3A · SD9
ApprAuto

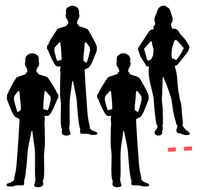
Berlin Price Prediction

Seong Woo AHN · Paul BÉRARD · Leila BERRADA · Tanguy BLEVARCQUE

Table of contents

1 Introduction

Us



Available data

CSV file **train_airbnb_berlin.csv**
with characteristics of the accommodations **features**
and their price per night **target variable**

15 692 instances

38 features

Our method

Process the data
Use appropriate ML approaches taught in ApprAuto:
Linear Regression | Random Forest | AdaBoost etc.
Compare their performance using relevant metrics
MSE | RMSE | MAE | Adjusted-R^2

Goal

Predict the price of 1 night's accommodation on Airbnb in the city of Berlin

2 Data exploration | Missing values

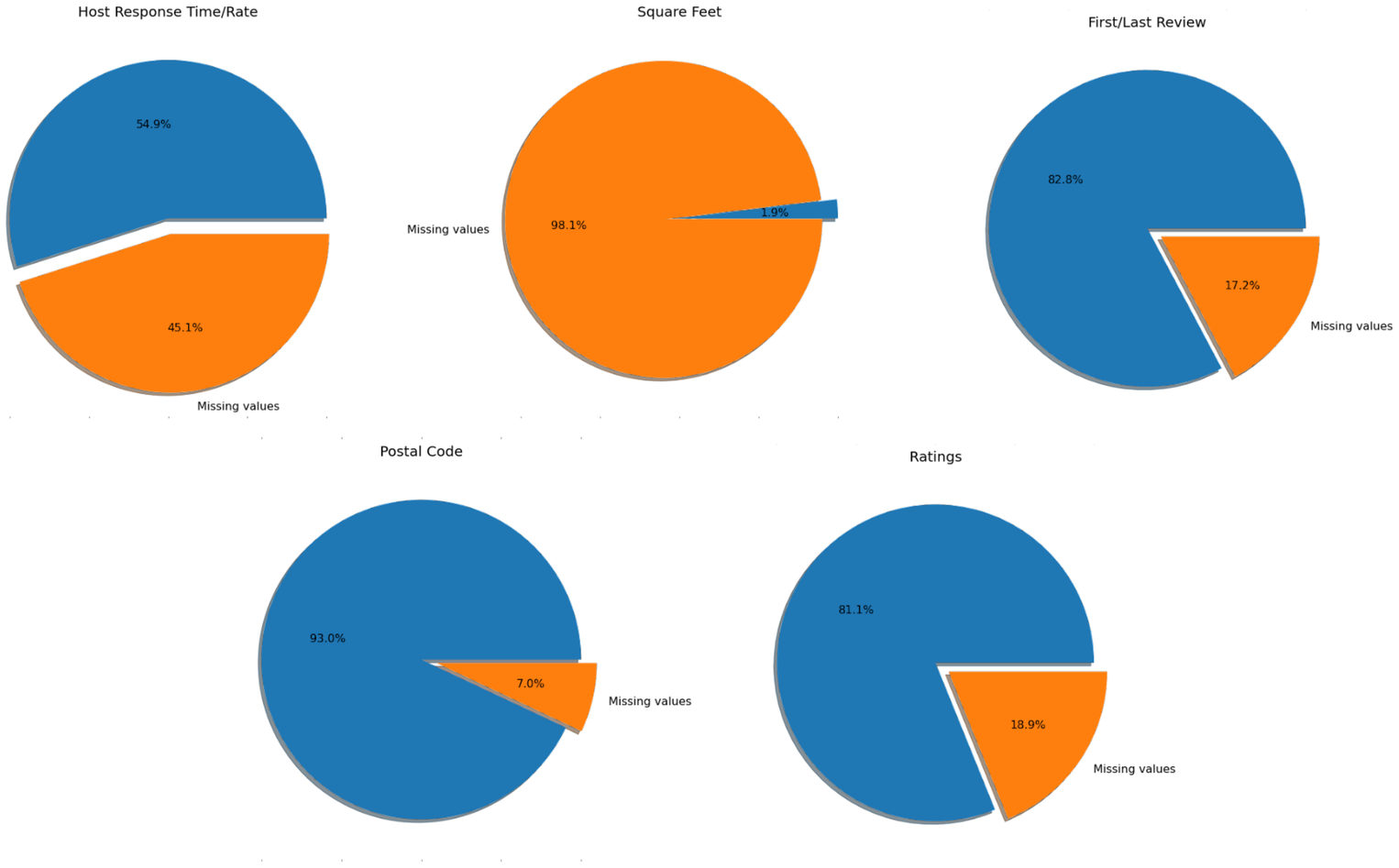


Fig. 1: Pie charts showing the proportion of missing values for some features

2 Data exploration | Map of Berlin

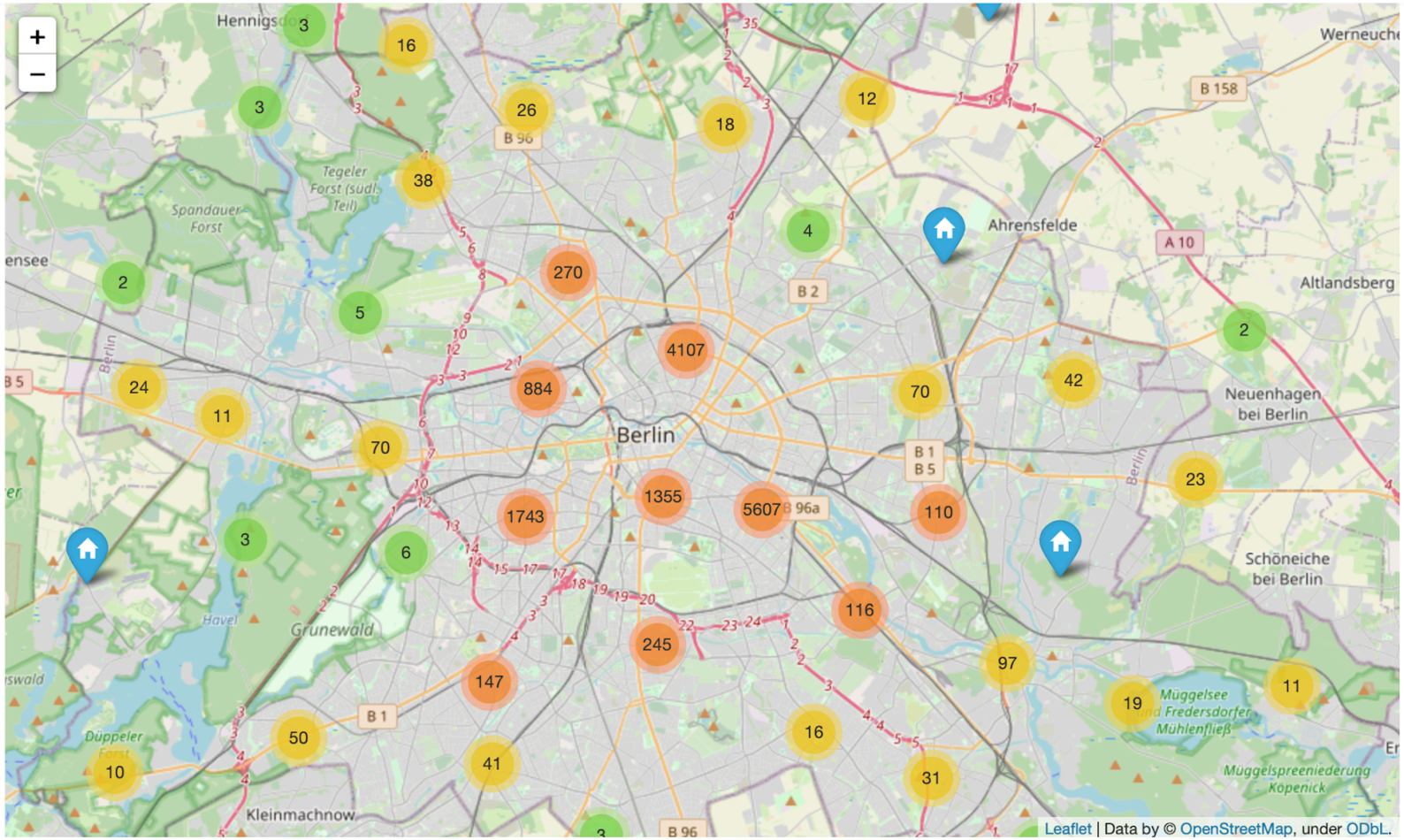


Fig. 2: Map of Berlin augmented with the location of all accommodations in the dataset

2 Data exploration | Distribution of prices

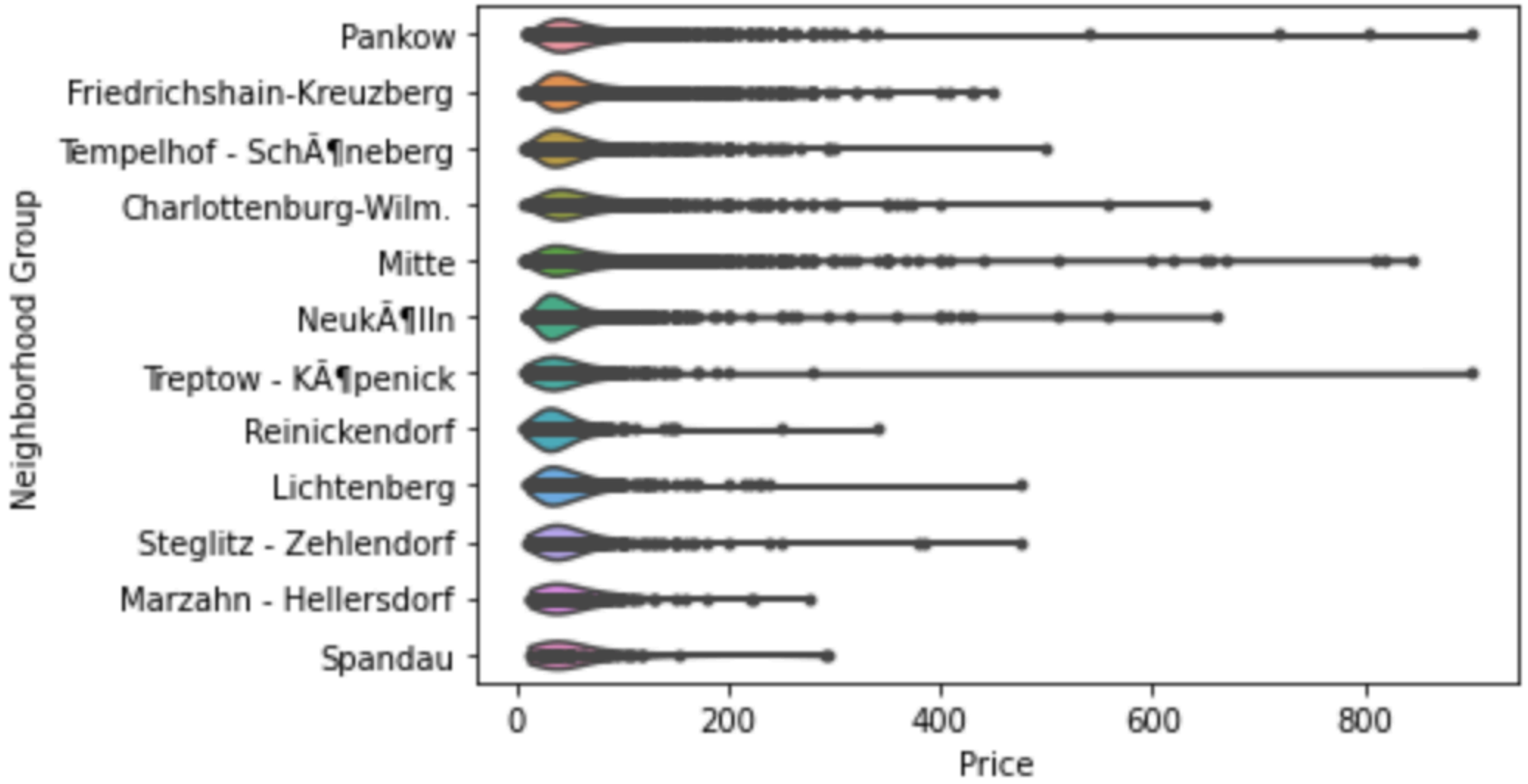


Fig. 3: Density and distribution of prices for each 'Neighborhood Group'

3 Data preprocessing | Preliminary + specific preprocessing

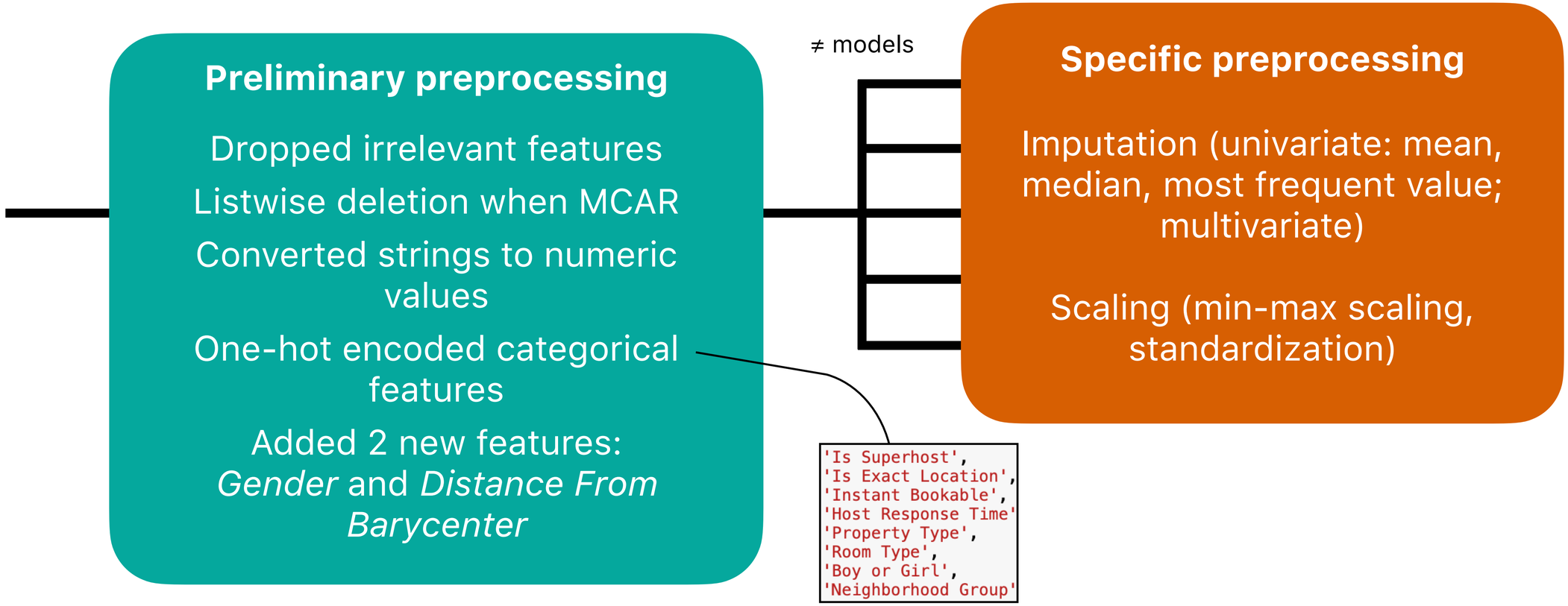


Fig. 4: The two phases of data preprocessing

3 Data preprocessing | Train/test split

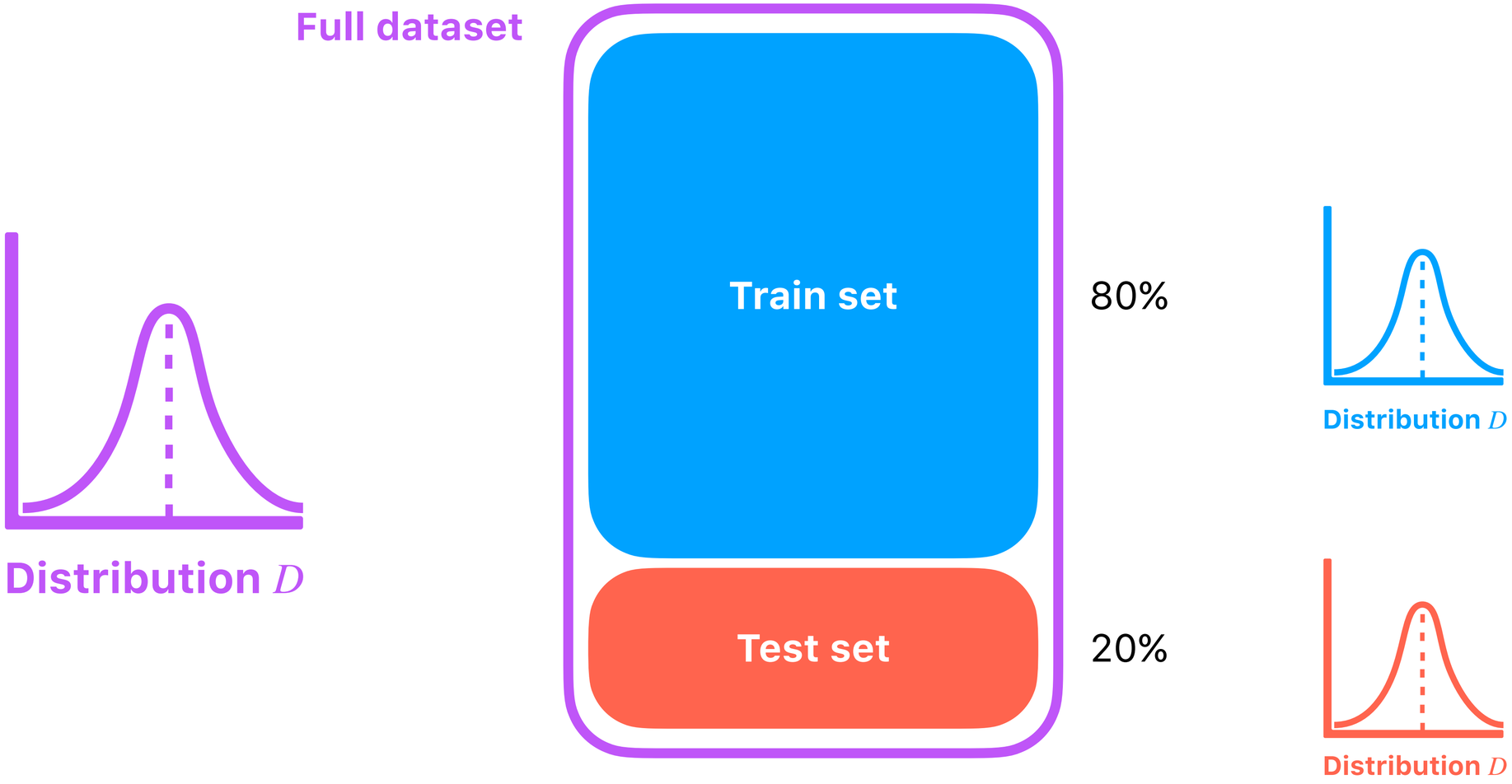


Fig. 5: Train/test split (80/20) of the dataset

4

Price prediction models

Linear Regression

Decision Tree

Bayesian Ridge Regression

K-Nearest Neighbors

Random Forest

Support Vector Regression

AdaBoost

XGBoost

4 Linear Regression | Brief explanation

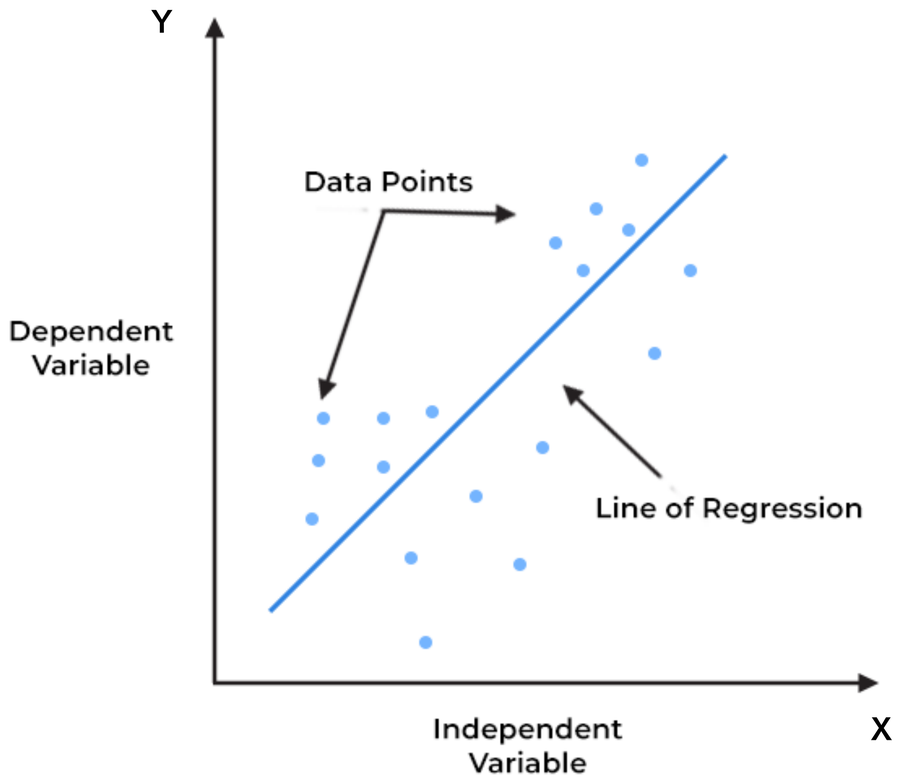
With n the number of instances and d the number of features (dimension):

$$\begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} x_{1,1} & \cdots & x_{1,d} \\ \vdots & \ddots & \vdots \\ x_{n,1} & \cdots & x_{n,d} \end{pmatrix} \begin{pmatrix} \theta_1 \\ \vdots \\ \theta_d \end{pmatrix}$$

Prices

Columns = features
Accommodates, Ratings, etc.

Lines = instances



4 Linear Regression | Imputation and scaling strategy

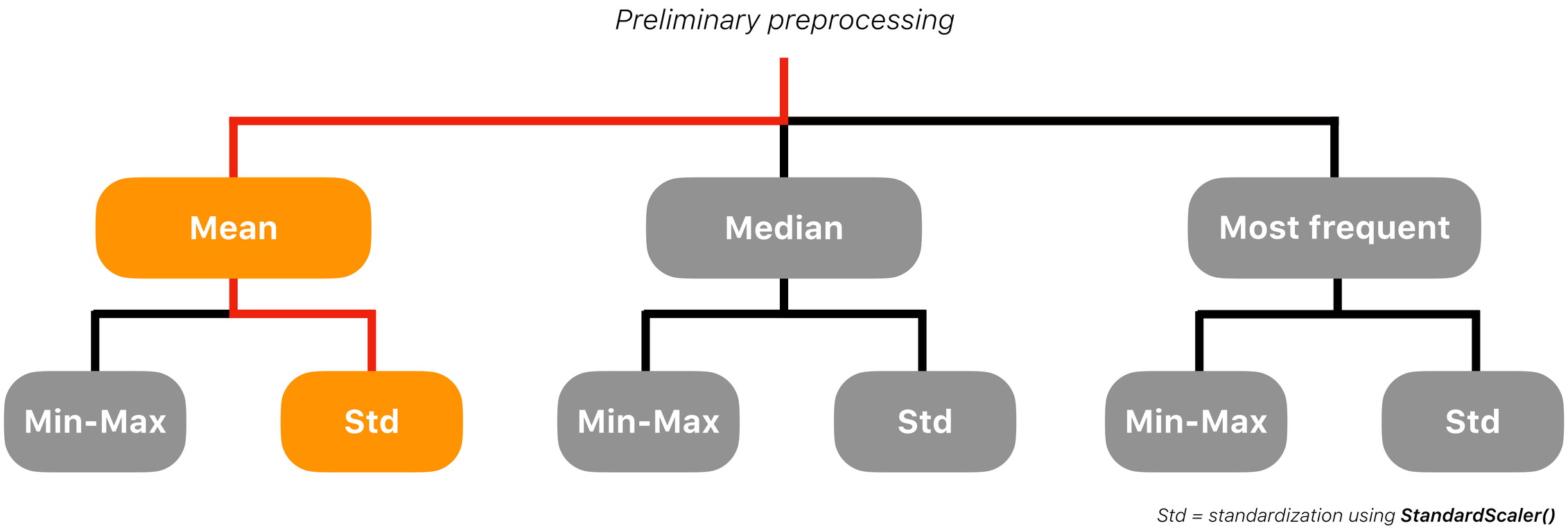
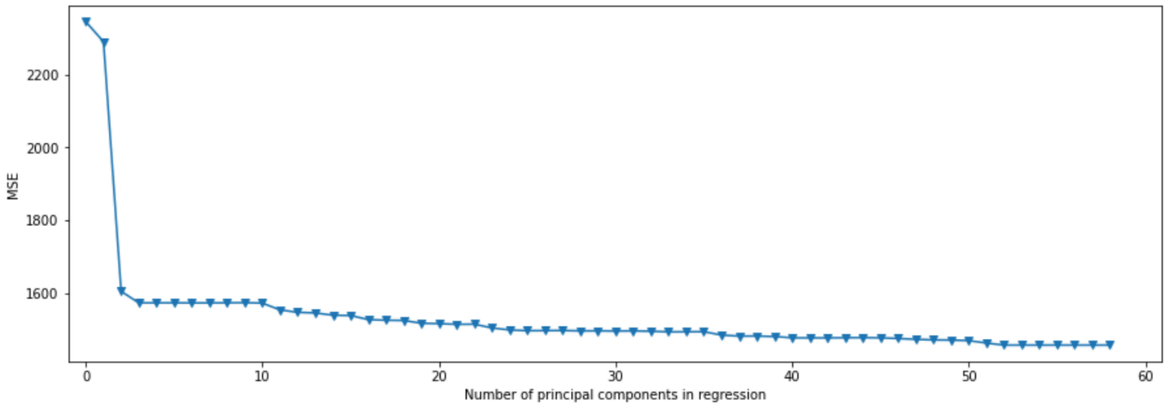


Fig. 6: Choice of imputation and scaling

4 Linear Regression | Dimensionality reduction

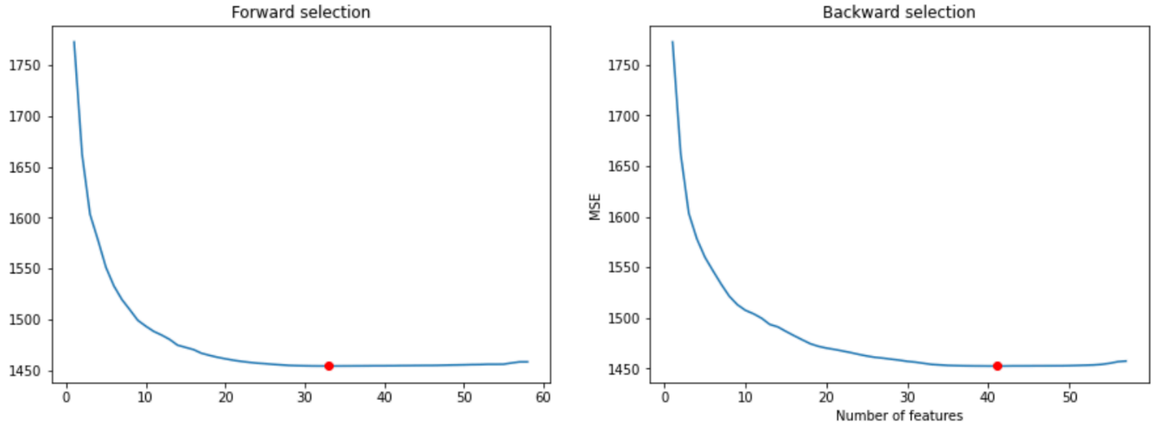
Principal Components Analysis (PCA)



Best model: 51 components

```
Model MSE on test set: 1047.1854633729333
Model RMSE on test set: 32.360245106811746
Model MAE on test set: 20.45637072243023
Model adjusted R-square on test set: 0.462178710391119
```

Feature selection



Best model: 33 features

```
Model MSE on test set: 1039.4734541692524
Model RMSE on test set: 32.24086621307269
Model MAE on test set: 20.44708903437829
Model adjusted R-square on test set: 0.469321451360749
```

4 Linear Regression | Parameter analysis & results

Fig. 7: Shapley Values for Linear Regression

The effect of each feature on the prediction is always in the same direction.
→ This is a characteristic of the linear regression model.

The features with the highest Shapley Values are also the ones that are the most correlated to the prediction.

MSE	RMSE	MAE	Adjusted-R^2
1039.5	32.2	20.4	0.47

Fig. 8: Final metrics of the model

4 Decision Tree | Brief explanation

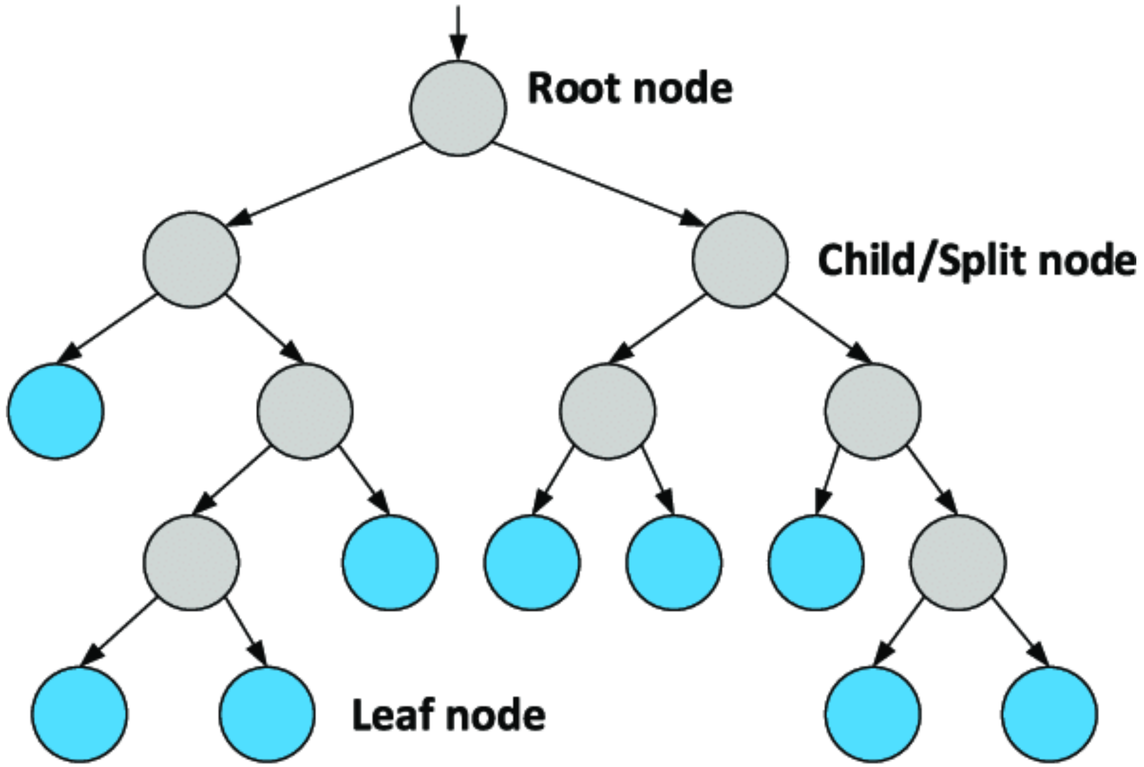


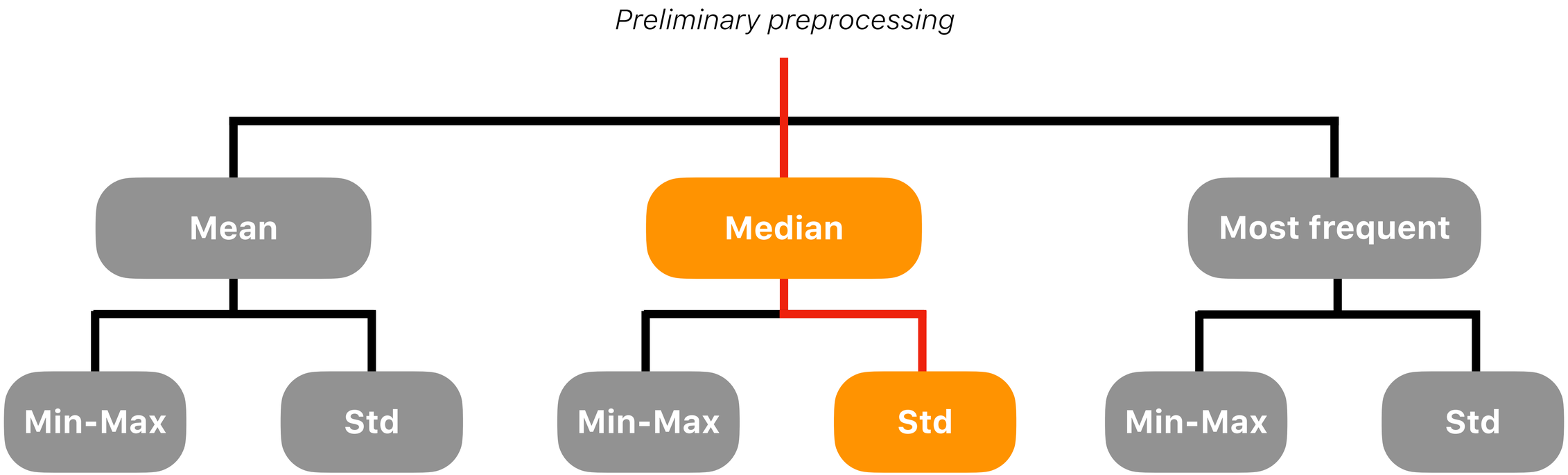
Fig. 9: Illustration of how the Decision Tree model works

- Binary tree that recursively segments the feature space.
- Creates K hyper rectangles R_k paired with a constant value γ_k.

The corresponding decision function for an instance $x \in \mathcal{X}$ is:

$$f(x) = \sum_{k=1}^{K} \gamma_k \mathbb{I}_{\{x \in R_k\}}$$

4 Decision Tree | Imputation and scaling strategy



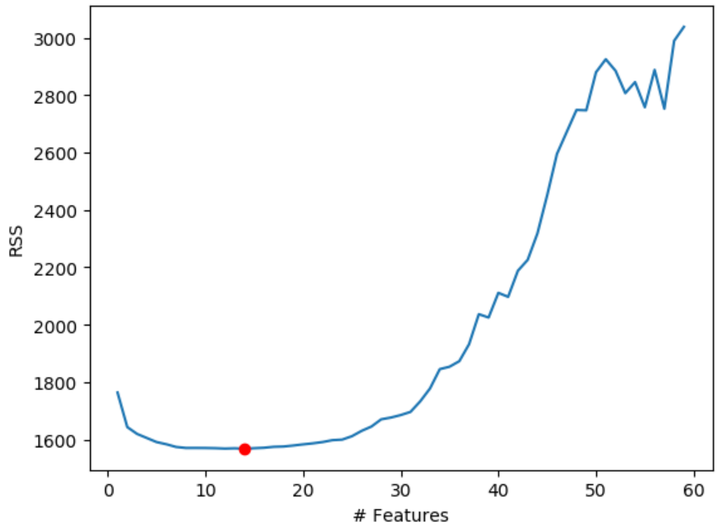
Std = standardization using ***StandardScaler()***

Fig. 10: Choice of imputation and scaling

4 Decision Tree | Dimensionality reduction

Feature selection

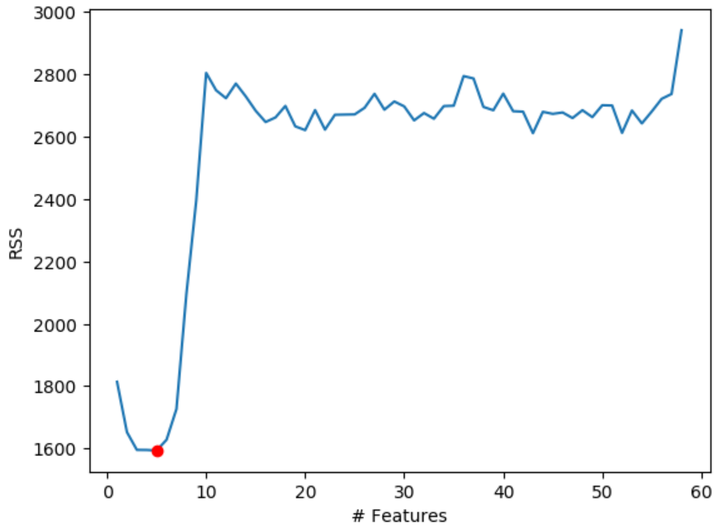
Forward selection



Best model: 14 features

`MSE: 1567.6747293899446`

Backward selection



Best model: 5 features

`MSE: 1593.6850951899457`

4 Decision Tree | Hyperparameter tuning & results

```
pgrid = {"max_depth": range(4, 35),
         "min_samples_split": range(100, 1001, 100)}
```

Best hyperparameters after the Grid Search:

- *max_depth* = 5
- *min_samples_split* = 300

MSE	RMSE	MAE	Adjusted-R^2
1373.01	37.05	21.12	0.34

Fig. 11: Final metrics of the model

4 Bayesian Ridge Regression | Brief explanation

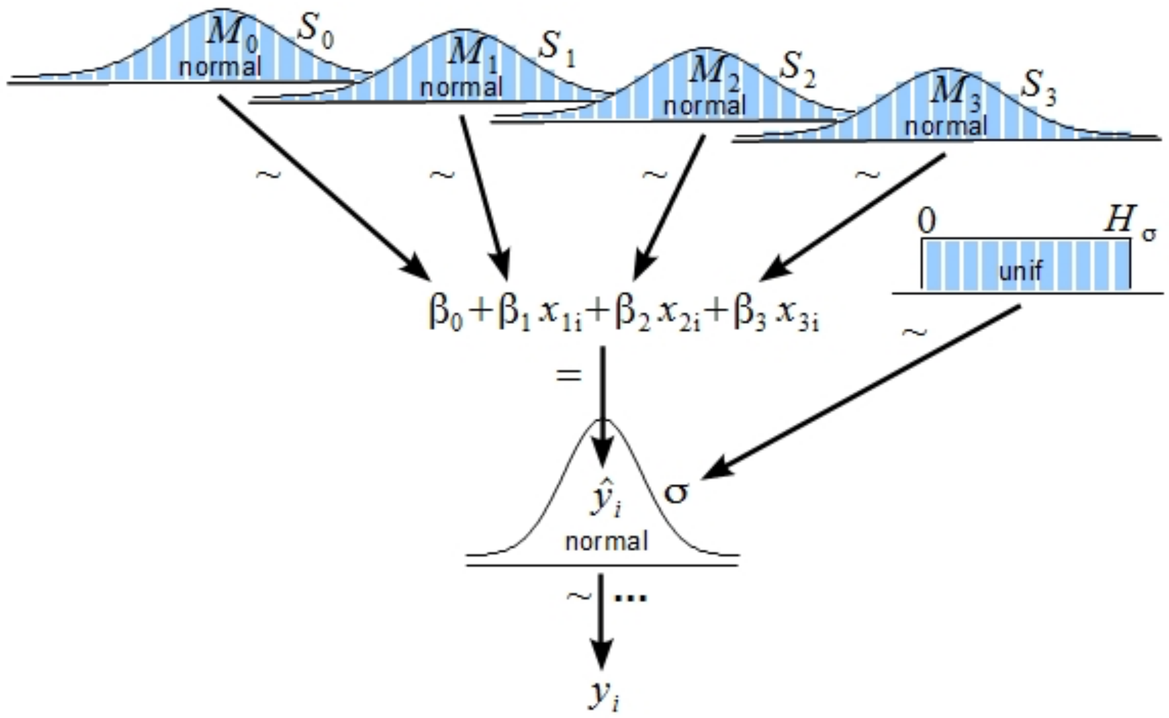


Fig. 12: Illustration of how the Bayesian Ridge Regression model works

- Use of Bayes' formalism (priors, likelihood, posterior distribution).
- The prediction is not estimated as a single value, but rather drawn from a probability distribution that needs to be determined.
- Regularization parameters tuned to the data.

4 Bayesian Ridge Regression | Imputation and scaling strategy

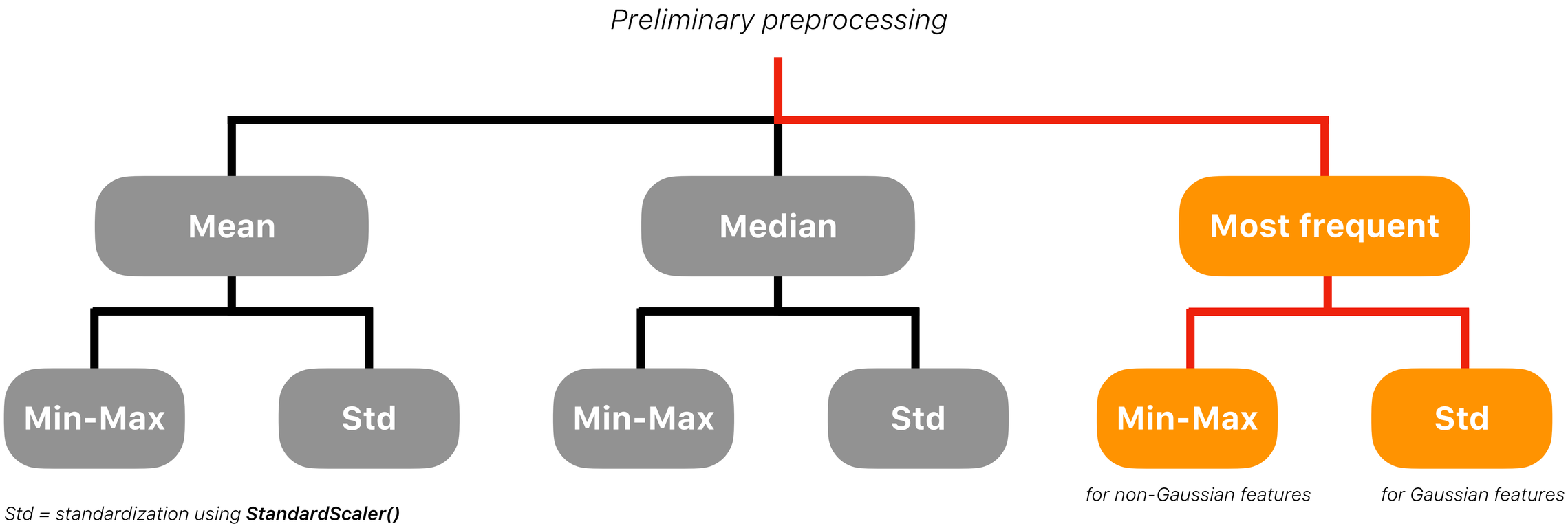
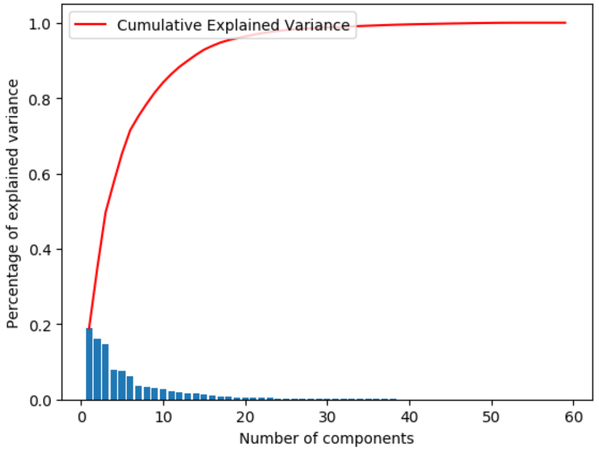
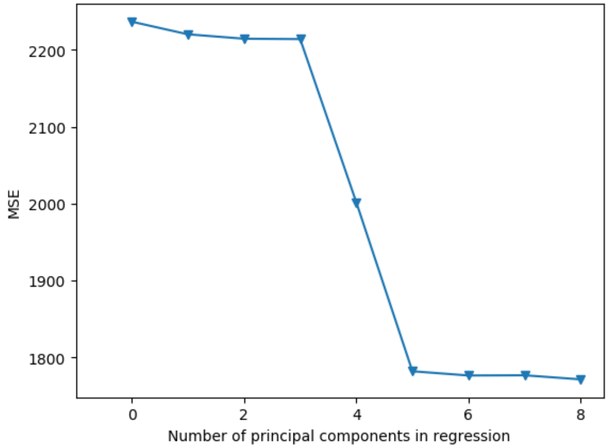


Fig. 13: Choice of imputation and scaling

4 Bayesian Ridge Regression | Dimensionality reduction

Principal Components Analysis (PCA)



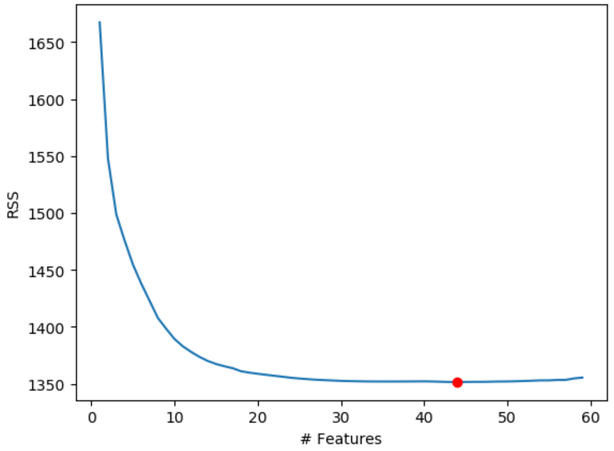


`MSE: 1771.1837392851426]`

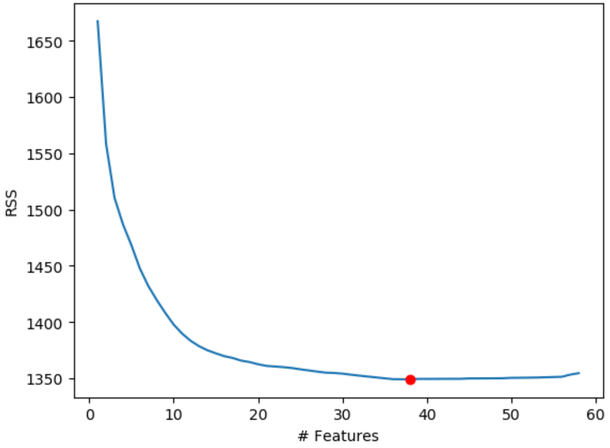
Best model: 8 components

Feature selection

Forward selection



Backward selection



44-feature model

`MSE: 1351.5318225829187`

38-feature model

`MSE: 1349.205456347221`

Best model: 38 features

4 Bayesian Ridge Regression | Hyperparameter tuning & results

```
bay_grid = {'alpha_init':[1, 1.1, 1.2, 1.3, 1.4, 1.5, 1.9],
            'lambda_init': [1e-1, 1e-2, 1e-3, 1e-4, 1e-5, 1e-6, 1e-9]}
```

Best hyperparameters after the Grid Search:

- *alpha_init* = 1.1
- *lambda_init* = 1e-06

MSE	RMSE	MAE	Adjusted-R^2
1449.62	38.07	20.33	0.38

Fig. 14: Final metrics of the model

4 K-Nearest Neighbors | Brief explanation

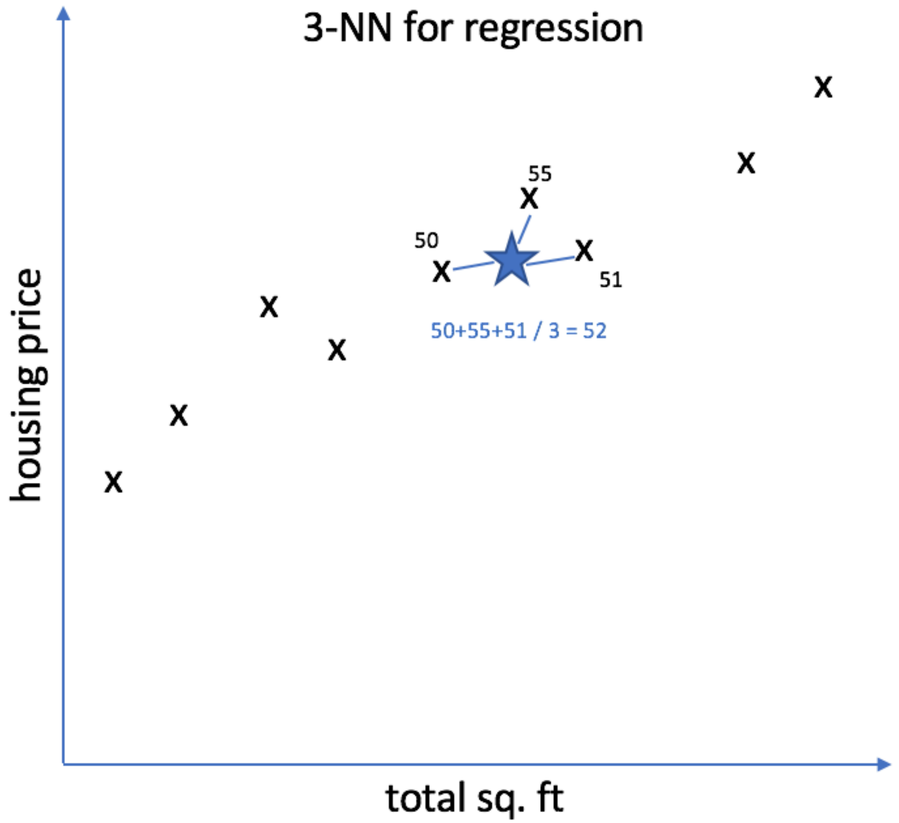


Fig. 15: Illustration of how the KNN model works

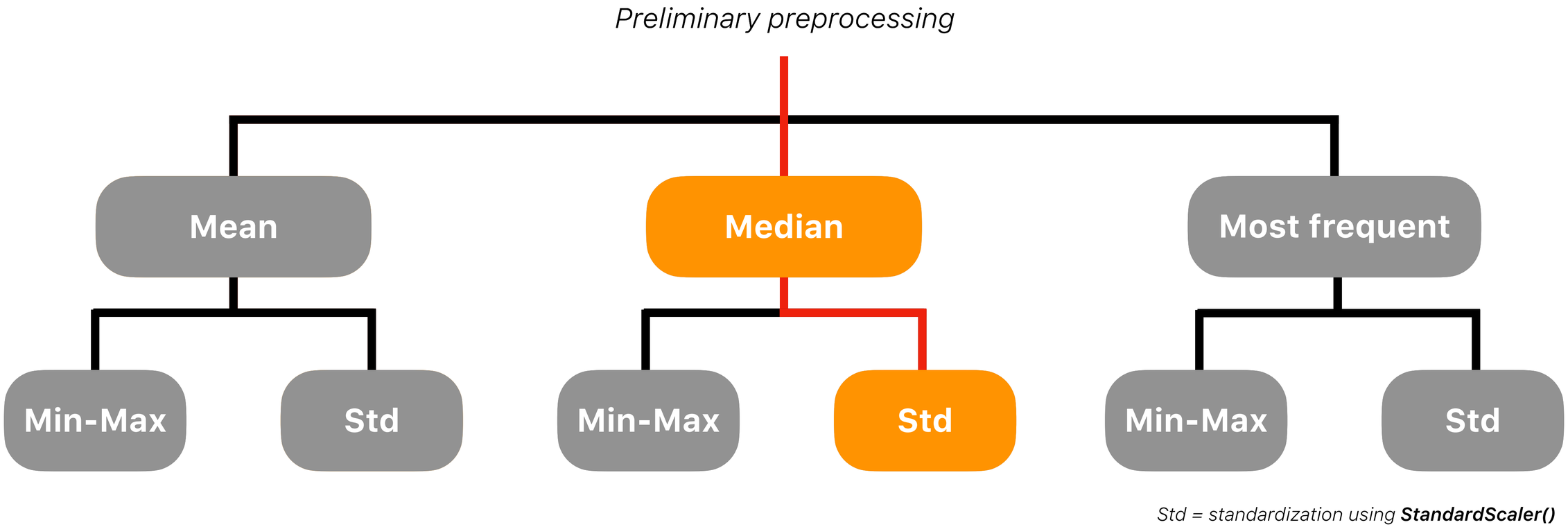


Fig. 16: Choice of imputation and scaling

4 K-Nearest Neighbors | Imputation and scaling strategy

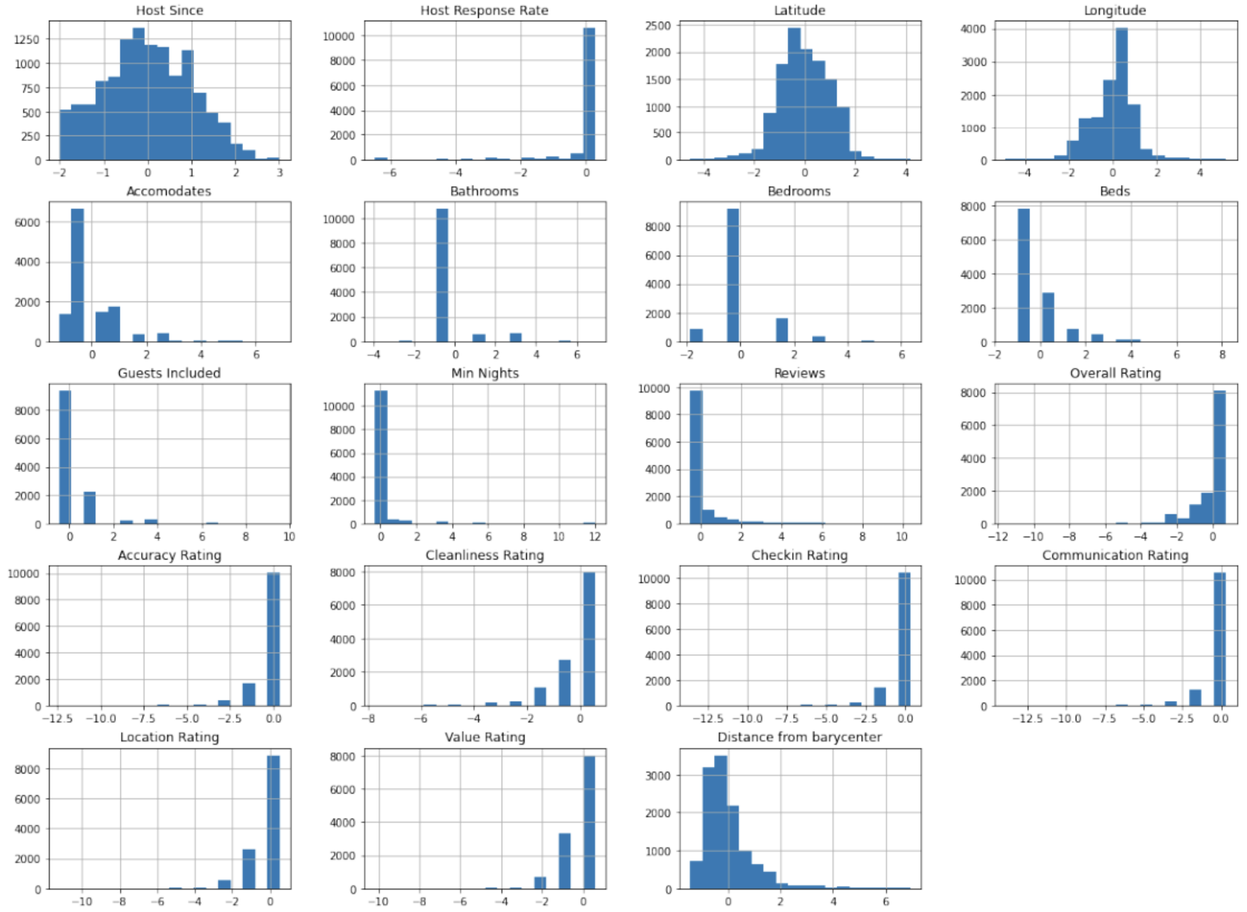
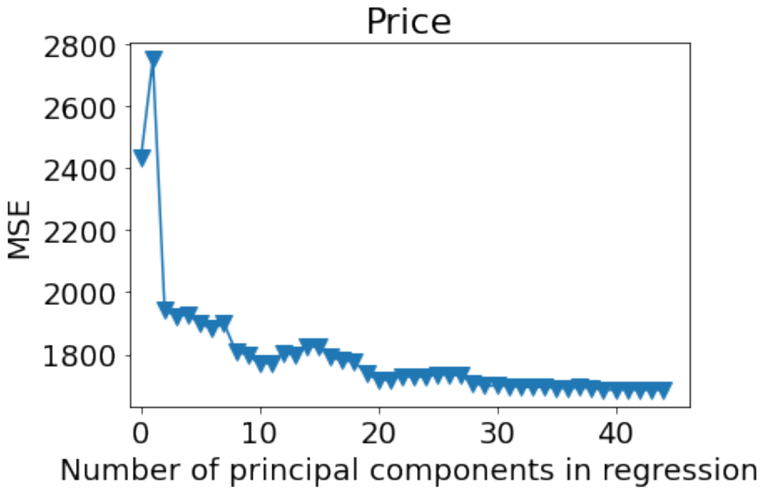


Fig. 17: Distributions of numeric features after imputing + scaling

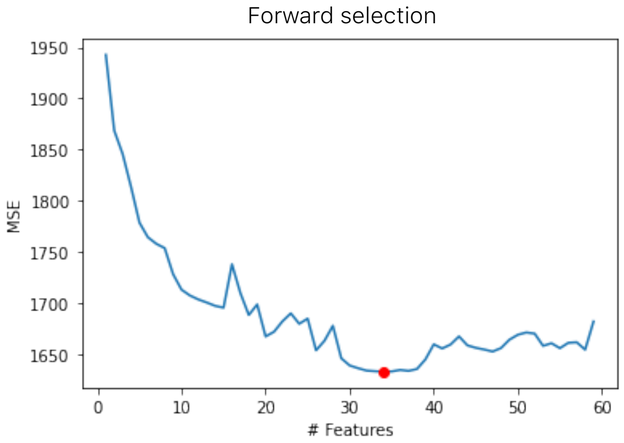
4 K-Nearest Neighbors | Dimensionality reduction

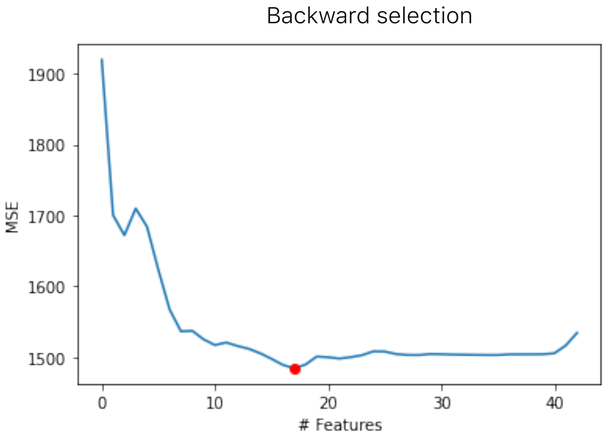
Principal Components Analysis (PCA)



Best model: 16 components

Feature selection





Best model: 17 features

4 K-Nearest Neighbors | Hyperparameter tuning & results

```
best_features_bwd_2 = ['Host Response Rate',
                       'Latitude',
                       'Longitude',
                       'Accomodates',
                       'Bathrooms',
                       'Bedrooms',
                       'Guests Included',
                       'Cleanliness Rating',
                       'Distance from barycenter',
                       'Host Response Time_unknown',
                       'Host Response Time_within an hour',
                       'Property Type_Apartment',
                       'Room Type_Entire home/apt',
                       'Room Type_Private room',
                       'Neighborhood Group_Mitte',
                       'Boy or Girl?_Boy',
                       'Boy or Girl?_Girl']
```

```
from sklearn.model_selection import GridSearchCV

pgrid = {"n_neighbors":[20,30,40,50],
         "weights":['uniform','distance'],
         "leaf_size":[1,10,20]}

grid_search = GridSearchCV(KNeighborsRegressor(),param_grid=pgrid,cv=10,verbose=11)
grid_search.fit(X_train_processed_2[best_features_bwd_2],y_train_2)
```

Fig. 18: Selected features and Grid Search definition

Best hyperparameters after the Grid Search:

- *n_neighbors* = 49
- *weights* = "distance"
- *leaf_size* = 1

MSE	RMSE	MAE	Adjusted-R^2
1074.76	33.32	19.94	0.45

Fig. 19: Final metrics of the model

4 Random Forest | Brief explanation

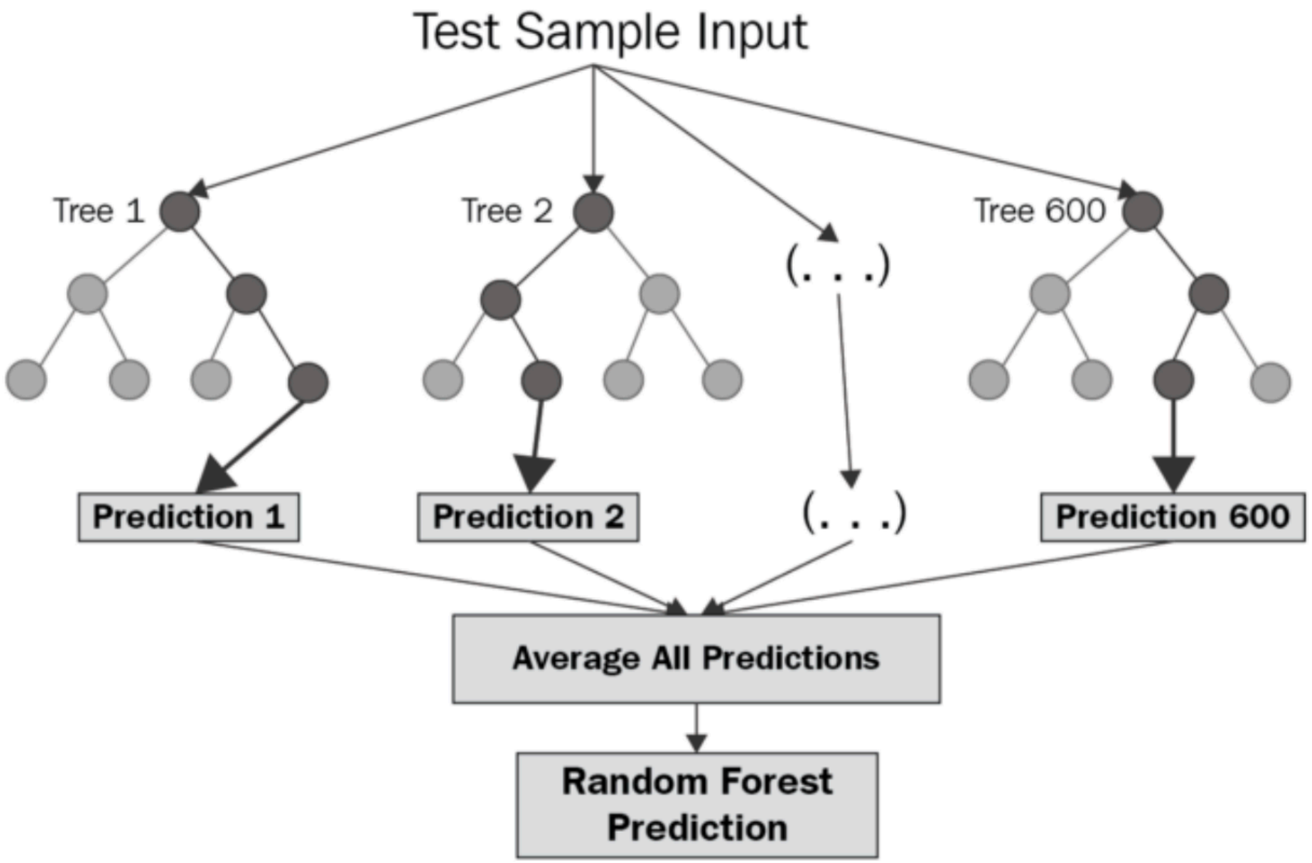
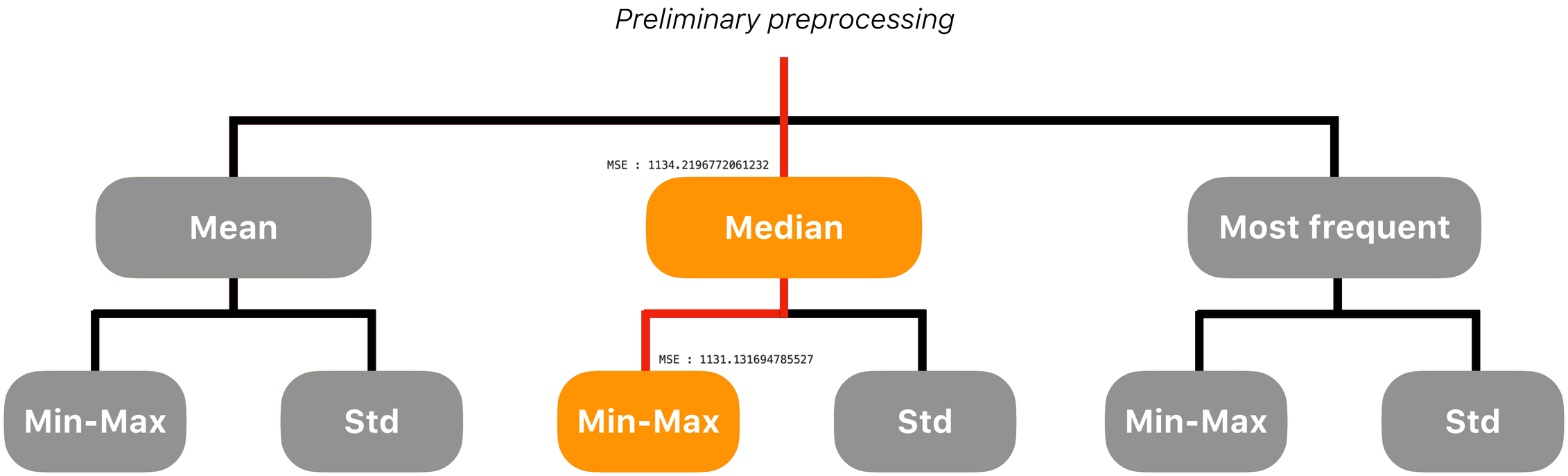


Fig. 20: Illustration of how the Random Forest model works

For each tree:

- Random selection of subset features
- Random subset of instances

Random Forest | Imputation and scaling strategy



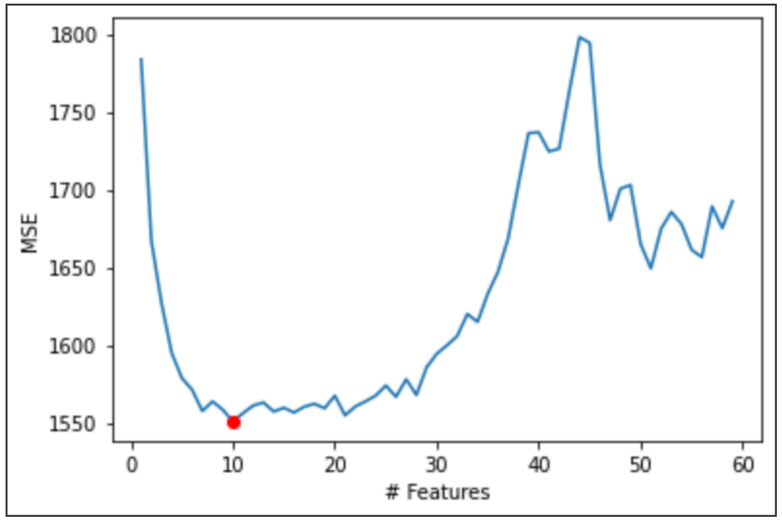
Std = standardization using ***StandardScaler()***

Fig. 21: Choice of imputation and scaling

4 Random Forest | Dimensionality reduction

Feature selection

Forward selection



Best model: 10 features

```
MSE : 1084.2303287566526
MAE : 20.550931479569435
RMSE: 32.92765294940793
R2 Score: 0.45245499690533253
```

```
Variable: Accomodates           Importance: 0.11
Variable: Bedrooms              Importance: 0.1
Variable: Room Type_Entire home/apt Importance: 0.08
Variable: Room Type_Private room Importance: 0.08
Variable: Beds                  Importance: 0.07
Variable: Bathrooms             Importance: 0.06
Variable: Distance from barycenter Importance: 0.06
Variable: Latitude              Importance: 0.05
Variable: Guests Included       Importance: 0.05
Variable: Host Since            Importance: 0.04
Variable: Longitude             Importance: 0.04
Variable: Reviews               Importance: 0.03
Variable: Min Nights            Importance: 0.02
Variable: Overall Rating        Importance: 0.02
Variable: Property Type_Loft    Importance: 0.02
Variable: Host Response Rate    Importance: 0.01
Variable: Cleanliness Rating    Importance: 0.01
Variable: Location Rating       Importance: 0.01
Variable: Value Rating          Importance: 0.01
Variable: Host Response Time_unknown Importance: 0.01
Variable: Host Response Time_within an hour Importance: 0.01
Variable: Property Type_Apartment Importance: 0.01
Variable: Property Type_Hotel   Importance: 0.01
Variable: Property Type_Serviced apartment Importance: 0.01
Variable: Neighborhood Group_Mitte Importance: 0.01
Variable: Boy or Girl?_Boy      Importance: 0.01
Variable: Boy or Girl?_Girl     Importance: 0.01
Variable: Is Superhost          Importance: 0.0
Variable: Is Exact Location     Importance: 0.0
Variable: Accuracy Rating       Importance: 0.0
Variable: Checkin Rating        Importance: 0.0
Variable: Communication Rating  Importance: 0.0
Variable: Instant Bookable      Importance: 0.0
```

MSE : 1015.8466948031265

MAE : 19.67591384347838

RMSE: 31.87235000440235

R2 Score: 0.48698927995536134

4 Random Forest | Hyperparameter tuning & results

```
param = {'n_estimators':[200,300,400],
         'max_depth':[5,10,20,30],
         'max_features': ['auto', 'sqrt'],
         'min_samples_leaf': [1, 3],
         'min_samples_split': [2, 7],
}
```

Best hyperparameters after the Random Search:

- *n_estimators* = 300
- *min_samples_split* = 7
- *max_features* = 'sqrt'
- *max_depth* = 30

MSE	RMSE	MAE	Adjusted-R^2
1003.91	31.68	19.64	0.48

Fig. 22: Final metrics of the model

Fig. 23: Visualization of a Decision Tree from the Random Forest

4 Random Forest | Additional observations

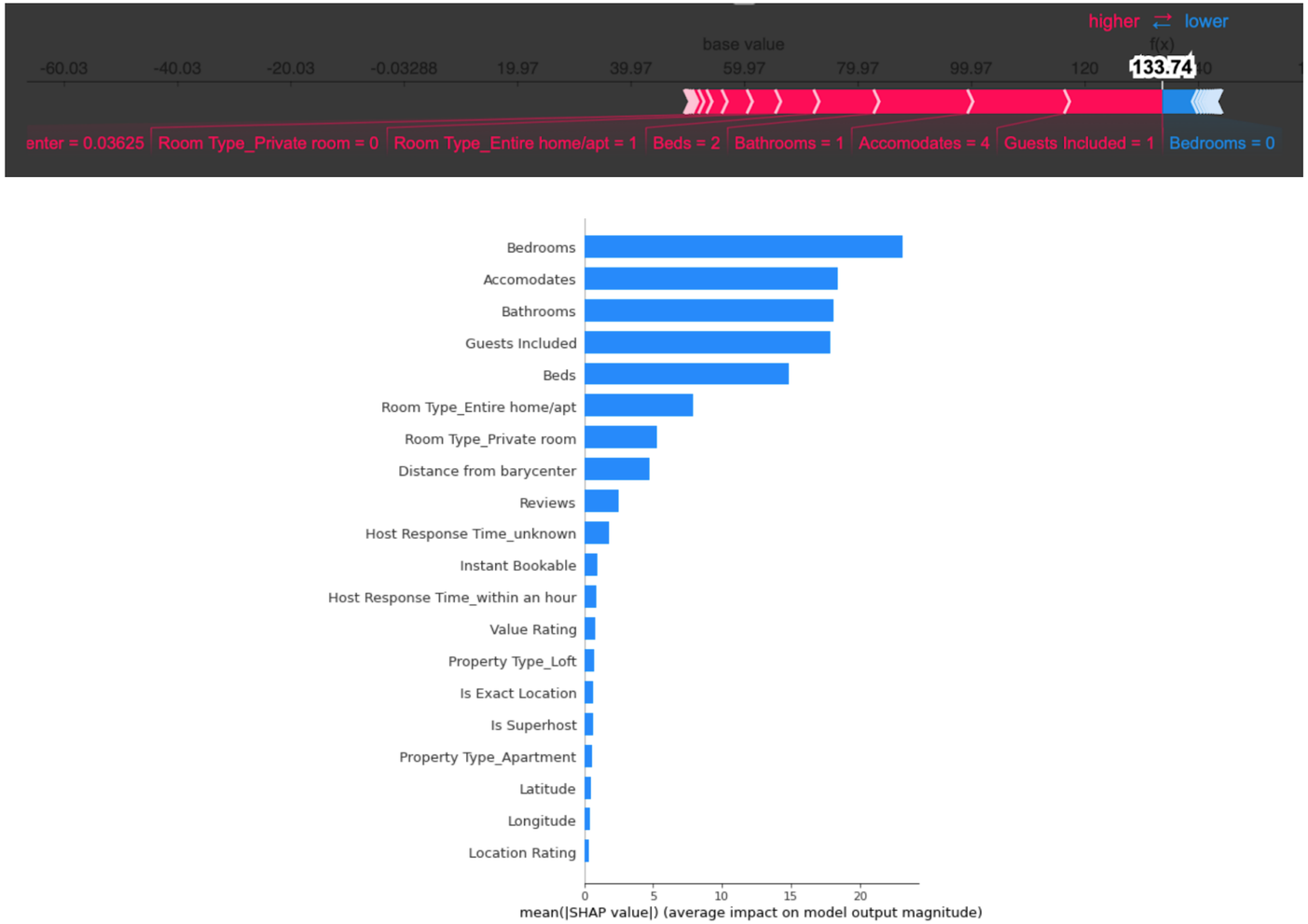


Fig. 24: Shapley Values for Random Forest

4 Support Vector Regression | Brief explanation

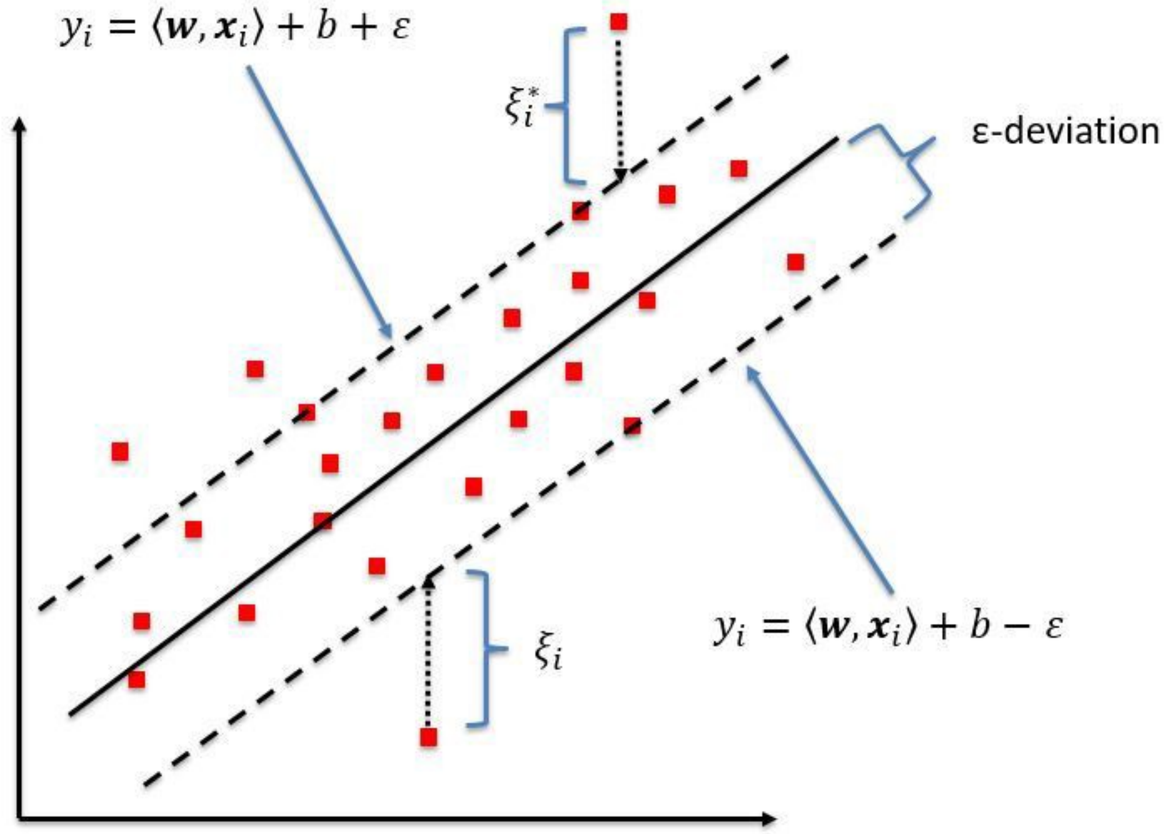
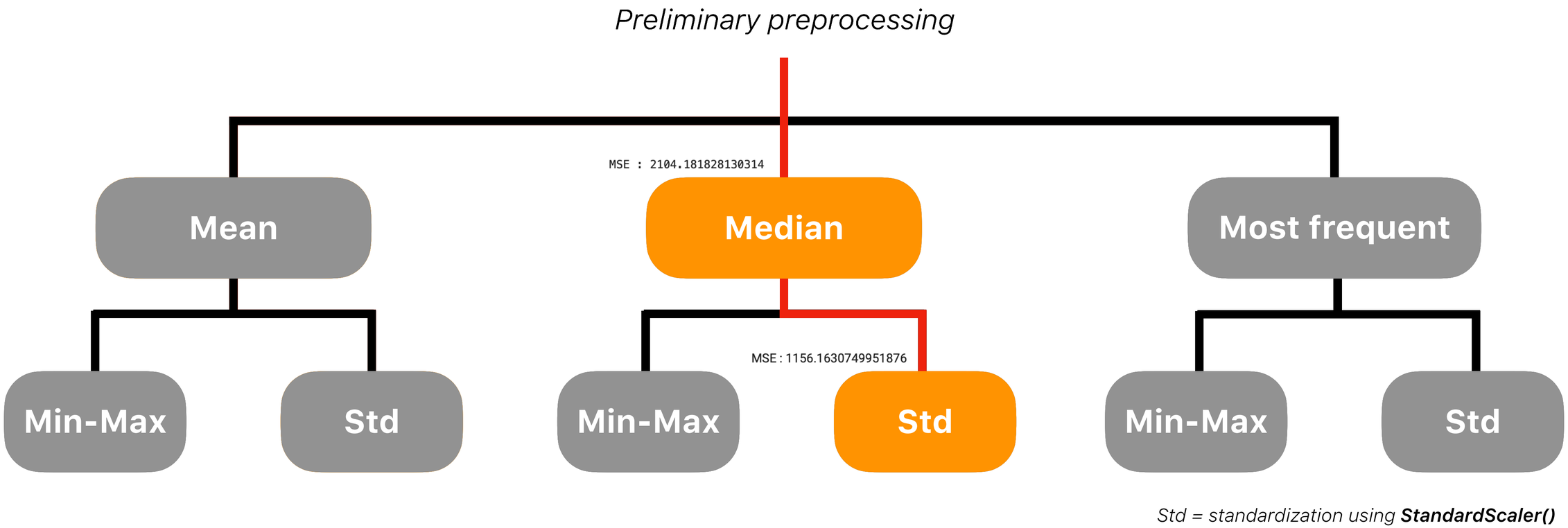


Fig. 25: Illustration of how the SVR model works

4 Support Vector Regression | Imputation and scaling strategy

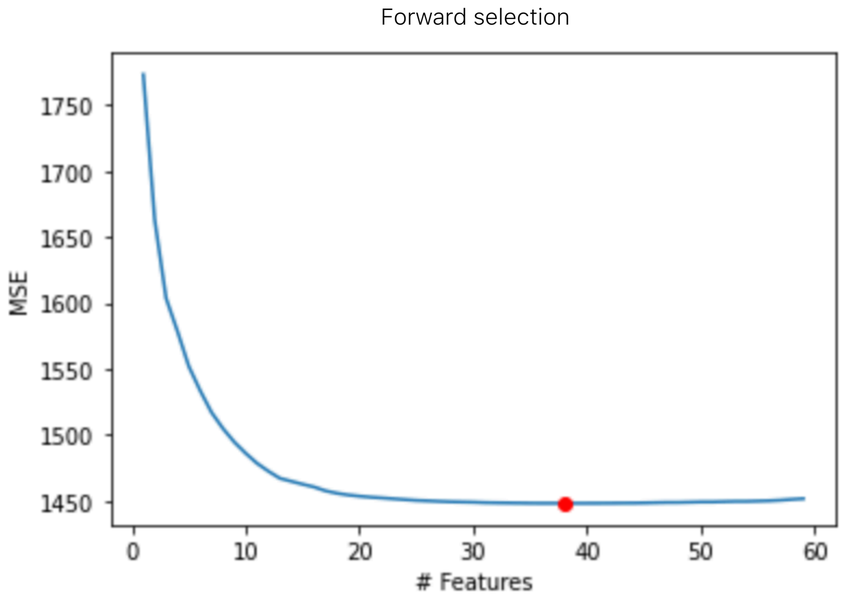


Std = standardization using ***StandardScaler()***

Fig. 26: Choice of imputation and scaling

4 Support Vector Regression | Dimensionality reduction

Feature selection



Best model: 38 features

4 Support Vector Regression | Hyperparameter tuning & results

```
param_grid = {'C': [0.1, 1, 10, 100, 1000],
              'loss': ['epsilon_insensitive', 'squared_epsilon_insensitive'],
              'dual': [False],
              'tol': [0.0001, 0.00001]}
```

Best hyperparameters after the Grid Search:

- *C* = 0.1
- *loss* = 'squared_epsilon_insensitive'
- *dual* = False
- *tol* = 0.0001

MSE	RMSE	MAE	Adjusted-R^2
1033.62	32.15	20.31	0.48

Fig. 27: Final metrics of the model

4 Support Vector Regression | Additional observations

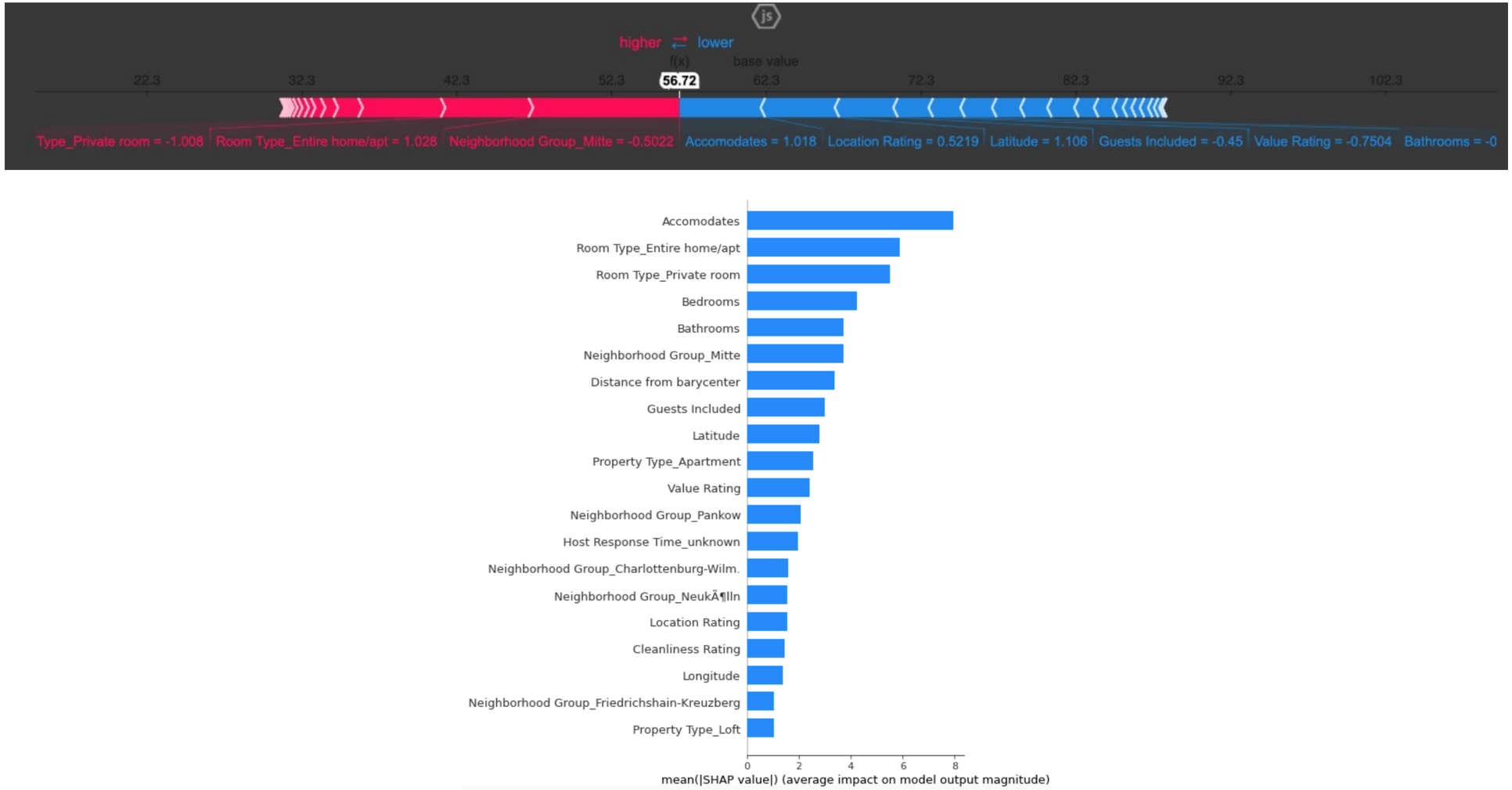


Fig. 28: Shapley Values for SVR

4 AdaBoost | Brief explanation

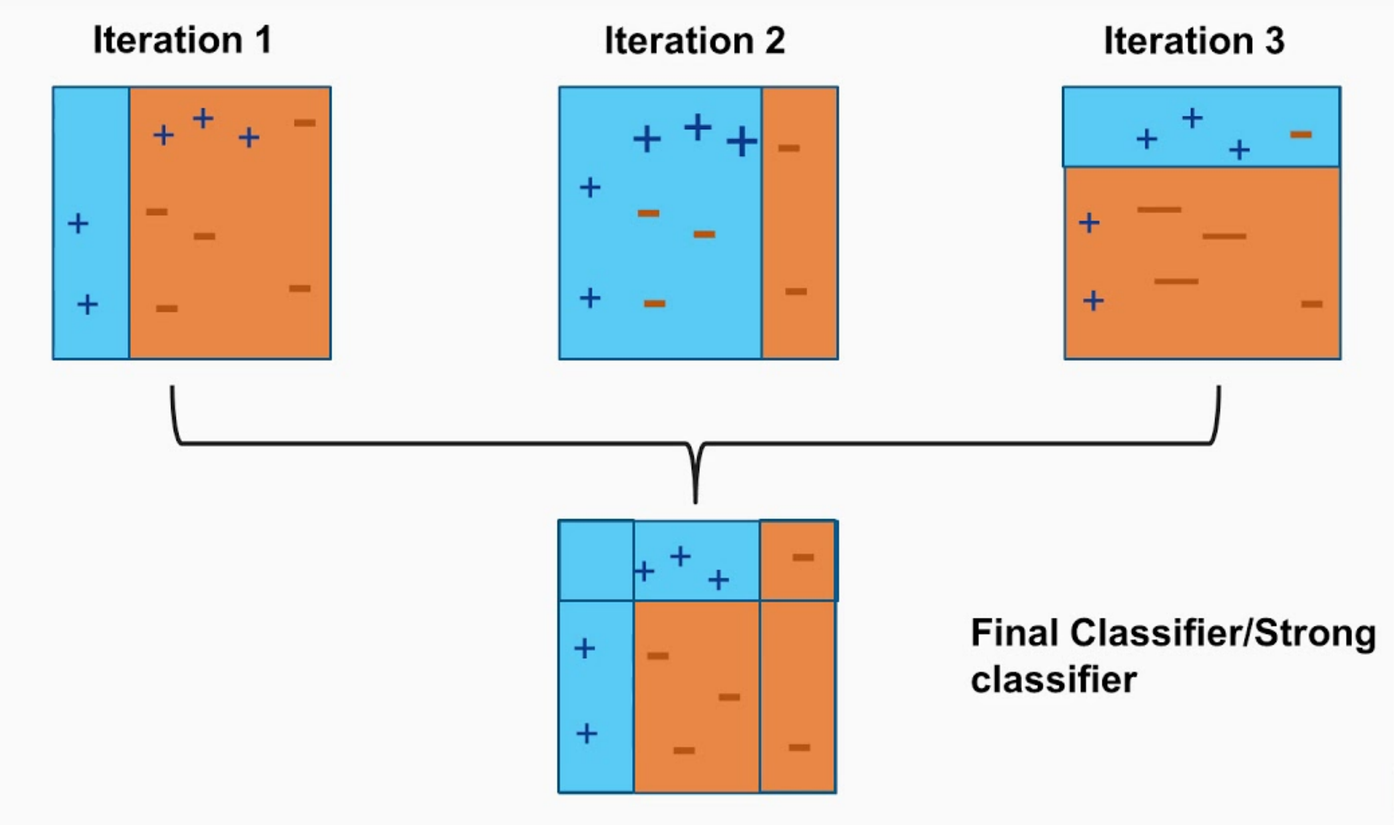


Fig. 29: Illustration of how the AdaBoost model works

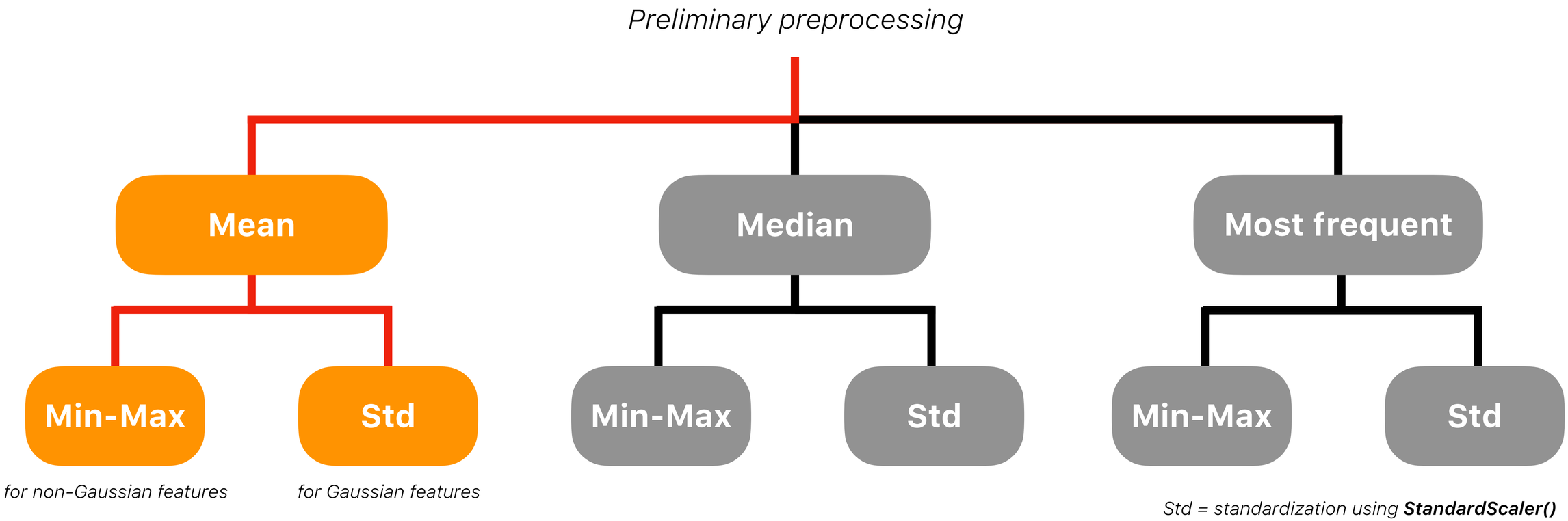
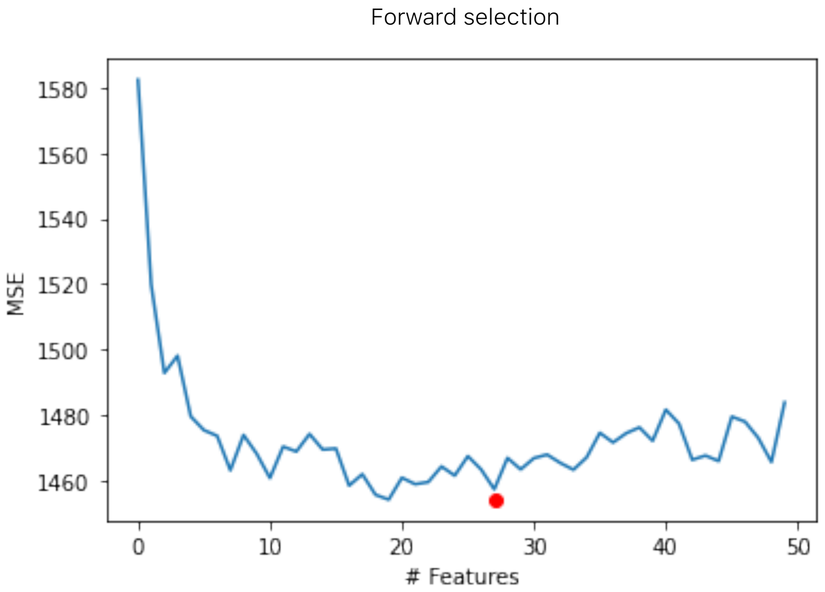


Fig. 30: Choice of imputation and scaling

```
num_pipeline_mn_minmax = Pipeline([
    ('missing_mean', SimpleImputer(strategy="mean")),
    ('minmax_scaler', MinMaxScaler())
])

num_pipeline_mdn_minmax = Pipeline([
    ('missing_median', SimpleImputer(strategy="median")),
    ('minmax_scaler', MinMaxScaler())
])

num_pipeline_mf_minmax = Pipeline([
    ('missing_mf', SimpleImputer(strategy="most_frequent")),
    ('minmax_scaler', MinMaxScaler())
])

num_pipeline_mn_std = Pipeline([
    ('missing_mean', SimpleImputer(strategy="mean")),
    ('std_scaler', StandardScaler())
])

num_pipeline_mdn_std = Pipeline([
    ('missing_median', SimpleImputer(strategy="median")),
    ('std_scaler', StandardScaler())
])

num_pipeline_mf_std = Pipeline([
    ('missing_mf', SimpleImputer(strategy="most_frequent")),
    ('std_scaler', StandardScaler())
])
```

Fig. 31: Pipelines used for testing imputation + scaling strategies

Feature selection



Best model: 27 features

4 AdaBoost | Hyperparameter tuning & results

```
from sklearn.tree import DecisionTreeRegressor

adaboost = AdaBoostRegressor(base_estimator=DecisionTreeRegressor(),random_state=42)

pgrid_initial_5 = {'base_estimator__max_depth':[1,5,10,15,20],
              'base_estimator__min_samples_leaf':[1,3,5,10],
              'n_estimators':[50,100,150,200,250,300],
              'learning_rate':[0.001,0.01,0.1,1]}

grid_search_raw_5 = GridSearchCV(adaboost,param_grid=pgrid_initial_5,cv=5, verbose=11)
grid_search_raw_5.fit(X_train_processed[eval(best_feature_bwd_1)],y_train)
```

Best hyperparameters after the Grid Search:

- *max_depth* = 10
- *min_samples_leaf* = 10
- n_*estimators* = 200
- *learning_rate* = 0.001

MSE	RMSE	MAE	Adjusted-R^2
1085.56	32.95	19.39	0.45

Fig. 32: Final metrics of the model

Idem AdaBoost.

(But quicker.)

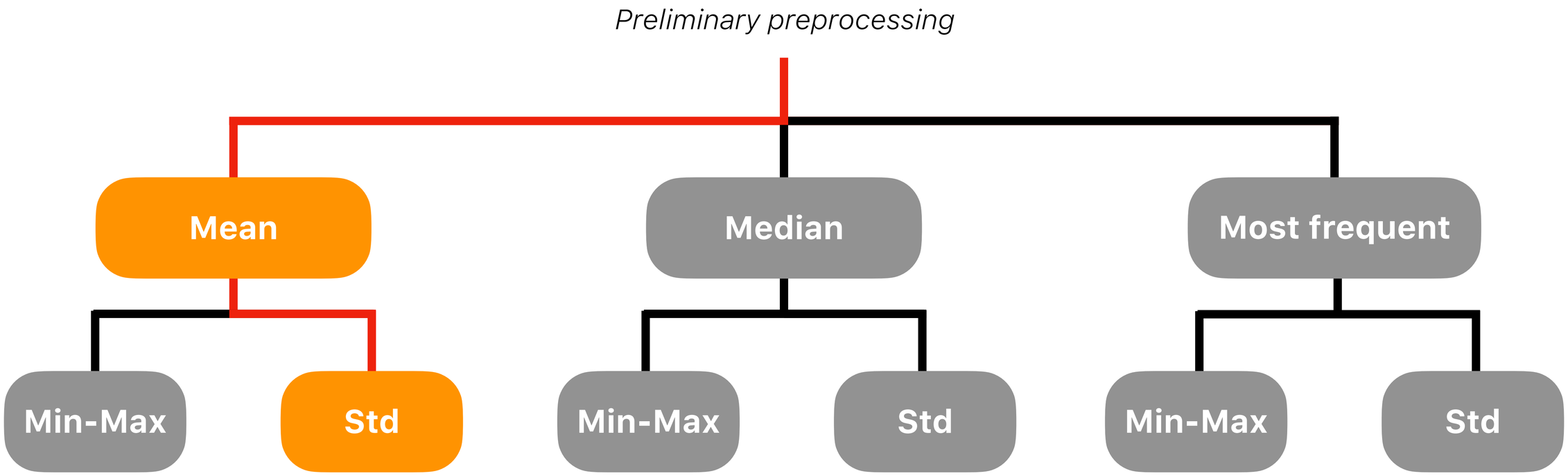
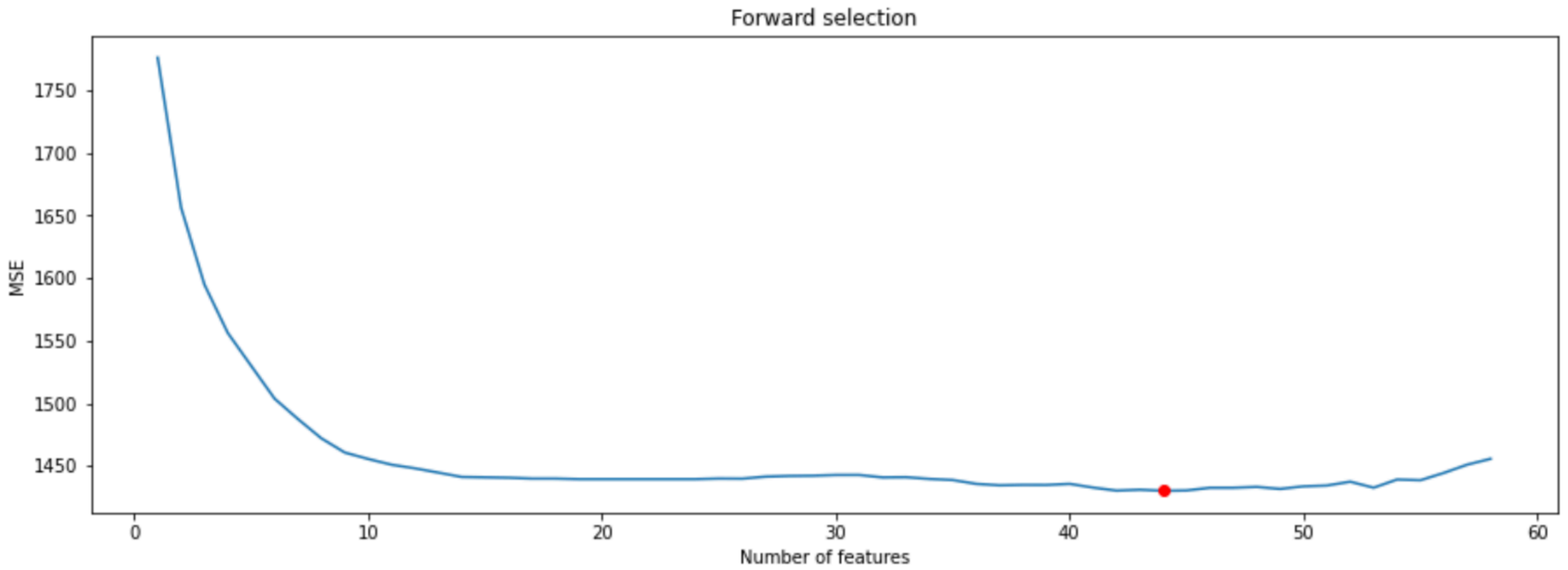


Fig. 33: Choice of imputation and scaling

Feature selection



Forward selection

44-feature model

```
Model MSE on test set: 1021.8354400132727
Model RMSE on test set: 31.966160858214938
Model MAE on test set: 19.654199820058206
Model adjusted R-square on test set: 0.476419038701424
```

Backward selection

15-feature model

```
Model MSE on test set: 1023.525696457521
Model RMSE on test set: 31.99258814878098
Model MAE on test set: 19.657280744566354
Model adjusted R-square on test set: 0.48055920601712554
```

4 XGBoost | Hyperparameter tuning & results

```
pgrid = {
    'n_estimators': [45, 46, 47],
    'eta': [0.1]
}
```

1. Look for the best *n_estimators* at fixed learning rate *eta*

```
pgrid = {
    'n_estimators': [46],
    'eta': [0.1],
    'max_depth':[4],
    'min_child_weight':[2],
    'gamma':[i/10.0 for i in range(0,5)]
}
```

3. Given all our parameters, search for best *gamma*

Best hyperparameters after the Grid Search:

- *eta* = 0.1
- *min_child_weight* = 2
- n_*estimators* = 46
- *max_depth* = 4
- *gamma* = 0

```
pgrid = {
    'n_estimators': [46],
    'eta': [0.1],
    'max_depth':[2, 3, 4, 5],
    'min_child_weight':[1, 2, 3, 4]
}
```

2. Given the best *eta* and *n_estimators*, look for the best *max_depth* and *min_child_weight* combination

MSE	RMSE	MAE	Adjusted-R^2
1010.2	31.8	19.6	0.48

Fig. 34: Final metrics of the model

4 XGBoost | Additional observations

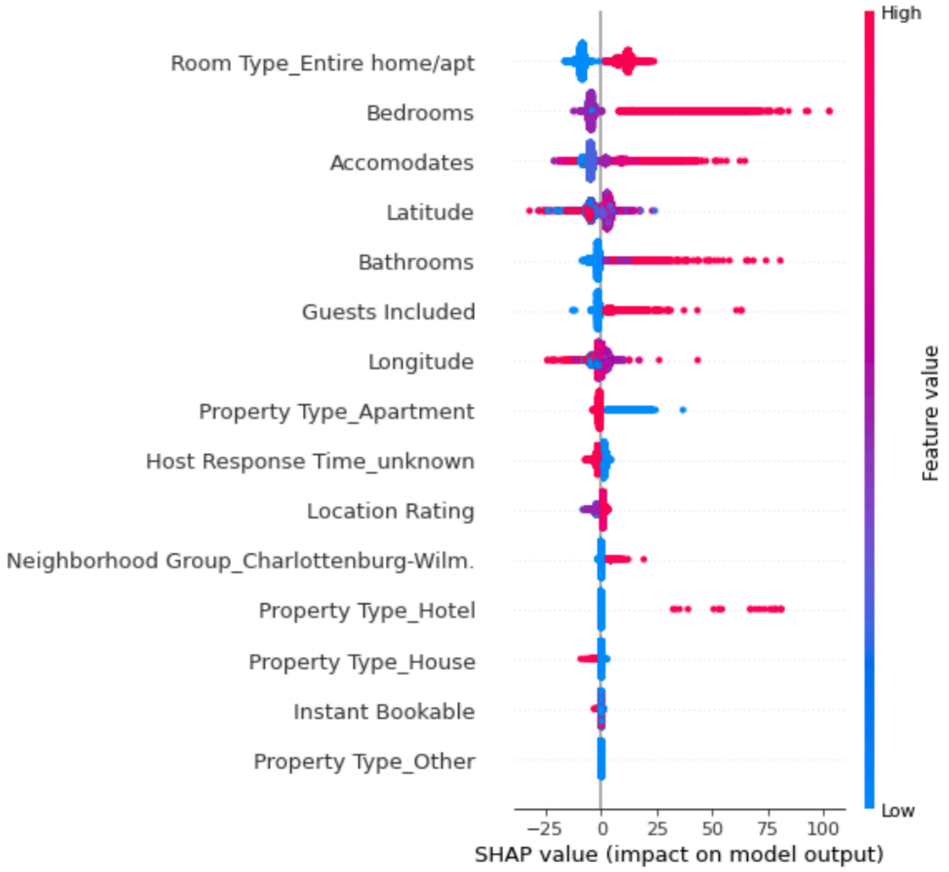


Fig. 35: Shapley Values for XGBoost

- Contrary to the linear regression for example, XGBoost allows for non-linear relationships, which means the impact of a feature on the prediction depends on the instance.
- Most features still have a “preferred impact” on the prediction.
- We find the same main features driving the result as for other models.

5 Conclusion | Recap of the results

- Random Forest and XGBoost yielded the best results in terms of MSE and Adjusted-R^2.
- Best result for MAE given by AdaBoost.
- However, we are still far from a perfect prediction: we manage to get the absolute error below 20 euros on average but there's still room for improvement.

Model	MSE	RMSE	MAE	Adjusted-R^2
Linear Regression	1039.5	32.2	20.4	0.47
SVR	1033.62	32.14	20.31	0.47
Bayesian Ridge	1449.62	38.07	20.33	0.38
KNN	1074.76	33.32	19.94	0.45
Decision Tree	1373.01	37.05	21.12	0.34
Random Forest	1003.91	31.68	19.64	0.48
AdaBoost	1085.56	32.95	19.39	0.45
XGBoost	1010.2	31.8	19.6	0.48

Fig. 36: Performance of all models as given by MSE, RMSE, MAE and Adjusted-R^2 metrics

5 Conclusion | Final observations

Our suppositions as to why we can’t reduce the prediction error further:

- The prices don’t present a large enough variance for an estimator to have an accurate prediction.
- The features themselves aren’t descriptive enough of the host’s behavior in the decision of price. Deeper information on his/her background (financial, social, professional) could give better insight into the way the price is set.

Thank you!

Seong Woo AHN • Paul BÉRARD • Leila BERRADA • Tanguy BLEVARCQUE