
- we didn’t use PCA for the trees, since it doesn’t result in a huge difference
- different optimizers and losses
- reduced the amount of amenities because of suspected low influence on the price

Conclusion



We cooked

or we were cooked...

Sources

- 3 Big Benefits of Data Cleansing - Teraflow.ai (Data Cleaning Picture)
- Forbes Model Training (Model Training Picture)
- Ostrich algorithm - Wikipedia (Ostrich)