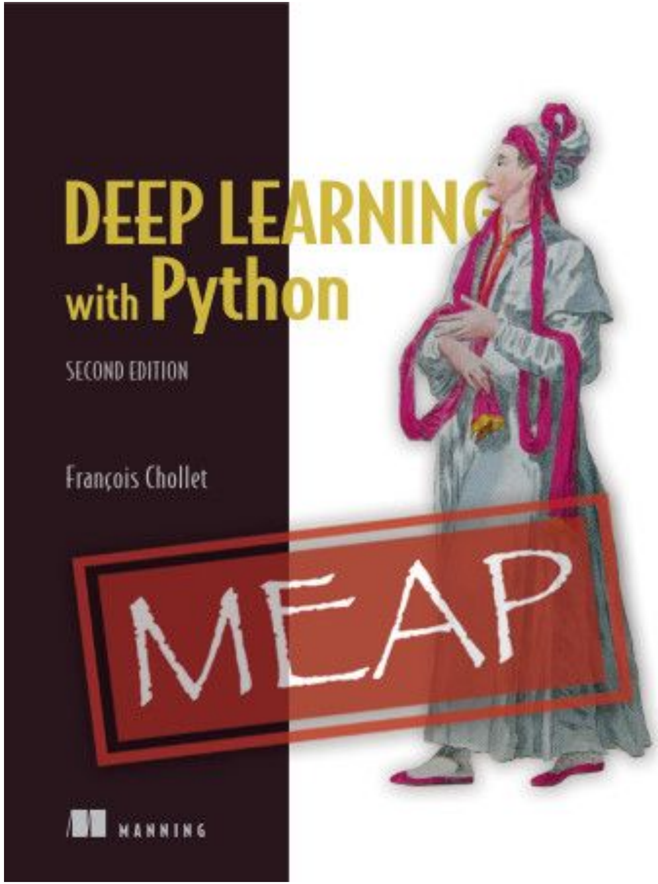
UFRN
UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

K-Nearest Neighbors

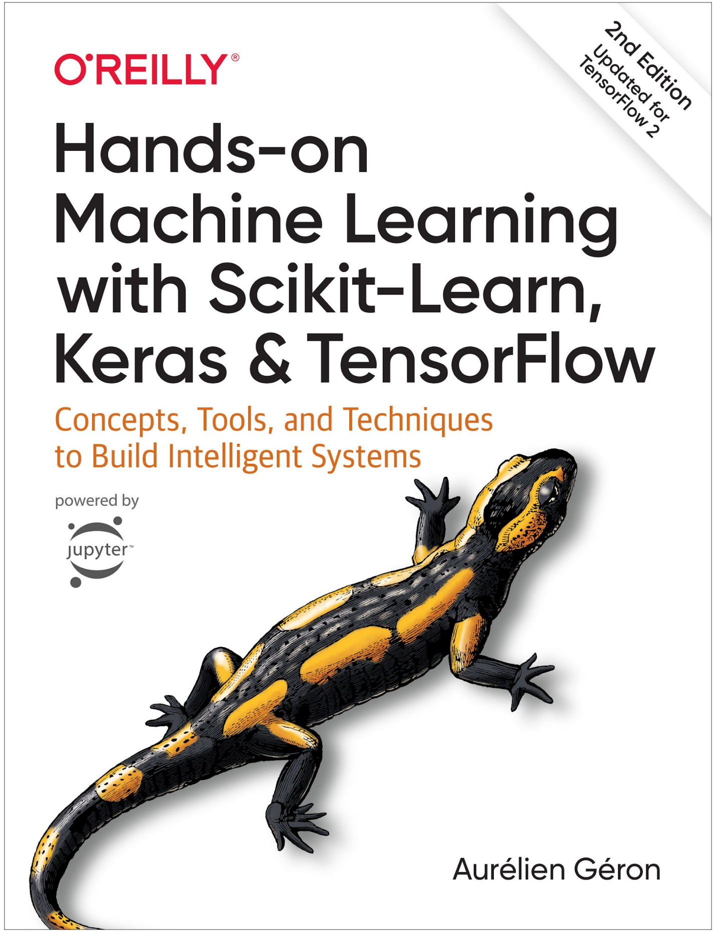
Lesson #03

Table of Contents

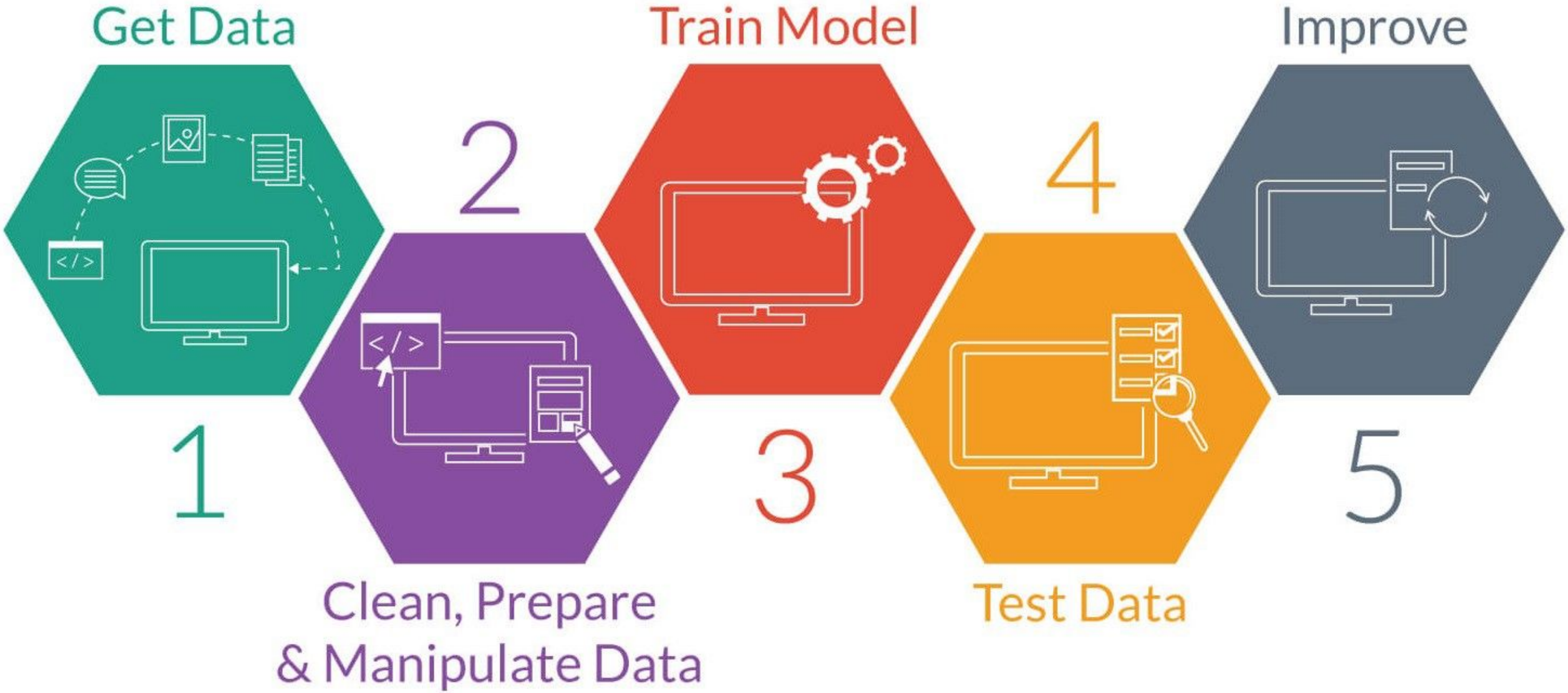
- Define the task
  - Frame the problem
  - Collect a dataset
  - Understand your data
  - Choose a measure of success
- Develop a model
  - Prepare the data
  - Choose an evaluation protocol
  - Beat a baseline
  - Scale up: develop a model that overfits
  - Regularize and tune your model
- Deploy your model
  - Explain your work to stakeholders and set expectations
    - Ship an inference model
    - Deploying a model as a rest API
    - Deploying a model on device
    - Deploying a model in the browser
  - Monitor your model in the wild
  - Maintain your model

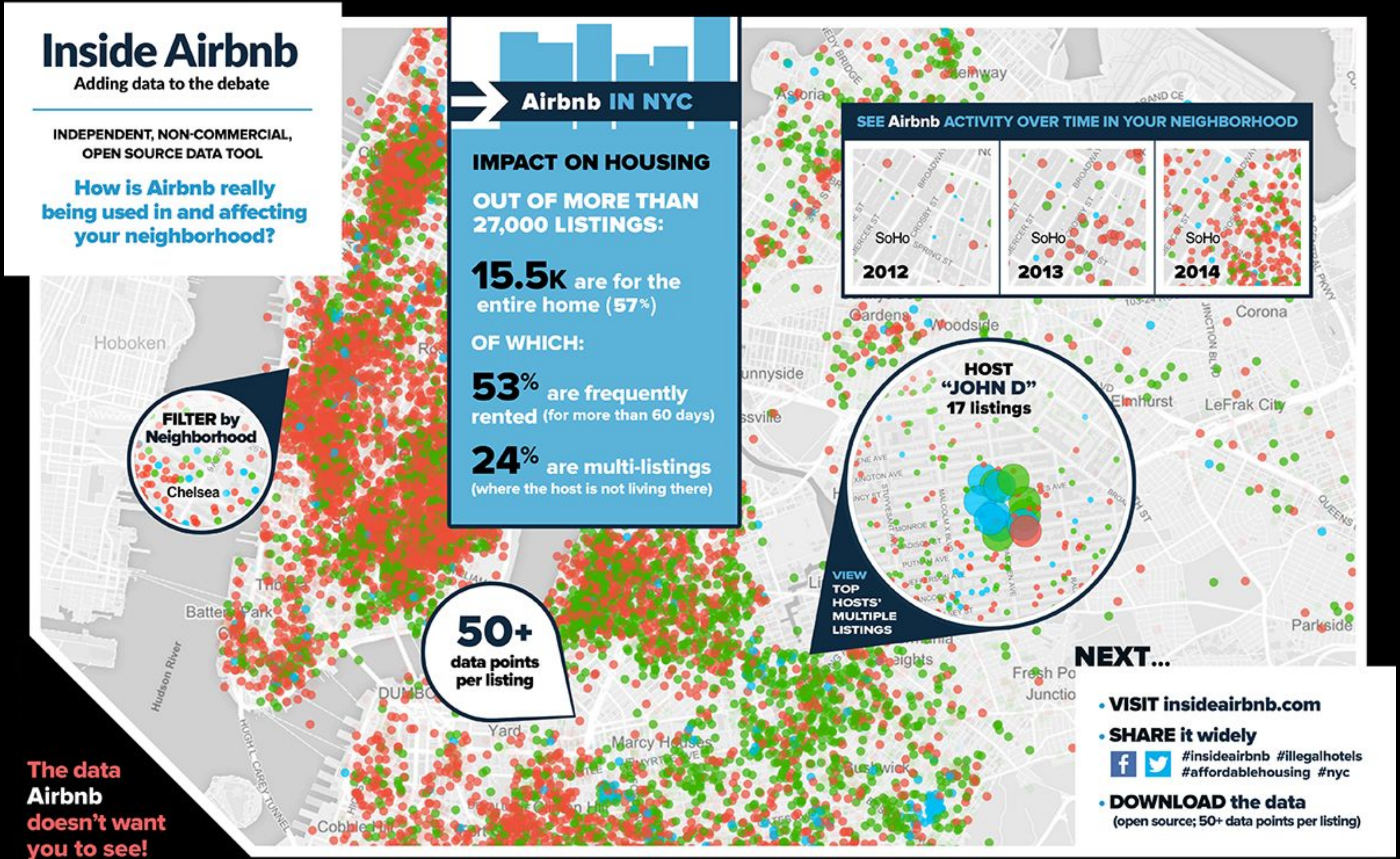


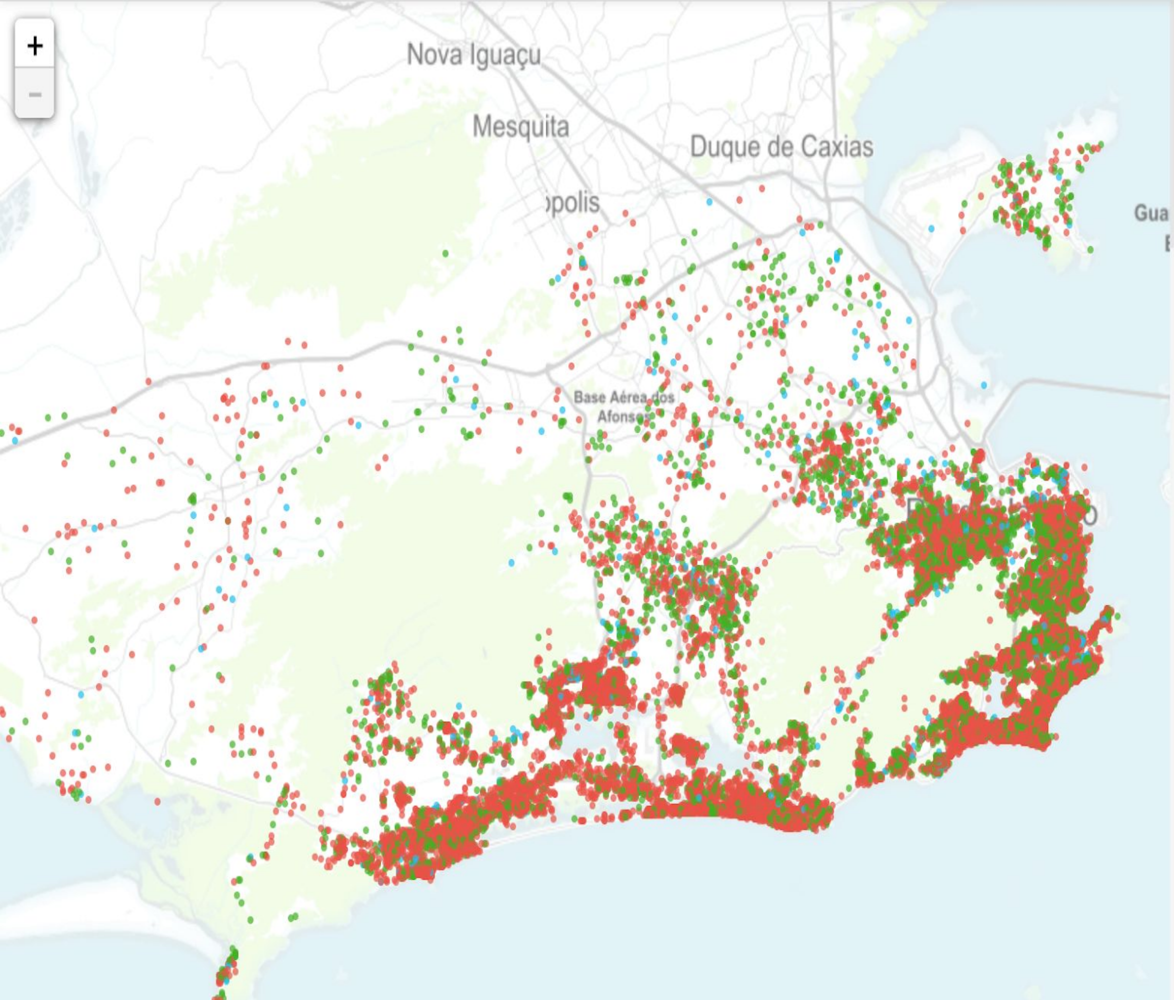
- Look at the Big Picture
    - Frame the problem
    - Select a performance measure
    - Check the assumptions
- Get the Data
    - Create the workspace
    - Download the data
    - Take a quick look at the data structure
- Prepare the Data
    - Data cleaning
    - Handling text and categorical attributes
    - Feature scaling
- Select and Train a Model
    - Training and evaluating on the training set
    - Better evaluation using cross-validation
- Fine-Tune your model
    - Grid search/Randomized search
    - Ensemble methods
- Launch, Monitor and Maintain



A general ML workflow




Inside Airbnb
Adding data to the debate
INDEPENDENT, NON-COMMERCIAL, OPEN SOURCE DATA TOOL
How is Airbnb really being used in and affecting your neighborhood?
Airbnb IN NYC
IMPACT ON HOUSING
OUT OF MORE THAN 27,000 LISTINGS:
15.5K are for the entire home (57%)
OF WHICH:
53% are frequently rented (for more than 60 days)
24% are multi-listings (where the host is not living there)
SEE Airbnb ACTIVITY OVER TIME IN YOUR NEIGHBORHOOD
SoHo 2012
SoHo 2013
SoHo 2014
FILTER by Neighborhood
Chelsea
HOST "JOHN D" 17 listings
VIEW TOP HOSTS' MULTIPLE LISTINGS
50+ data points per listing
The data Airbnb doesn't want you to see!
NEXT...
• VISIT insideairbnb.com
• SHARE it widely
#insideairbnb #illegalhotels #affordablehousing #nyc
• DOWNLOAD the data
(open source; 50+ data points per listing)
Hoboken
Astoria
Woodside
Corona
Elmhurst
LeFrak City
Sunnyside
Battery Park
Hudson River
DUMBO
Marcy Houses
Bushwick
Fresh Pond
Parkside



Determining the optimal nightly rent price

Rio de Janeiro

Filter by:

Rio de Janeiro

35,887

out of **35,887** listings (100%)

About Airbnb in Rio de Janeiro

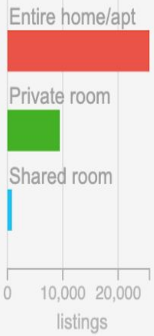
How is Airbnb really being used in and affecting your neighbourhoods?

Room Type

Only entire homes/apartments

Airbnb hosts can list entire homes/apartments, private or shared rooms.

Depending on the room type and activity, an airbnb listing could be more like a hotel, disruptive for neighbours, taking away housing, and illegal.



71.4%
entire homes/apartments

R$626
price/night

25,629 (71.4%)
entire home/apartments

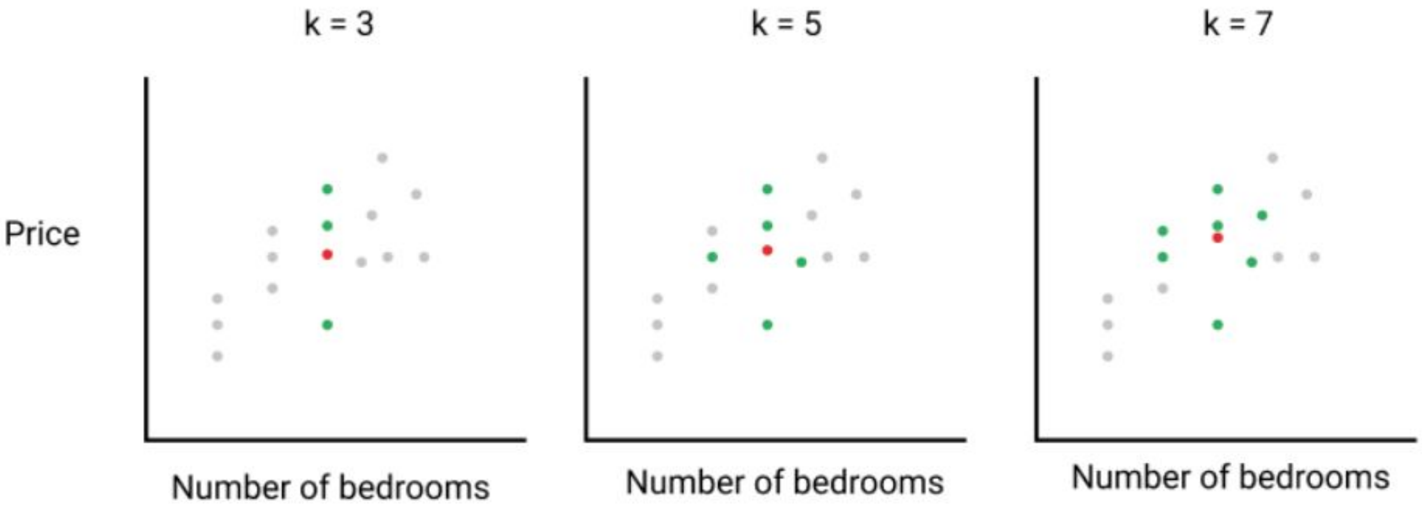
9,440 (26.3%)
private rooms

818 (2.3%)
shared rooms

- host_response_rate: the response rate of the host
- host_acceptance_rate: number of requests to the host that convert to rentals
- host_listings_count: number of other listings the host has
- latitude: latitude dimension of the geographic coordinates
- longitude: longitude part of the coordinates
- city: the city the living space resides
- zipcode: the zip code the living space resides
- state: the state the living space resides
- accommodates: the number of guests the rental can accommodate
- room_type: the type of living space (Private room, Shared room or Entire home/apt
- bedrooms: number of bedrooms included in the rental
- bathrooms: number of bathrooms included in the rental
- beds: number of beds included in the rental
- price: nightly price for the rental
- cleaning_fee: additional fee used for cleaning the living space after the guest leaves
- security_deposit: refundable security deposit, in case of damages
- minimum_nights: minimum number of nights a guest can stay for the rental
- maximum_nightss: maximum number of nights a guest can stay for the rental
- number_of_reviews: number of reviews that previous guests have left

106 cols

Select the number of similar listings, **k**, you want to compare with.



For this example, we'll use 3 for our **k** value.

dataset

bedrooms	price
1	160
3	350
1	60
1	95
1	50

Rank each listing by the similarity metric and select the first **k** listings.

dataset (ordered by similarity)

bedrooms	price	similarity
1	160	0
1	60	0
1	95	0
1	50	0
3	350	2

mean price

105

Euclidean distance - Univariate

	accommodates
our listing	8

Univariate case

$$d = \sqrt{(q_1 - p_1)^2}$$

$$d = |q_1 - p_1|$$

rio_listings

index	accommodates	distance
0	4	$(4 - 8)^2$
1	6	$(6 - 8)^2$
2	1	$(1 - 8)^2$
3	2	$(2 - 8)^2$

Euclidean distance (multivariate)

$$d = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \ldots + (q_n - p_n)^2}$$

index	host_listings_count	accommodates	bedrooms	bathrooms	beds
0	26	4	1	1	2
1	1	6	3	3	3

$(q_1 - p_1) + (q_2 - p_2) + \ldots + (q_n - p_n)$ — differences: $(26 - 1) + (4 - 6) + (1 - 3) + (1 - 3) + (2 - 3)$

$(q_1 - p_1)^2 + (q_2 - p_2)^2 + \ldots + (q_n - p_n)^2$ — squared differences: $(25)^2 + (-2)^2 + (-2)^2 + (-2)^2 + (-1)^2$

$\sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \ldots + (q_n - p_n)^2}$

Euclidean distance

$= \sqrt{625 + 4 + 4 + 4 + 1}$

$= \sqrt{638}$

$= 25.258661$

Error metrics (regression problem)

Mean Absolute Error

$$MAE = \frac{|actual_1 - predicted_1| + |actual_2 - predicted_2| + \ \ldots + |actual_n - predicted_n|}{n}$$

Mean Squared Error

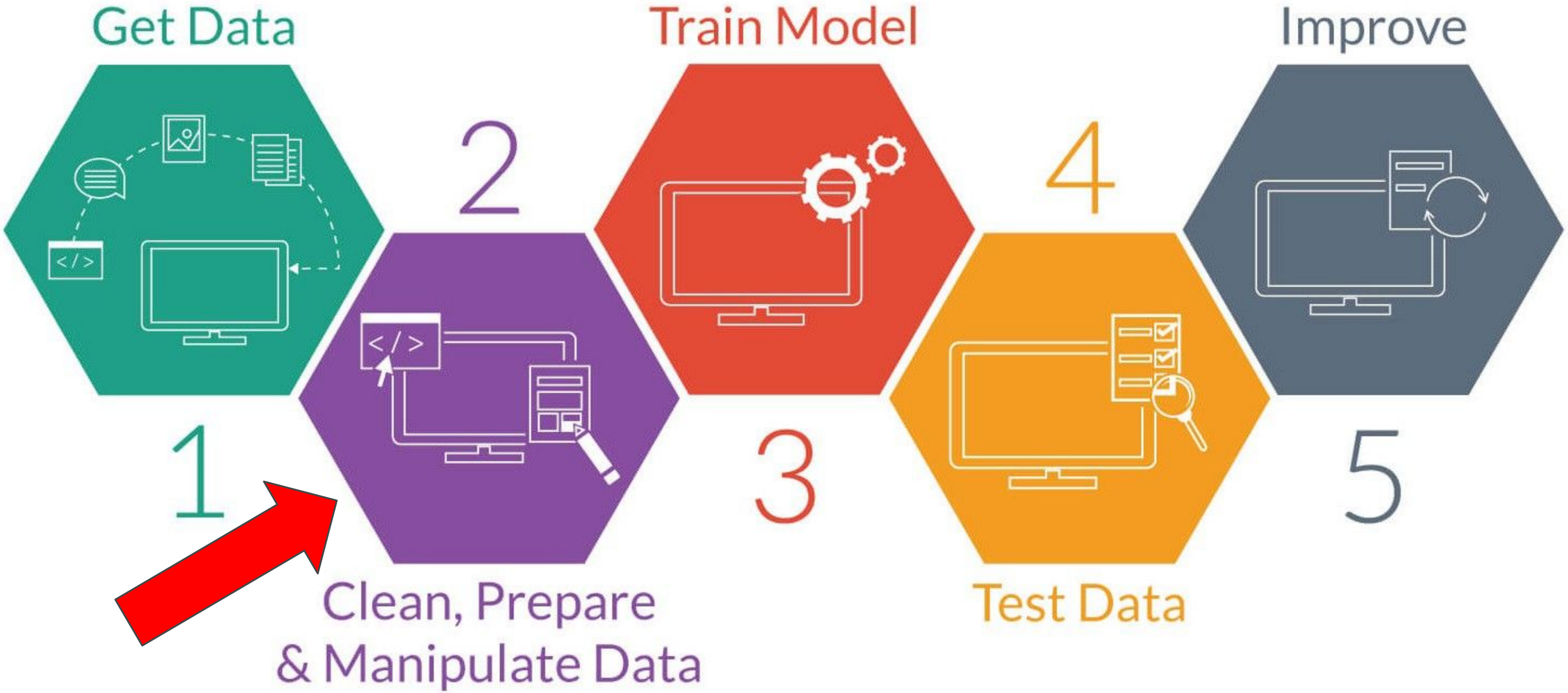
$$MSE = \frac{(actual_1 - predicted_1)^2 + (actual_2 - predicted_2)^2 + \ \ldots + (actual_n - predicted_n)^2}{n}$$

Root Mean Squared Error

$$RMSE = \sqrt{MSE}$$

https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d

A general ML workflow



Removing features

- **room_type**: e.g. **Private room**
- **city**: e.g. **Rio de Janeiro**
- **state**: e.g. **RJ**

- **host_response_rate**
- **host_acceptance_rate**
- **host_listings_count**

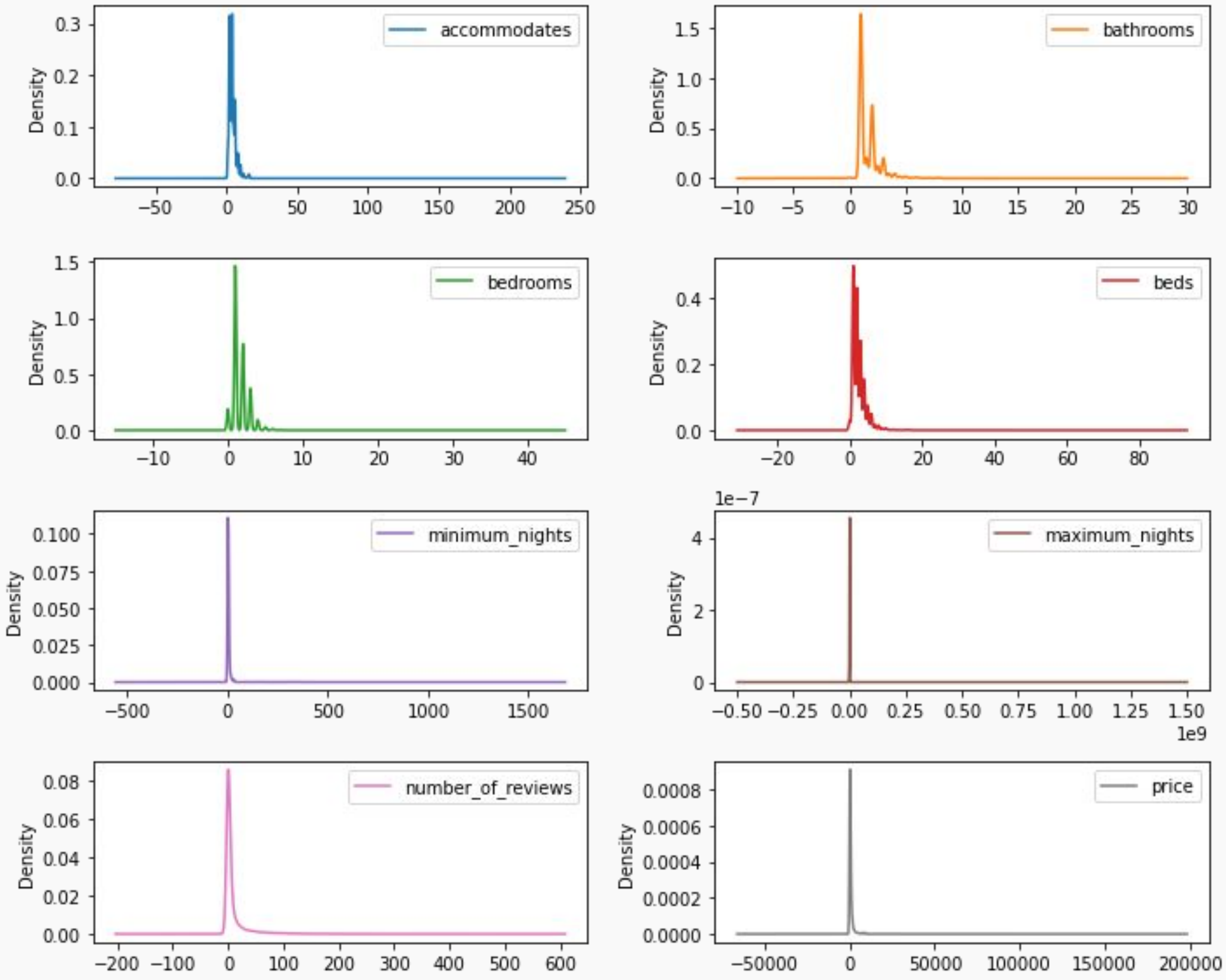
- **latitude**: e.g. **-22.92**
- **longitude**: e.g. **-43.23**
- **zipcode**: e.g. **20550-012**

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 35731 entries, 16575 to 33003
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   accommodates       35731 non-null  int64  
 1   bathrooms          35664 non-null  float64
 2   bedrooms           35652 non-null  float64
 3   beds               35400 non-null  float64
 4   price              35731 non-null  float64
 5   security_deposit   20051 non-null  object 
 6   cleaning_fee       24147 non-null  object 
 7   minimum_nights     35731 non-null  int64  
 8   maximum_nights     35731 non-null  int64  
 9   number_of_reviews  35731 non-null  int64  
dtypes: float64(4), int64(4), object(2)
memory usage: 3.0+ MB
```

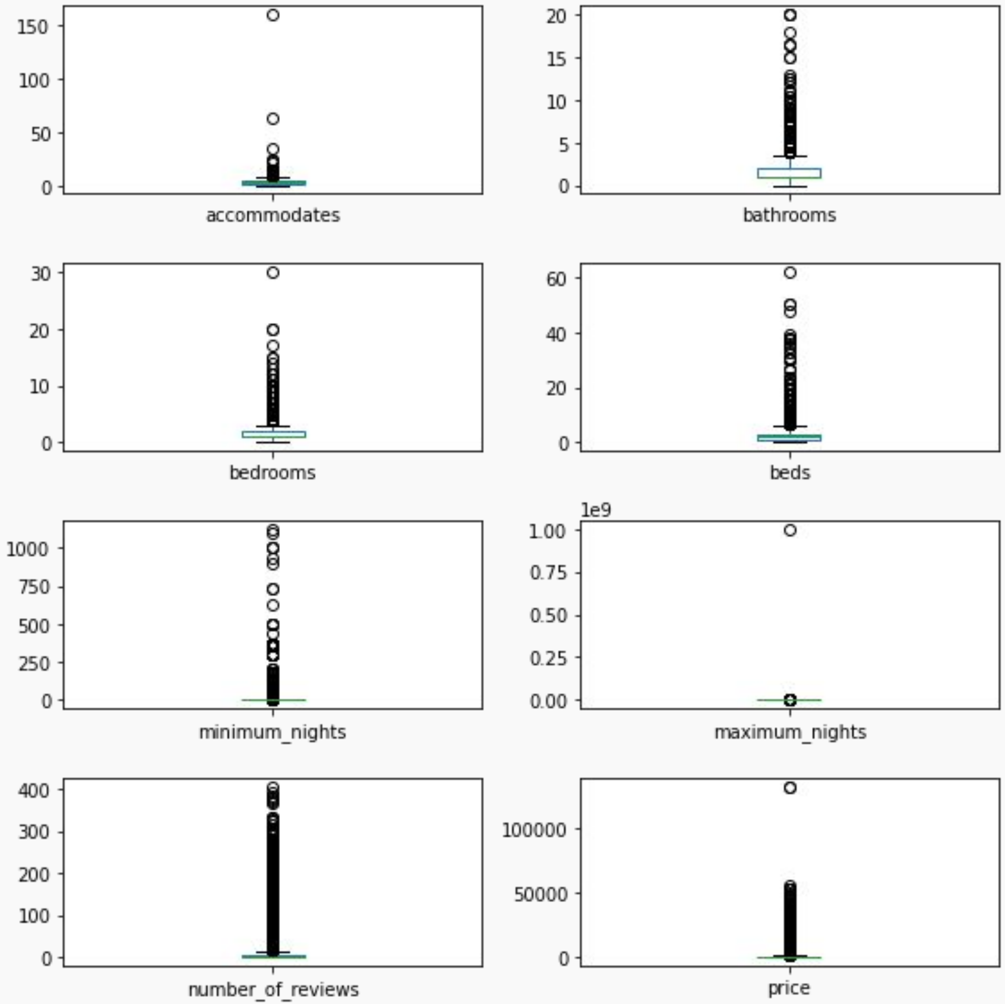
Data Preparation

	accommodates	bathrooms	bedrooms	beds	price	minimum_nights	maximum_nights	number_of_reviews
15364	6	3.0	3.0	3.0	$1,501.00	1	1125	0
33163	2	2.0	1.0	1.0	$181.00	5	10	0
28598	3	1.0	1.0	2.0	$140.00	1	1125	1
24716	1	1.0	1.0	1.0	$108.00	1	31	0
10299	6	2.0	3.0	5.0	$189.00	3	1125	53

Exploratory Data Analysis (EDA)

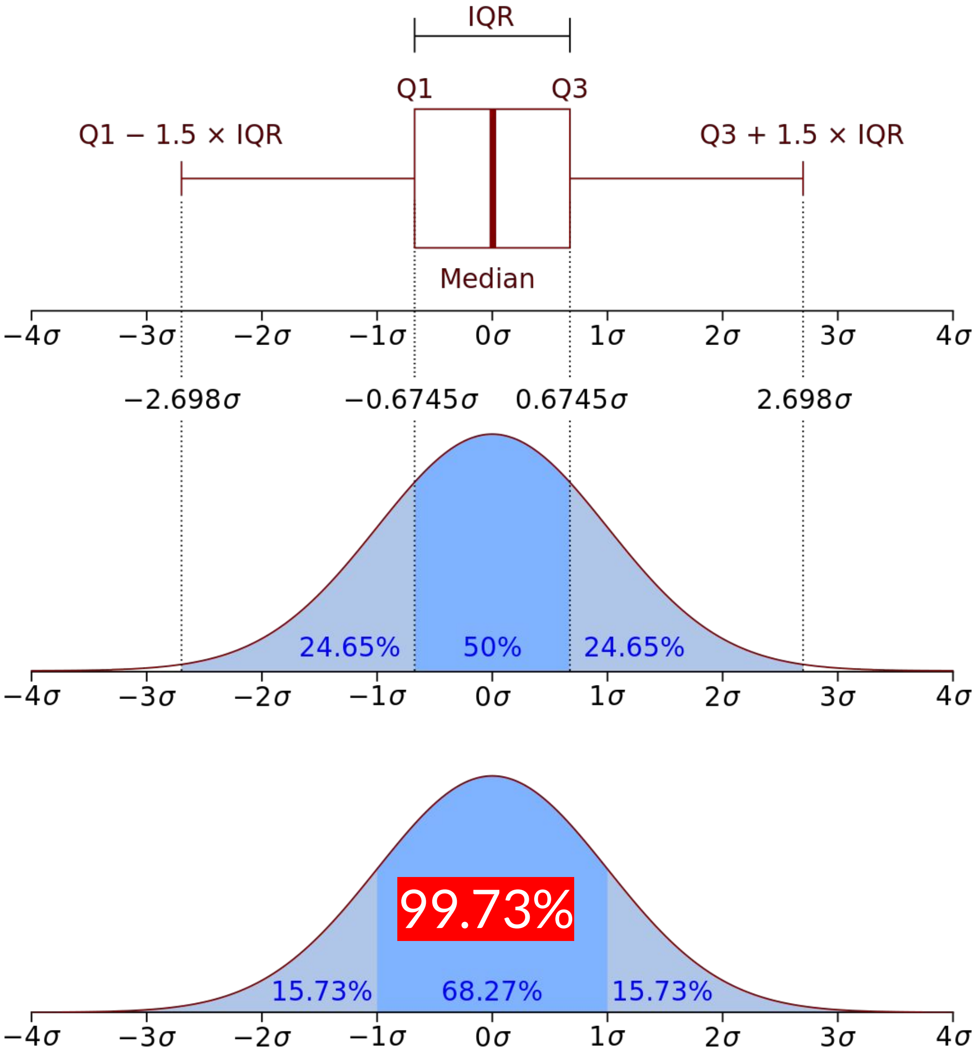


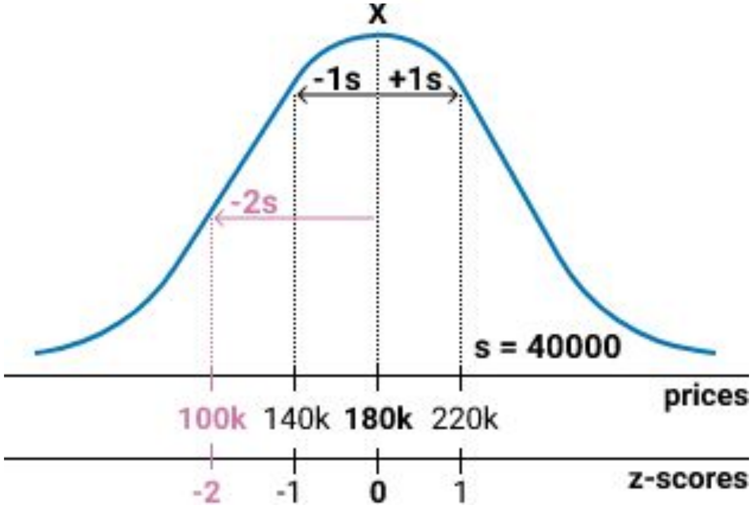
Outliers???



accommodates
bathrooms
bedrooms
beds
minimum_nights
maximum_nights
1e9
number_of_reviews
price

Using standardization and IQR for eliminate outlier





$$z = \frac{x - \mu}{\sigma}$$

Standardization (option #01)

	accommodates	bathrooms	bedrooms	beds	minimum_nights	maximum_nights	number_of_reviews	price
0	5	1.0	2.0	2.0	5	30	230	305.0
1	3	1.0	1.0	2.0	4	30	232	158.0
2	3	1.0	1.0	2.0	2	1125	256	251.0
3	3	1.5	1.0	2.0	2	89	155	347.0
4	2	1.0	1.0	1.0	3	28	299	220.0

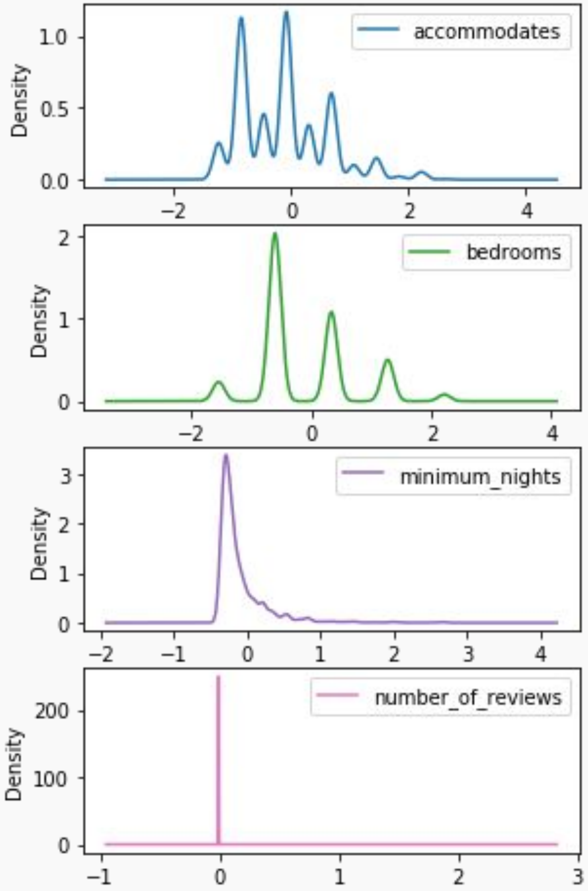
	accommodates	bathrooms	bedrooms	beds	minimum_nights	maximum_nights	number_of_reviews	price
0	0.309556	-0.665668	0.336640	-0.298117	-0.201519	0.013014	-0.005473	10.148919
1	-0.457286	-0.665668	-0.601309	-0.298117	-0.291600	-0.033019	-0.005473	10.240240
2	-0.457286	-0.665668	-0.601309	-0.298117	-0.234610	-0.125087	-0.005267	11.336088
3	-0.457286	-0.181869	-0.601309	-0.298117	-0.175782	-0.125087	-0.005462	6.724392
4	-0.840707	-0.665668	-0.601309	-0.793011	-0.253607	-0.079053	-0.005474	13.299483

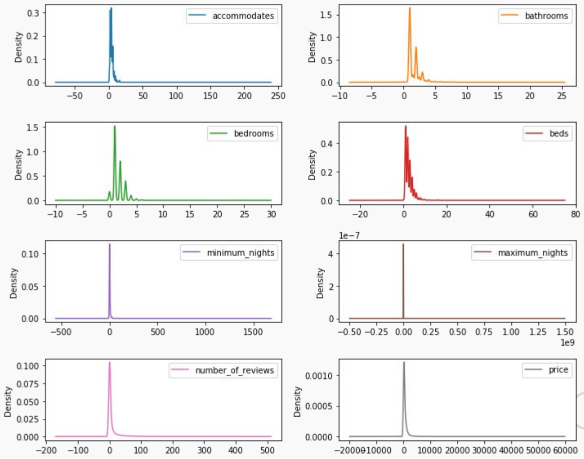
Standardization (mean zero and unitary std)

	accommodates	bathrooms	bedrooms	beds	minimum_nights	maximum_nights	number_of_reviews	price
count	3.530400e+04	3.530400e+04	3.530400e+04	3.530400e+04	3.530400e+04	3.530400e+04	3.530400e+04	3.530400e+04
mean	-1.096936e-14	2.071714e-14	-2.881814e-14	-8.062631e-16	-8.904096e-16	2.348094e-15	-1.210238e-15	-4.291996e-14
std	1.000014e+00	1.000014e+00	1.000014e+00	1.000014e+00	1.000014e+00	1.000014e+00	1.000014e+00	1.000014e+00
min	-1.214459e+00	-1.597979e+00	-1.488599e+00	-1.224339e+00	-3.296648e-01	-1.752524e-01	-5.447550e-03	-3.776610e-01
25%	-8.344702e-01	-6.426740e-01	-5.729848e-01	-7.435051e-01	-2.632334e-01	-1.752524e-01	-5.442101e-03	-3.776610e-01
50%	-7.449301e-02	-6.426740e-01	-5.729848e-01	-2.626715e-01	-2.043224e-01	-1.305888e-01	-5.236354e-03	-3.388490e-01
75%	3.054956e-01	3.126306e-01	3.426290e-01	2.181620e-01	-5.892523e-02	-4.126168e-02	-5.236354e-03	-1.447888e-01
max	5.920373e+01	1.750811e+01	2.597982e+01	2.858734e+01	5.497057e+01	4.993727e+01	1.878908e+02	1.538003e+01

```python
from sklearn.preprocessing import StandardScaler

# get data
target_columns = ["accommodates", "bathrooms","bedrooms",
          "beds","minimum_nights",
          "maximum_nights","number_of_reviews","price"]
rio_listings = pd.read_csv("listings.csv",usecols=target_columns)

# clean missing values
rio_listings.dropna(axis=0,inplace=True)

# apply z-score (mean=0, std=1)
rio_z_scored = pd.DataFrame(StandardScaler().fit_transform(rio_listings),
                            columns=target_columns,
                            index=rio_listings.index)

# remove outliers
rio_z_scored = rio_z_scored[(rio_z_scored < 2.698).all(axis=1)
               & (rio_z_scored > -2.698).all(axis=1)]
```

After Standardization





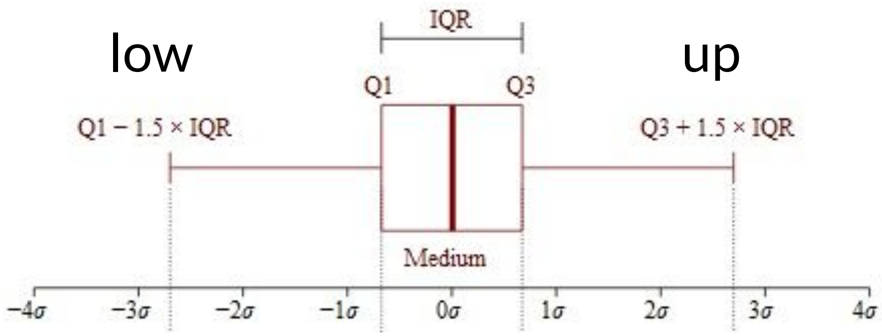
```
Q1 = rio_iqr.quantile(0.25)

accommodates          2.0
bathrooms             1.0
bedrooms              1.0
beds                  1.0
minimum_nights        1.0
maximum_nights       30.0
number_of_reviews     0.0
price               151.0
```

```
Q3 = rio_iqr.quantile(0.75)

accommodates           5.0
bathrooms              2.0
bedrooms               2.0
beds                   3.0
minimum_nights         4.0
maximum_nights      1125.0
number_of_reviews      4.0
price                598.0
```

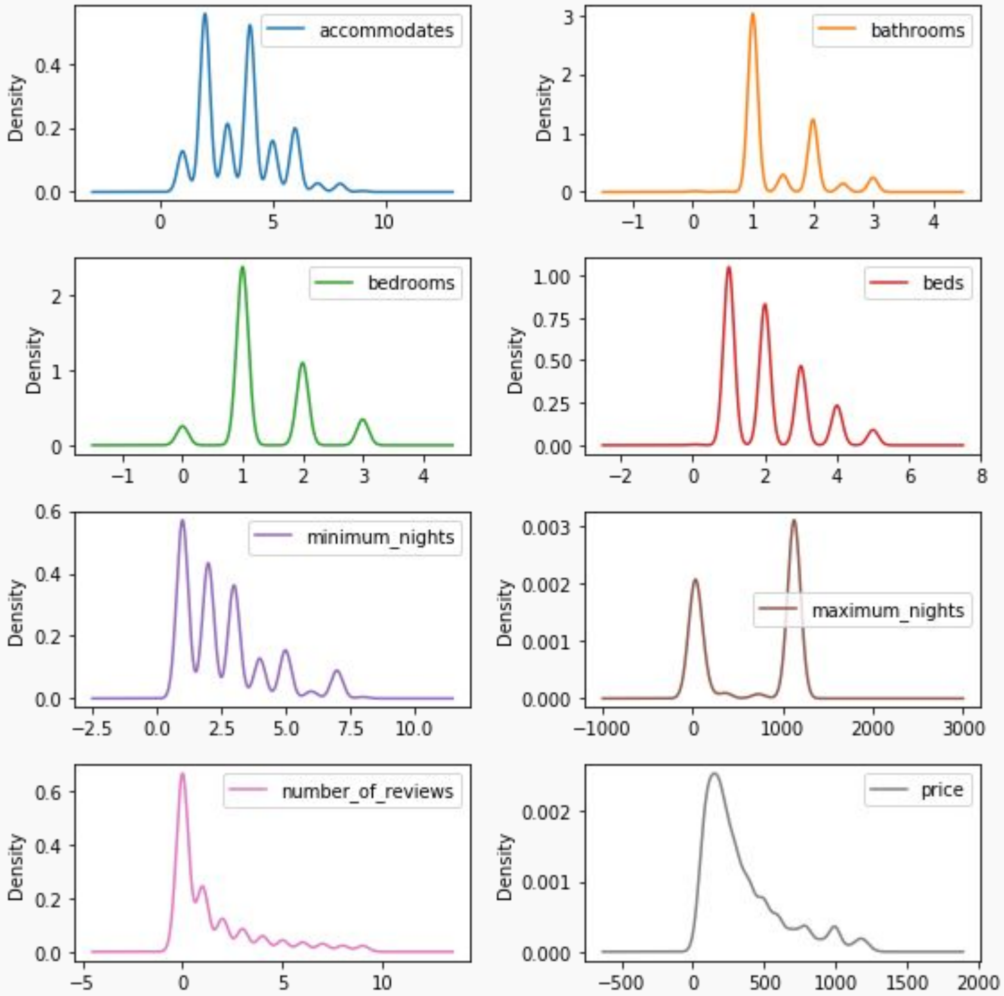
```
IQR = Q3 - Q1

accommodates           3.0
bathrooms              1.0
bedrooms               1.0
beds                   2.0
minimum_nights         3.0
maximum_nights      1095.0
number_of_reviews      4.0
price                447.0
```

low

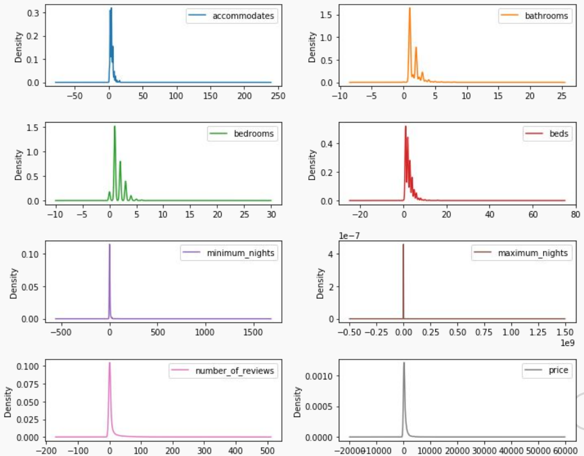
up



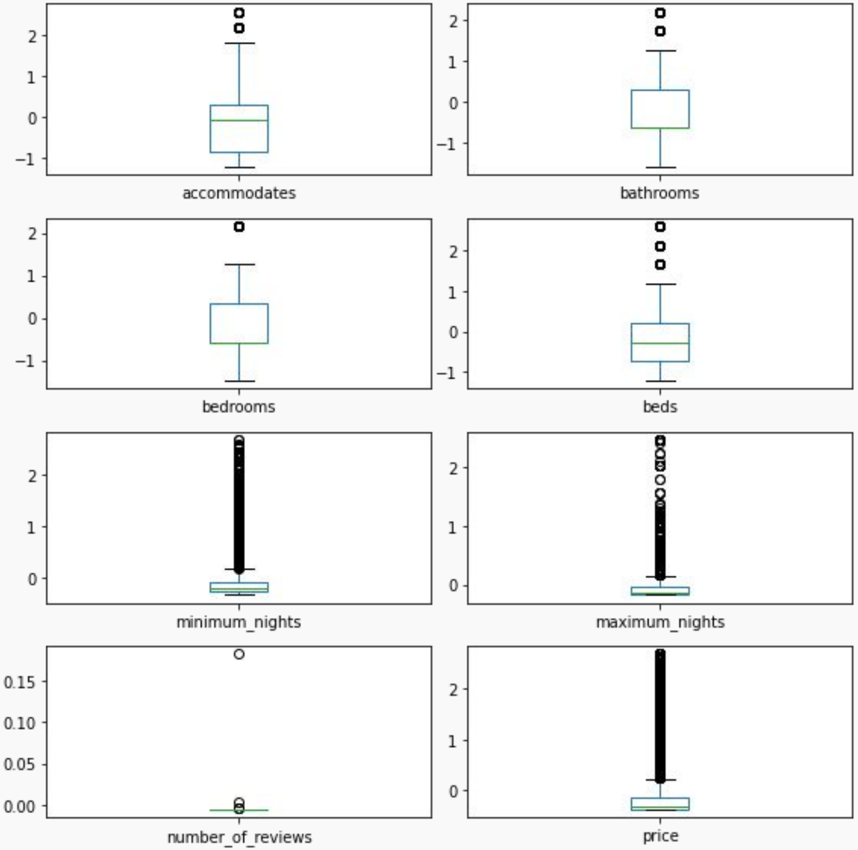

```
rio_iqr = rio_listings[target_columns].copy()
```



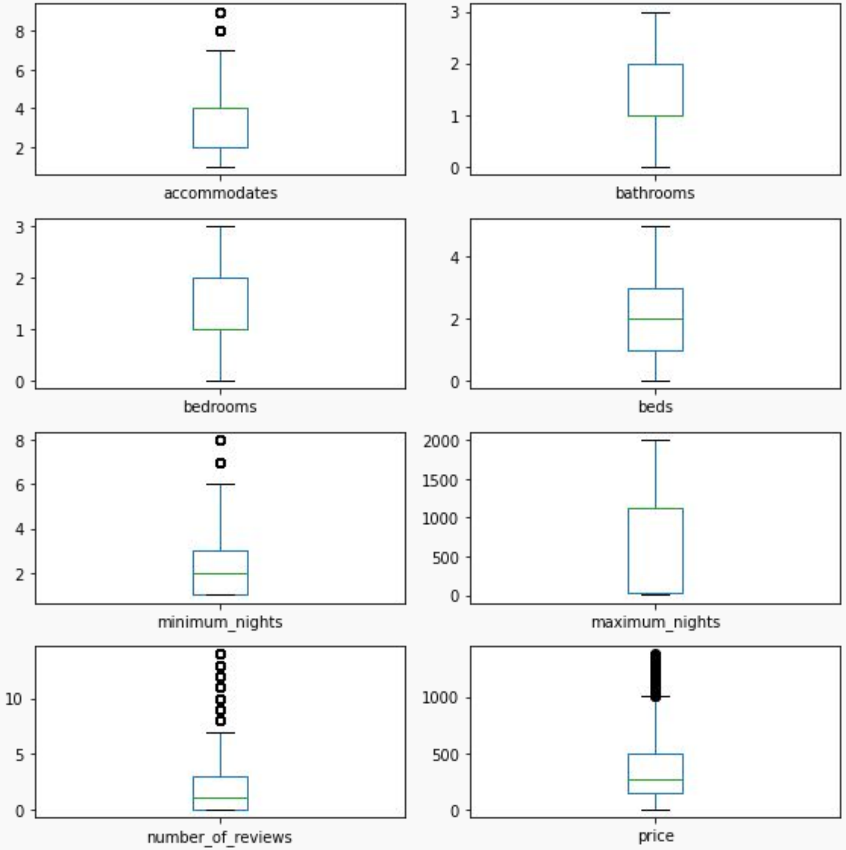
After outlier elimination (IQR method)



Z-Score Method

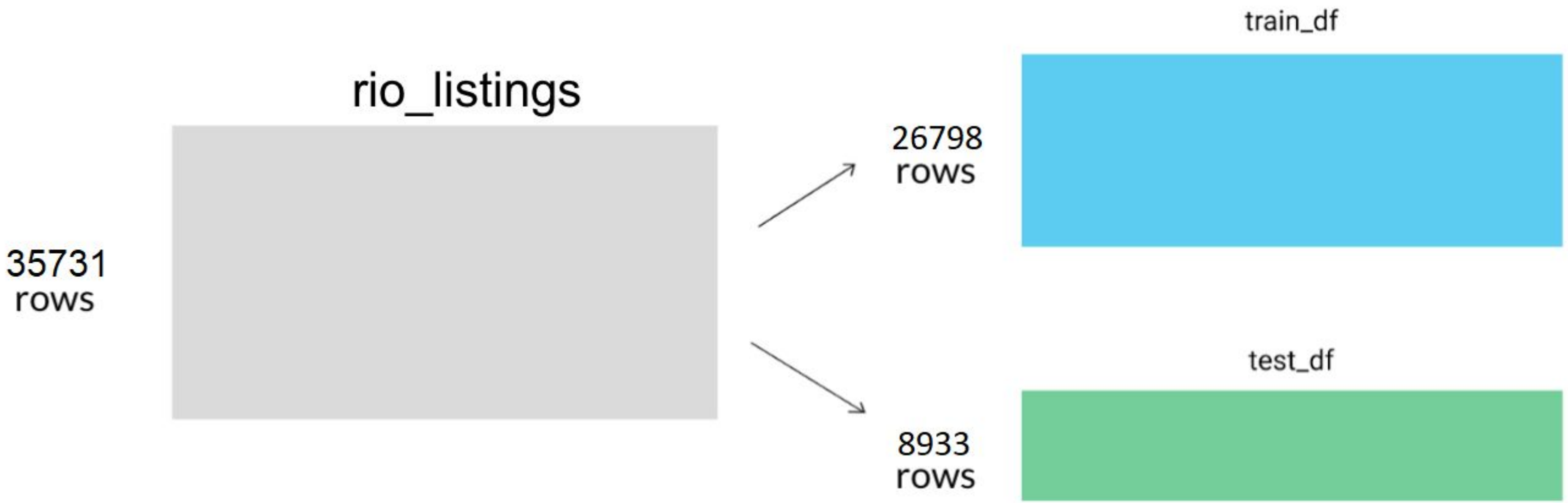


IQR Method

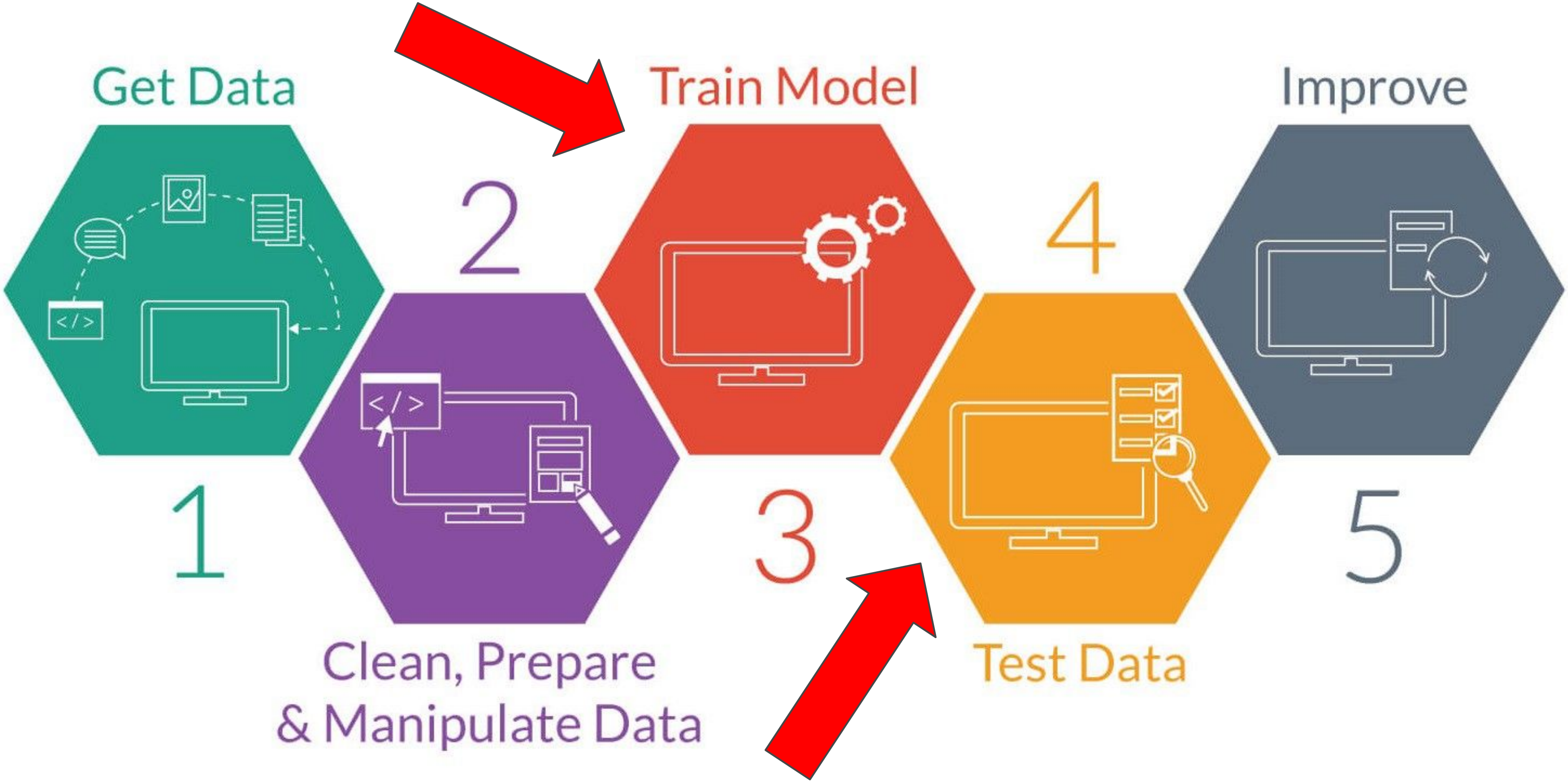


Separate data into Training and Test

```
X_train, X_test, Y_train, Y_test = train_test_split(rio_iqr.drop(axis=1,labels=["price"]),
                                                     rio_iqr["price"],
                                                     test_size=test_size,
                                                     random_state=seed)
```



A general ML workflow



scikit-learn

Machine Learning in Python

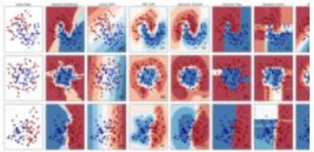
Getting Started | Release Highlights for 0.23 | GitHub

- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.
Algorithms: SVM, nearest neighbors, random forest, and more...

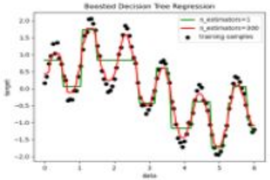


Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, nearest neighbors, random forest, and more...



Examples

Clustering

Automatic grouping of similar objects into sets.

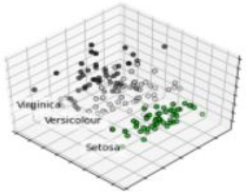
Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, and more...

Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: k-Means, feature selection, non-negative matrix factorization, and more...

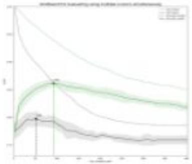


Examples

Model selection

Comparing, validating and choosing parameters and models.

Applications: Improved accuracy via parameter tuning
Algorithms: grid search, cross validation, metrics, and more...

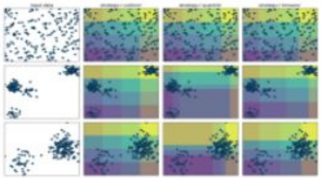


Examples

Preprocessing

Feature extraction and normalization.

Applications: Transforming input data such as text for use with machine learning algorithms.
Algorithms: preprocessing, feature extraction, and more...



Examples

Scikit-learn workflow

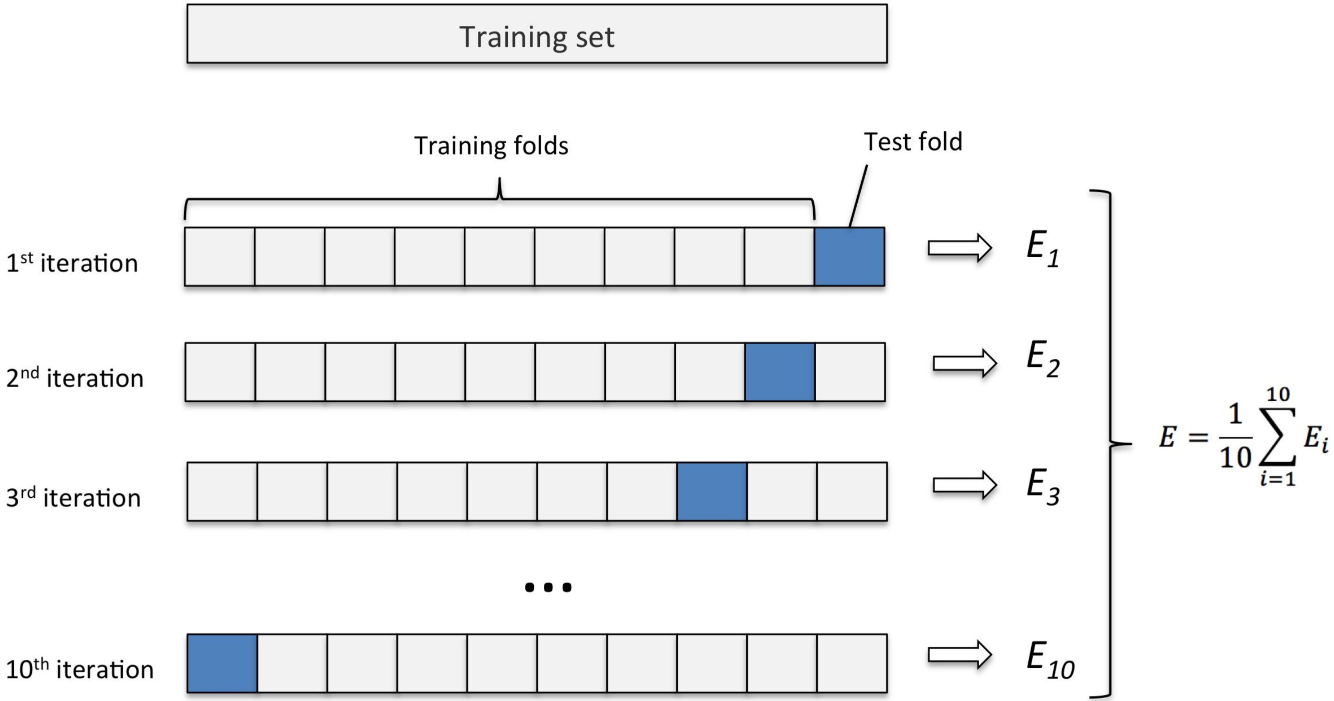
The scikit-learn workflow consists of 4 main steps:

- instantiate the specific machine learning model you want to use
- fit the model to the training data
- use the model to make predictions
- evaluate the accuracy of the predictions

```python
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error
import numpy as np

# instantiate a knn object
knn = KNeighborsRegressor(n_neighbors=5, n_jobs=-1)

# train the model
knn.fit(X_train,Y_train)

# predict
predict = knn.predict(X_test)

# evaluate
rmse = np.sqrt(mean_squared_error(Y_test,predict))
```

261.70

Better Evaluation using Cross-Validation



K-Fold Cross Validation

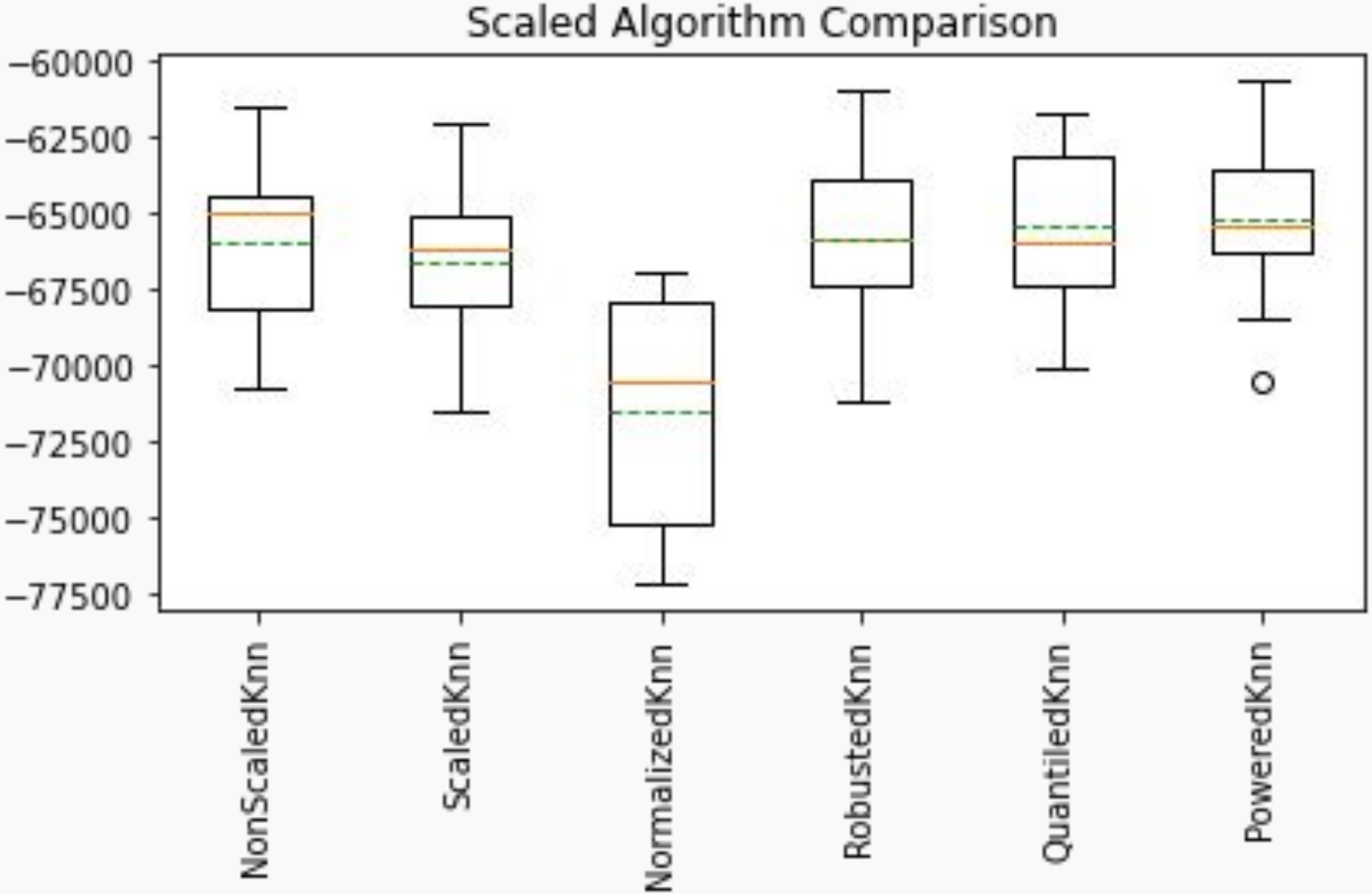
	Test	Train	Train	Train	Train
	Train	Test	Train	Train	Train
	Train	Train	Test	Train	Train
	Train	Train	Train	Test	Train
	Train	Train	Train	Train	Test
Errors	120.55	122.11	125.91	123.41	122.81

Mean Error 122.96

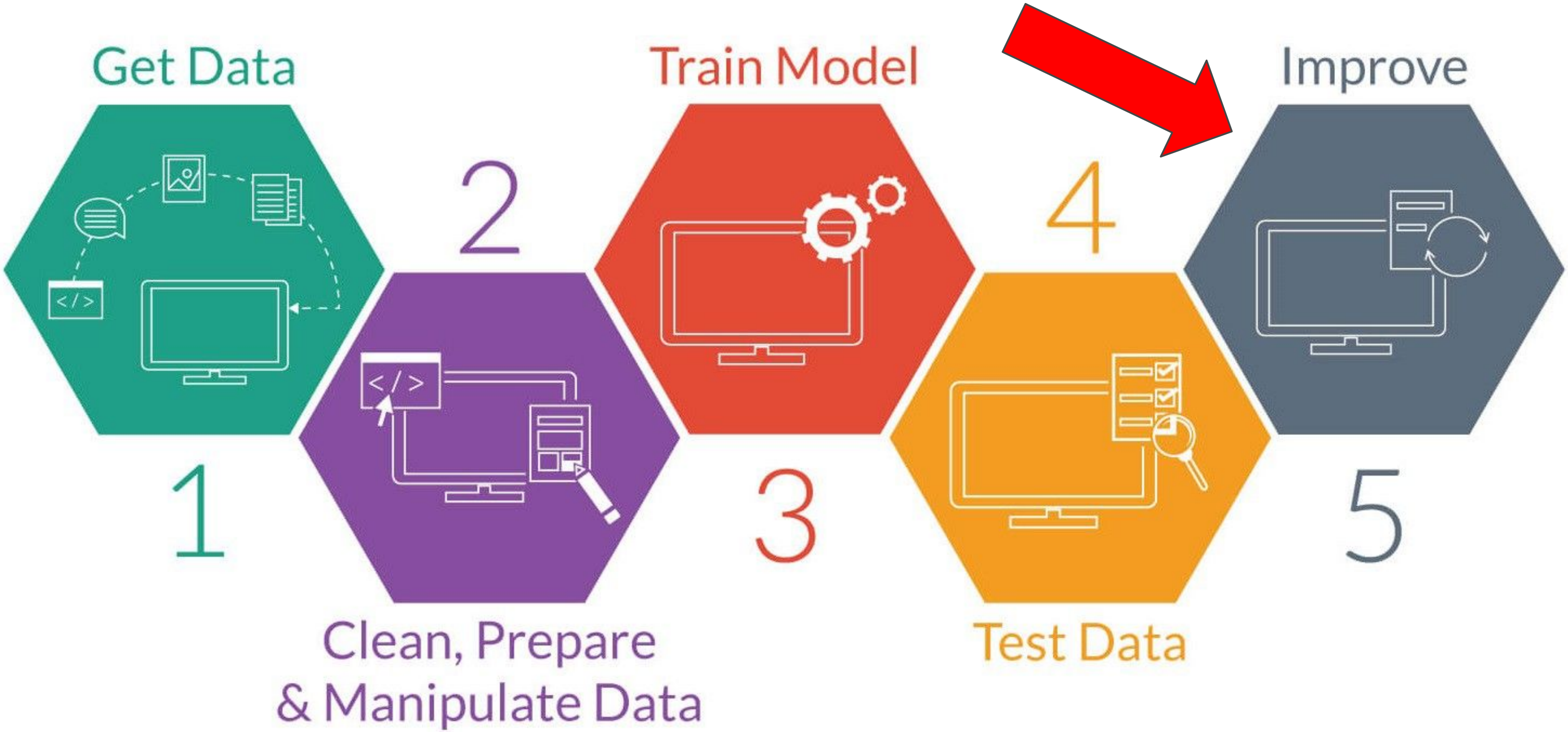
```python
# Test options and evaluation metric
num_folds = 10
seed = 7
scoring = 'neg_mean_squared_error'

# Standardize the dataset
pipelines = []
pipelines.append(('NonScaledKnn',
                  Pipeline([('KNN',
                             KNeighborsRegressor(n_neighbors=5, n_jobs=-1))])))
pipelines.append(('ScaledKnn',
                  Pipeline([('Scaler',
                             StandardScaler()),
                            ('KNN',
                             KNeighborsRegressor(n_neighbors=5, n_jobs=-1))])))

pipelines.append(('NormalizedKnn',
                  Pipeline([('Normalizer',
                             Normalizer()),
                            ('KNN',
                             KNeighborsRegressor(n_neighbors=5, n_jobs=-1))])))
```

```
results = []
names = []
for name, model in pipelines:
  kfold = KFold(n_splits=num_folds, random_state=seed)
  cv_results = cross_val_score(model, X_train, Y_train, cv=kfold, scoring=scoring)
  results.append(cv_results)
  names.append(name)
  print("%s Mean: %f Std: %f" % (name,
                                  np.sqrt(-cv_results.mean()),
                                  np.sqrt(cv_results.std())))
```

```
NonScaledKnn Mean: 256.768306 Std: 51.803277
ScaledKnn Mean: 257.982424 Std: 51.044869
NormalizedKnn Mean: 267.400248 Std: 61.709087
RobustedKnn Mean: 256.593685 Std: 54.918664
QuantiledKnn Mean: 255.834571 Std: 51.341062
PoweredKnn Mean: 255.384108 Std: 52.825958
```


Scaled Algorithm Comparison
−60000
−62500
−65000
−67500
−70000
−72500
−75000
−77500
NonScaledKnn
ScaledKnn
NormalizedKnn
RobustedKnn
QuantiledKnn
PoweredKnn

A general ML workflow



Improve the accuracy

- Increase the number of attributes the model uses to calculate similarity when ranking the closest neighbors
- When we vary the features that are used in the model, we're affecting the data that the model uses.
- Increase k, the number of nearby neighbors the model uses when computing the prediction
- On the other hand, varying the k value affects the behavior of the model independently of the actual data that's used when making predictions. Values that affect the behavior and performance of a model that are unrelated to the data that's used are referred to as hyperparameters.

Hyperparameter Optimization

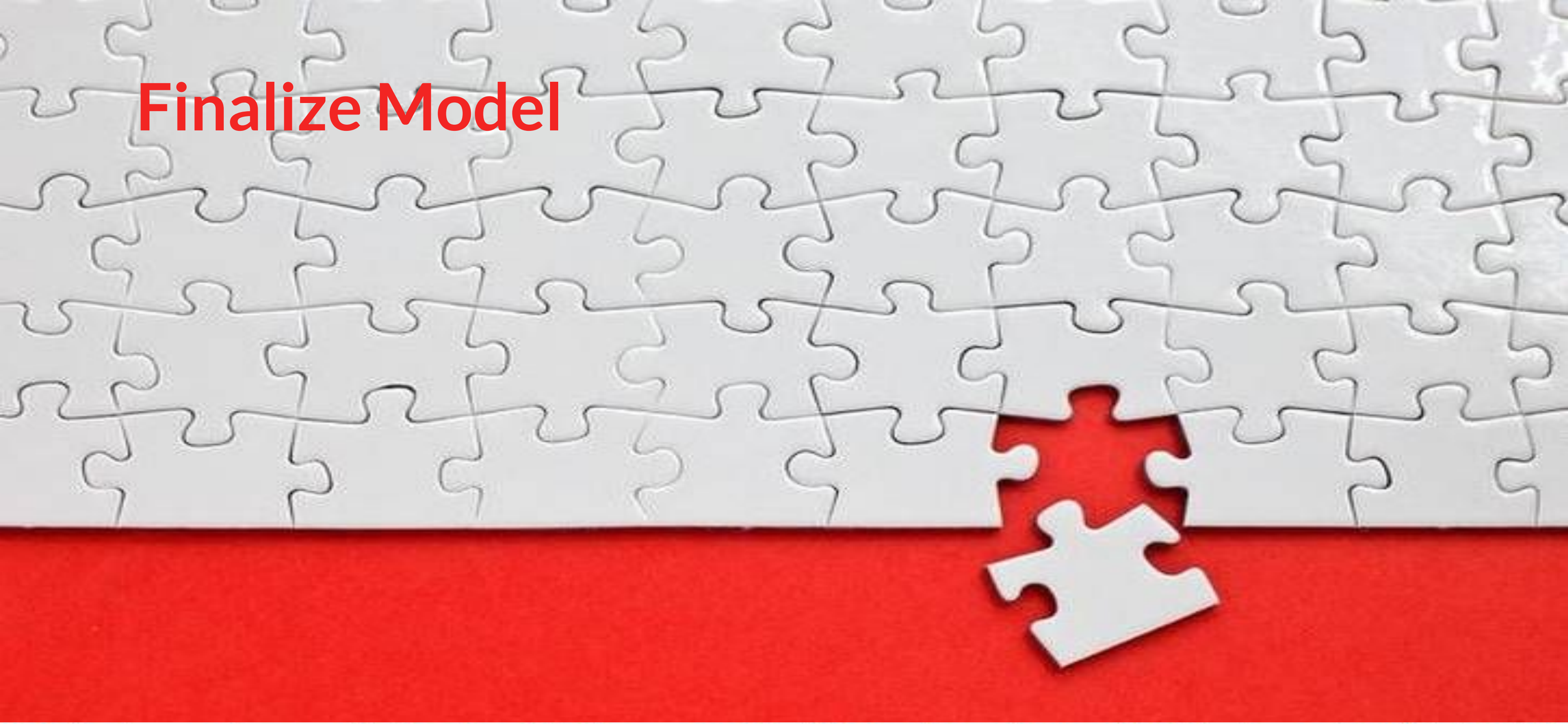
A simple but common hyperparameter optimization technique is known as grid search:

- selecting a subset of the possible hyperparameter values,
- training a model using each of these hyperparameter values,
- evaluating each model's performance,
- selecting the hyperparameter value that resulted in the lowest error value.

```python
# hyperparameter
k_values = np.array([1,3,5,7,9,11,13,15,17,19,21])
param_grid = dict(n_neighbors=k_values)

# scaler
scaler = PowerTransformer().fit_transform(X_train)

# instantiate a model
model = KNeighborsRegressor()

# Test options and evaluation metric
num_folds = 10
scoring = 'neg_mean_squared_error'

# cross-validation
kfold = KFold(n_splits=num_folds)
grid = GridSearchCV(estimator=model,
                    param_grid=param_grid,
                    scoring=scoring,
                    cv=kfold)

grid_result = grid.fit(scaler, Y_train)
# case you'd to use X_train without transformation
# grid_result = grid.fit(X_train, Y_train)
```

```
Best: 240.926226 using {'n_neighbors': 21}
323.382032 (53.549683) with: {'n_neighbors': 1}
268.216384 (54.585847) with: {'n_neighbors': 3}
255.372703 (52.610709) with: {'n_neighbors': 5}
250.221520 (48.154780) with: {'n_neighbors': 7}
247.199047 (49.738081) with: {'n_neighbors': 9}
244.862005 (48.366993) with: {'n_neighbors': 11}
243.531630 (46.711116) with: {'n_neighbors': 13}
242.880024 (45.295806) with: {'n_neighbors': 15}
242.143035 (44.669619) with: {'n_neighbors': 17}
241.561674 (46.065303) with: {'n_neighbors': 19}
240.926226 (45.935214) with: {'n_neighbors': 21}
```

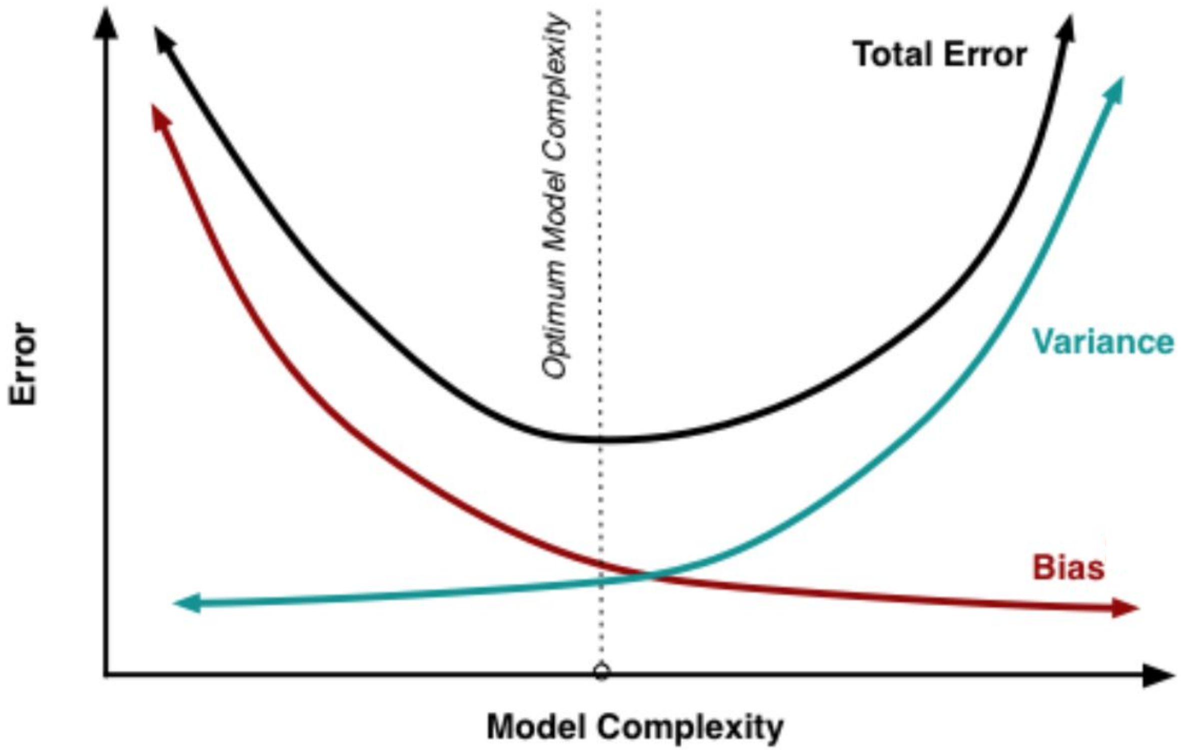
Finalize Model



```
predict = grid_result.best_estimator_.predict(PowerTransformer().fit_transform(X_test))
rmse = np.sqrt(mean_squared_error(predict,Y_test))
```

239.66376134298886

Bias-Variance Tradeoff



Next ....

https://www.imd.ufrn.br/

https://github.com/ivanovitchm/machinelearning2020.2