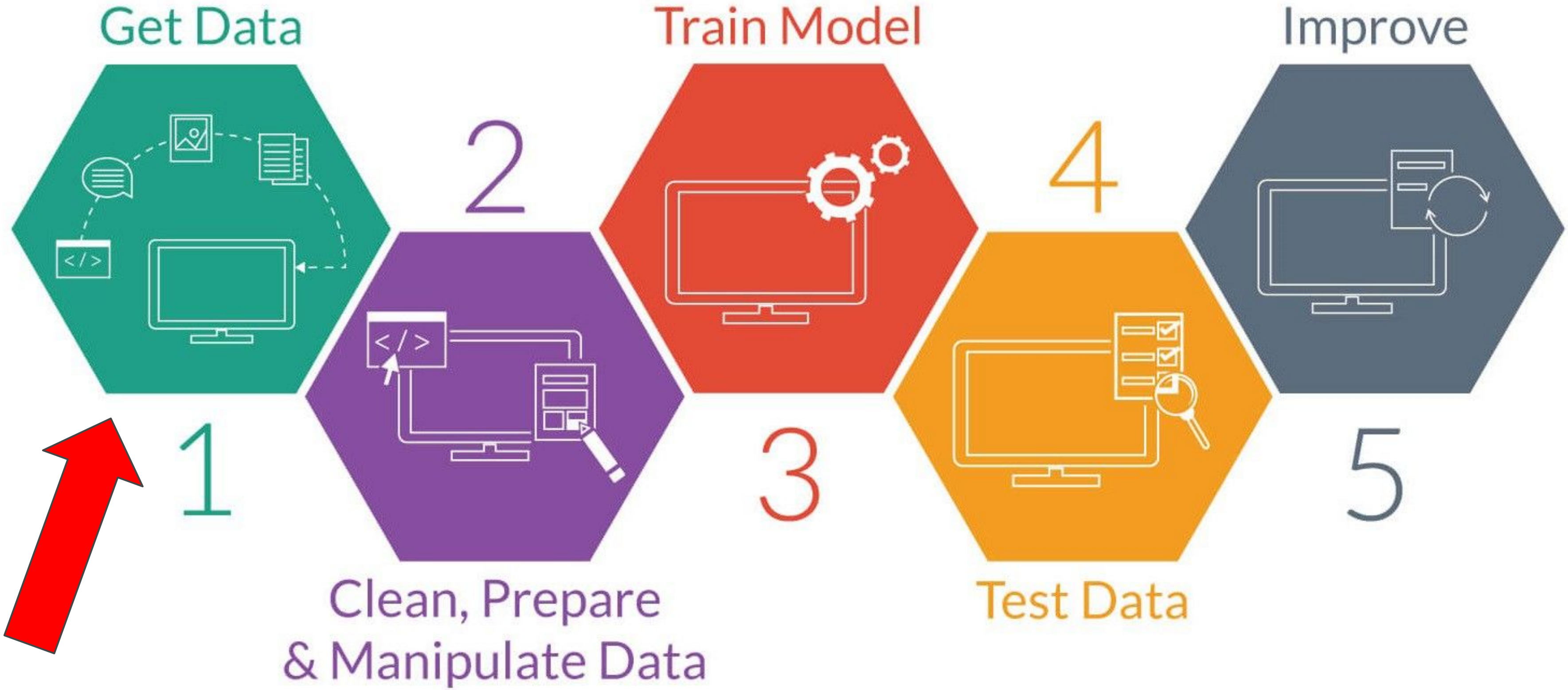
Lesson #07
K-Nearest Neighbors

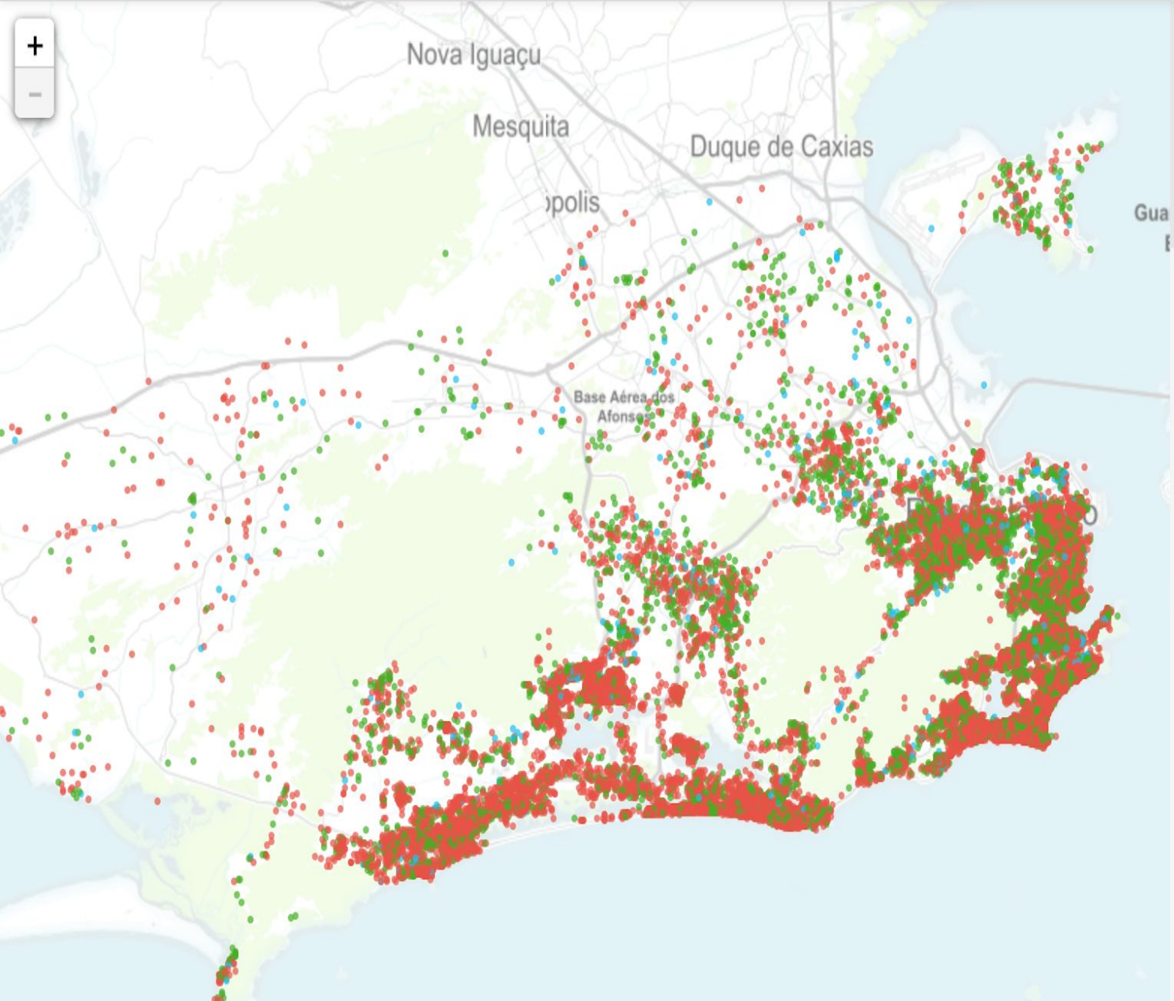
notas

- Univariate KNN
  - Euclidean distance for univariate
  - Function to make predictions
  - Error metrics
- Multivariate KNN
  - Normalize columns
  - Euclidean distance for multivariate
- Hyperparameter optimization
- Cross-Validation
- Pipeline & Gridsearch

notas

A general ML workflow





Rio de Janeiro

Filter by:

Rio de Janeiro

35,887

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

About Airbnb in Rio de Janeiro

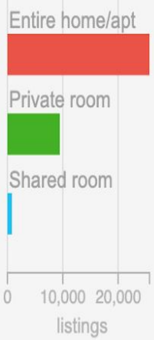
How is Airbnb really being used in and affecting your neighbourhoods?

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

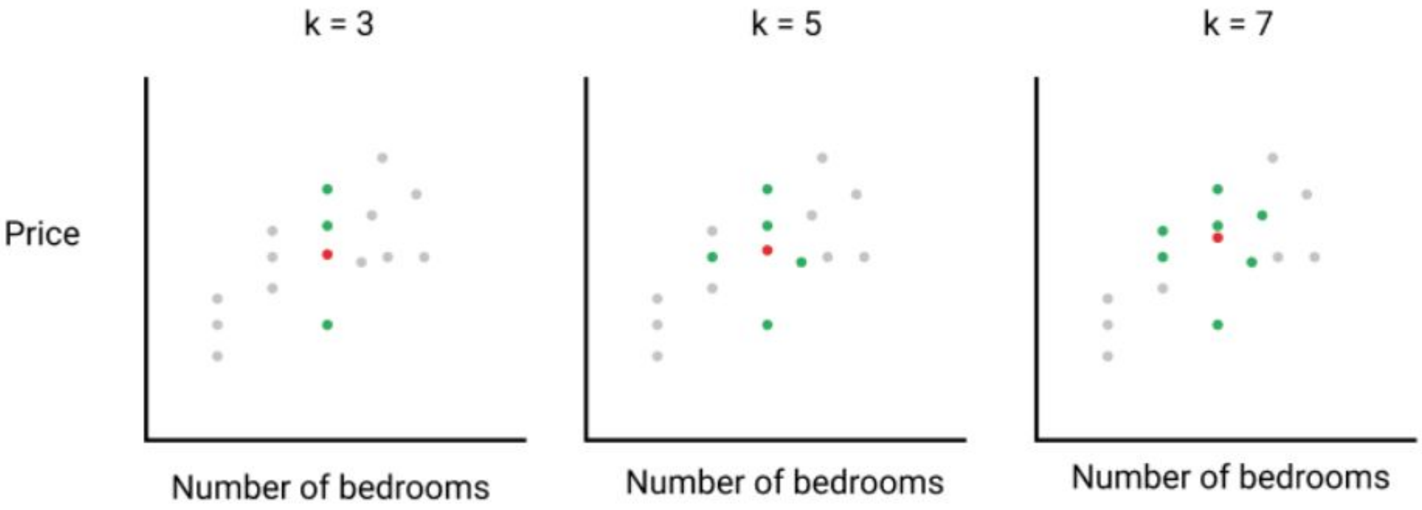
Determining the optimal nightly rent price

- 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

```
total 112M
-rw-r--r-- 1 root root 112M Aug  4 18:21 listings.csv
```

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



dataset

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

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

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

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

Mean Squared Error

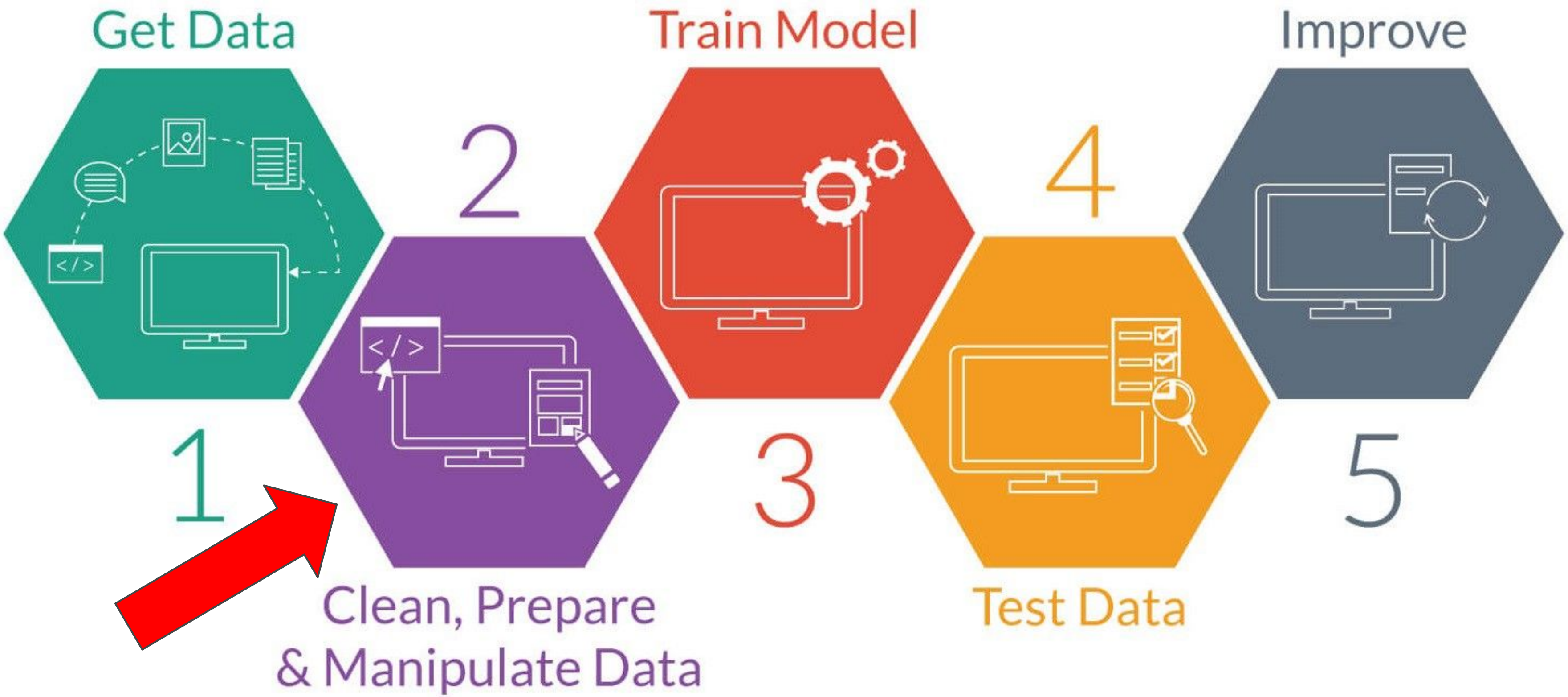
$$MSE = \frac{(actual_1 - predicted_1)^2 + (actual_2 - predicted_2)^2 + \ \dots + (actual_n - predicted_n)^2}{n}$$

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. **Washington**
- state: e.g. **DC**

- host_response_rate
- host_acceptance_rate
- host_listings_count

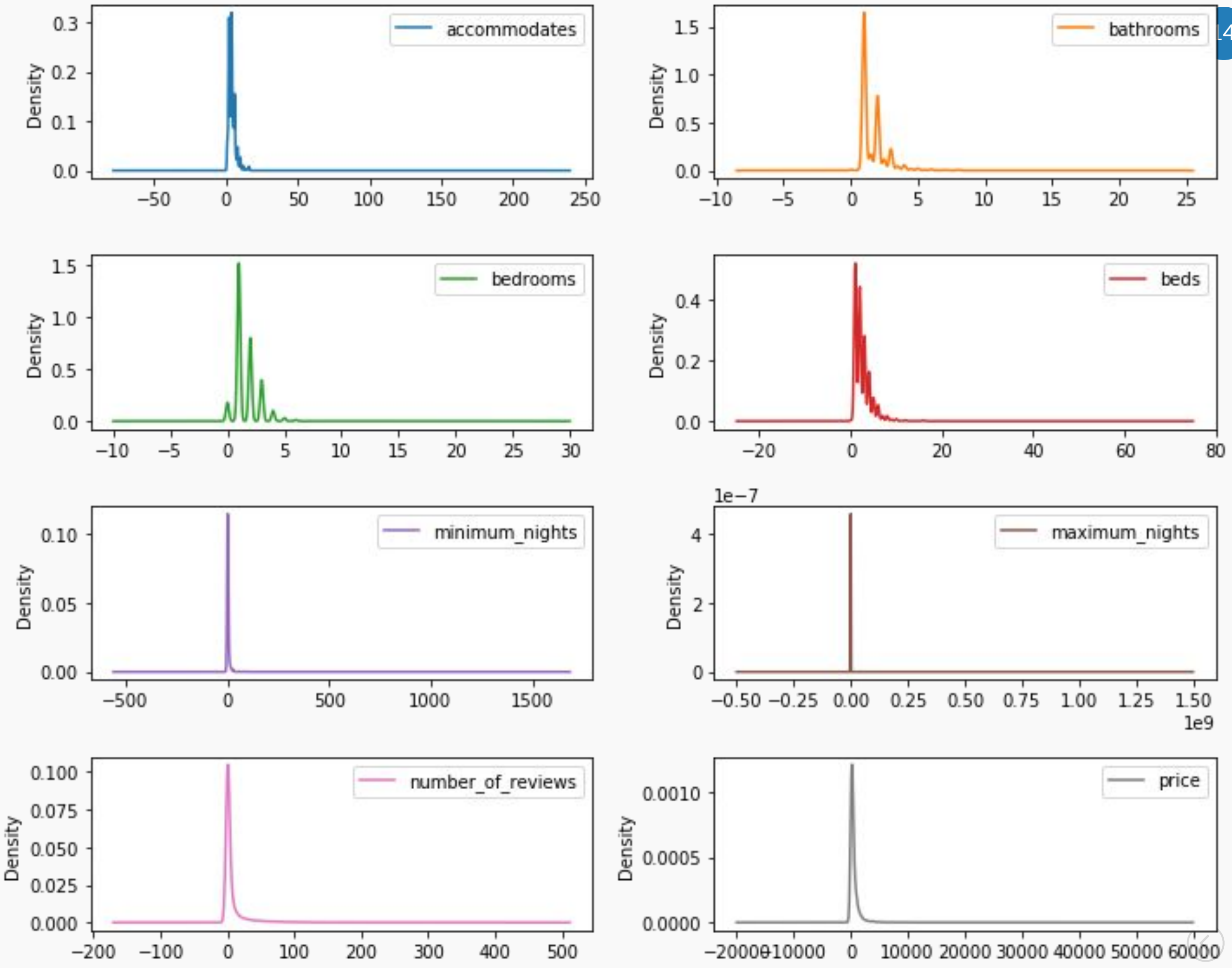
- latitude: e.g. **38.913458**
- longitude: e.g. **-77.031**
- zipcode: e.g. **20009**

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 35736 entries, 30160 to 33003
Data columns (total 8 columns):
accommodates        35736 non-null int64
bedrooms            35714 non-null float64
bathrooms           35660 non-null float64
beds                35693 non-null float64
price               35736 non-null float64
cleaning_fee        23389 non-null object
minimum_nights      35736 non-null int64
maximum_nights      35736 non-null int64
dtypes: float64(4), int64(3), object(1)
memory usage: 2.5+ MB
```

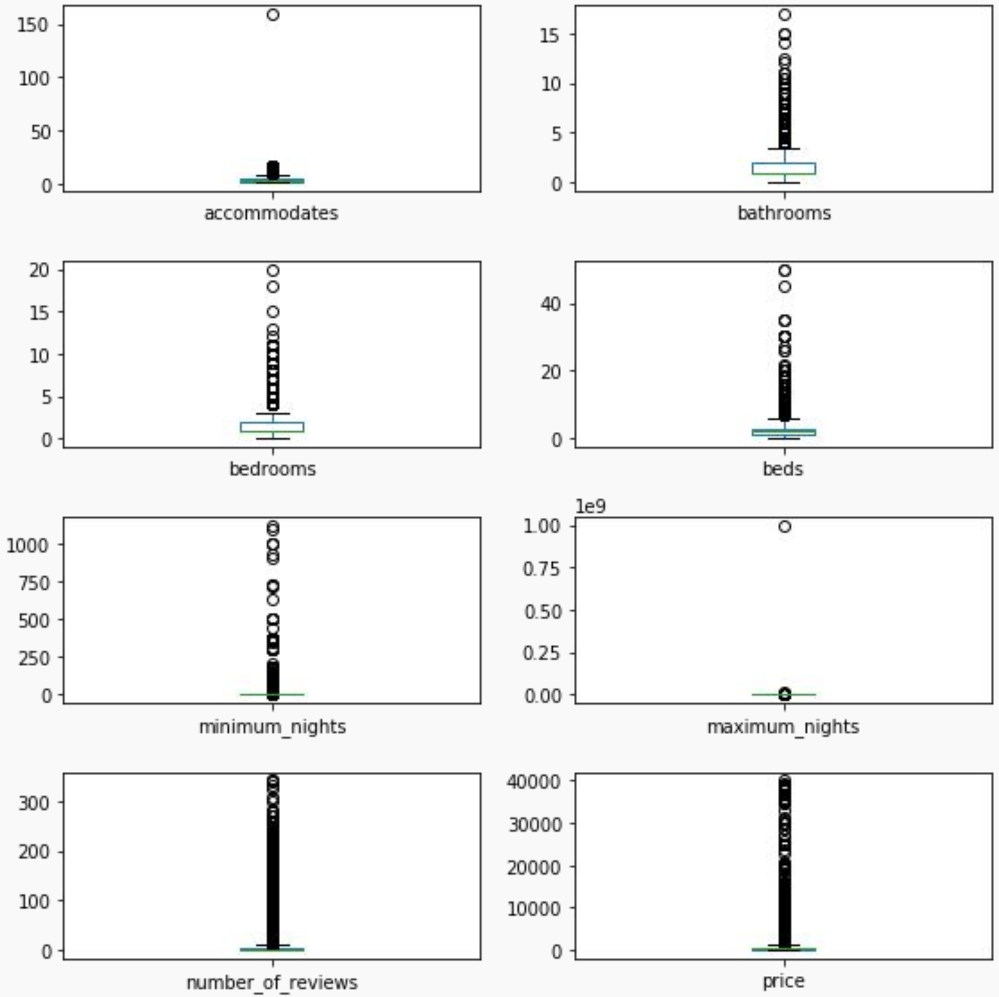
Data Preparation

	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)

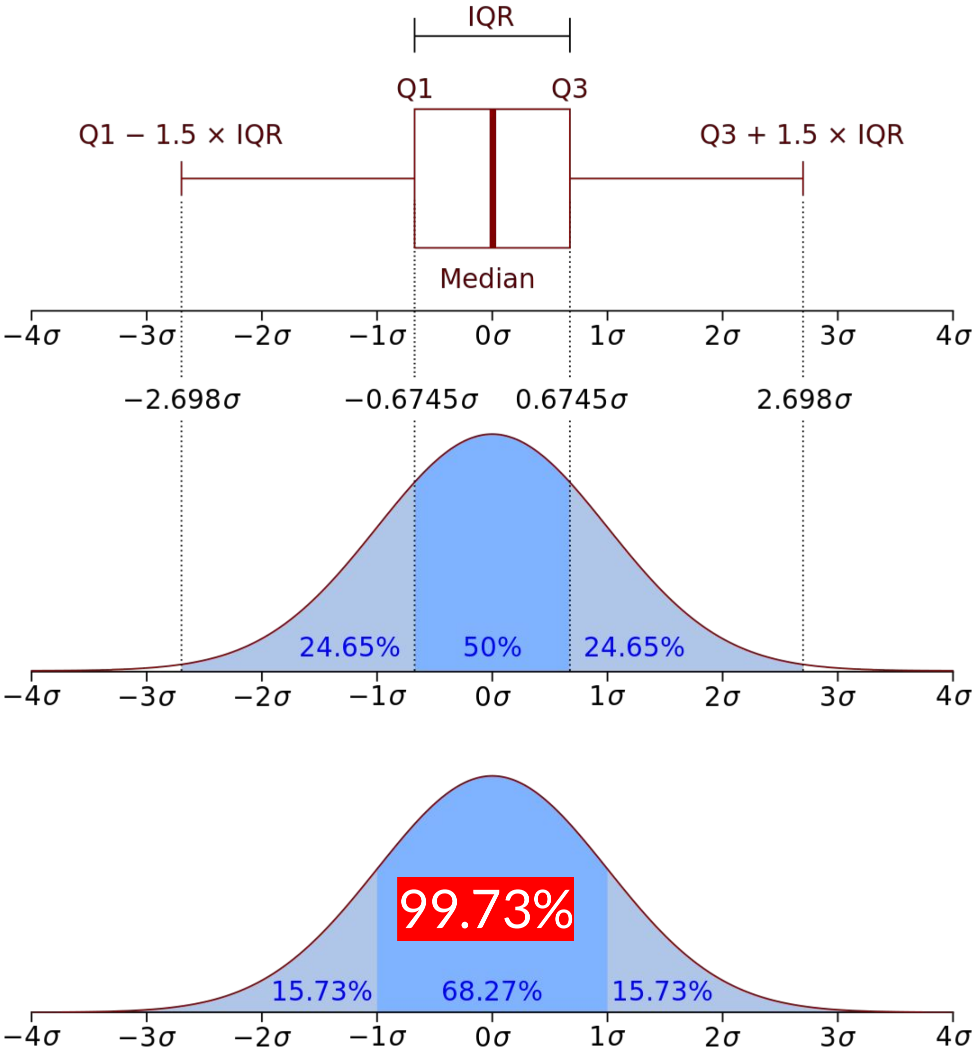


Outliers???

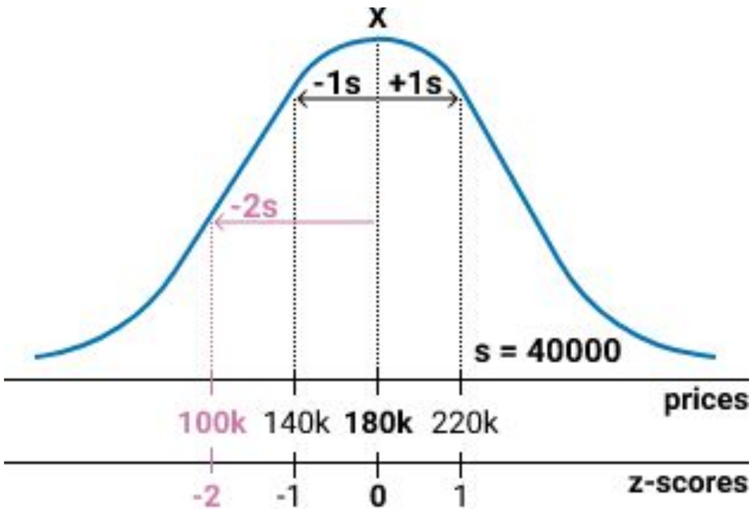


accommodates
bathrooms
bedrooms
beds
1e9
minimum_nights
maximum_nights
number_of_reviews
price

Outliers Removal



Using standardization and IQR for eliminate outlier



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

Standardization (option #01)

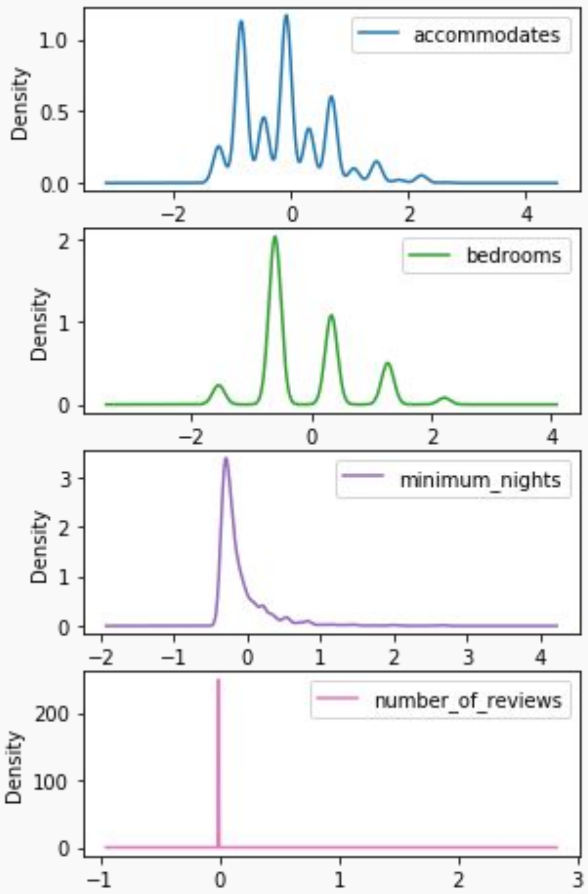
	accommodates	bathrooms	bedrooms	beds	minimum_nights	maximum_nights	number_of_reviews	price
0	5	1.0	2.0	2.0	5	30	230	305.0
1	3	1.0	1.0	2.0	4	30	232	158.0
2	3	1.0	1.0	2.0	2	1125	256	251.0
3	3	1.5	1.0	2.0	2	89	155	347.0
4	2	1.0	1.0	1.0	3	28	299	220.0

	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

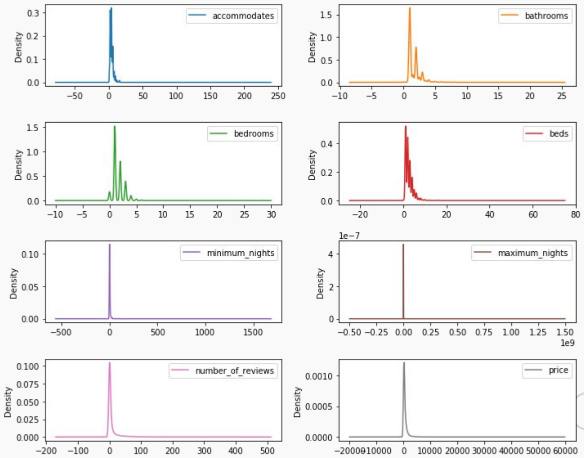
Standardization (mean zero and unitary std)

	accommodates	bathrooms	bedrooms	beds	minimum_nights	maximum_nights	number_of_reviews	price
count	3.563000e+04	3.563000e+04	3.563000e+04	3.563000e+04	3.563000e+04	3.563000e+04	3.563000e+04	3.563000e+04
mean	1.443714e-15	-4.422863e-15	2.209665e-16	7.564037e-16	-8.207401e-16	1.233092e-15	2.345833e-16	9.341216e-15
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	-1.224128e+00	-1.633266e+00	-1.539257e+00	-1.287905e+00	-3.884207e-01	-1.711206e-01	-5.478829e-03	-3.529624e-01
25%	-8.407069e-01	-6.656684e-01	-6.013086e-01	-7.930111e-01	-2.958891e-01	-1.711206e-01	-5.473355e-03	-3.529624e-01
50%	-7.386478e-02	-6.656684e-01	-6.013086e-01	-2.981171e-01	-2.113239e-01	-1.250869e-01	-5.266671e-03	-3.073020e-01
75%	3.095563e-01	3.019297e-01	3.366401e-01	1.967770e-01	-2.197118e-02	-3.301947e-02	-5.266671e-03	-1.703209e-01
max	5.973982e+01	1.481590e+01	1.721972e+01	2.345680e+01	2.412264e+01	5.147871e+01	1.887469e+02	1.521722e+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



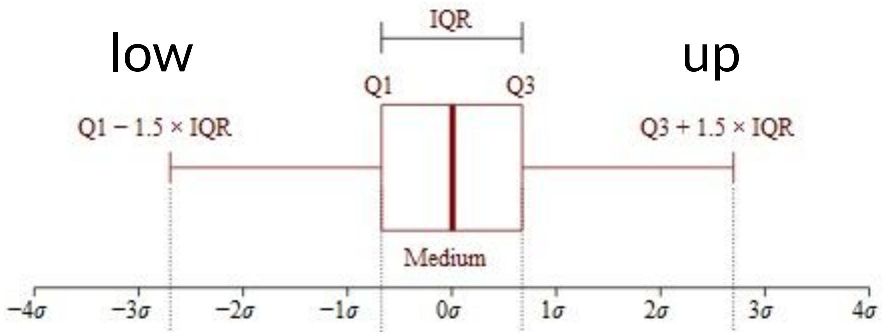
```
Q1 = rio_iqr.quantile(0.25)

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
```

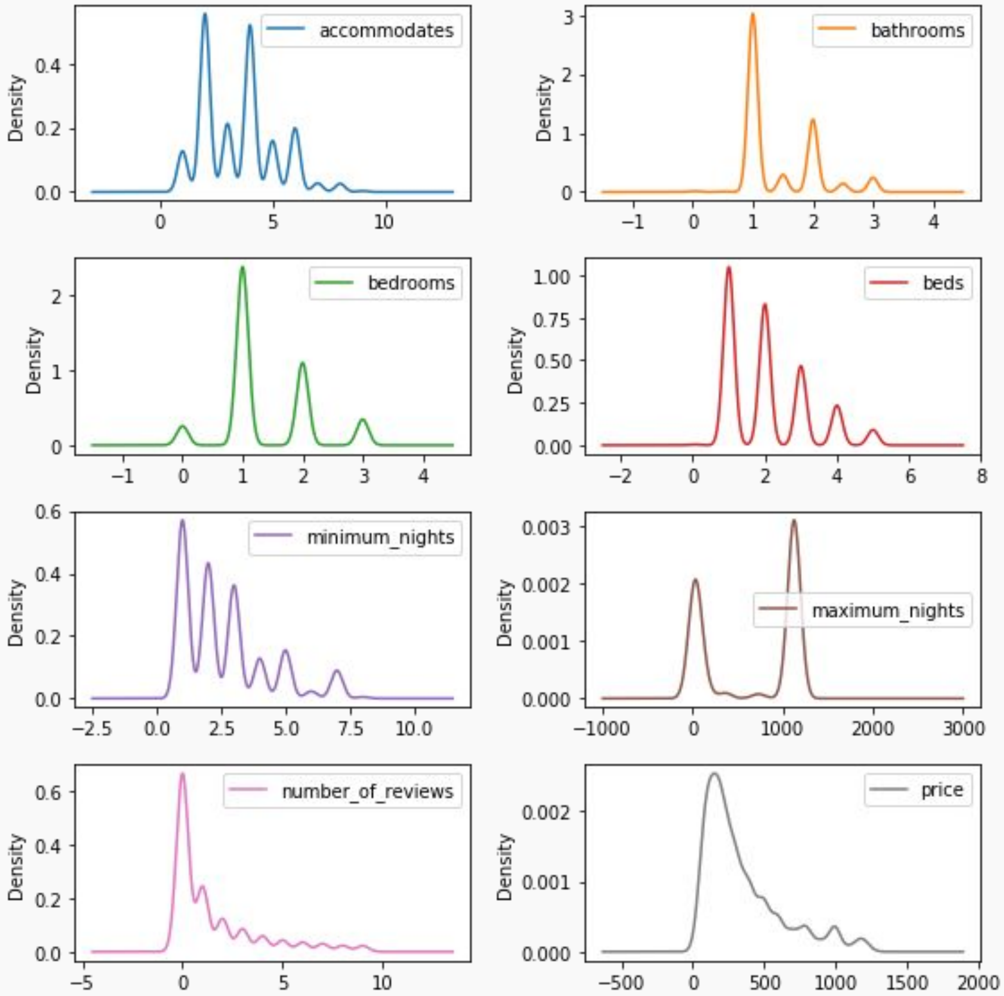
```
IQR = Q3 - Q1

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

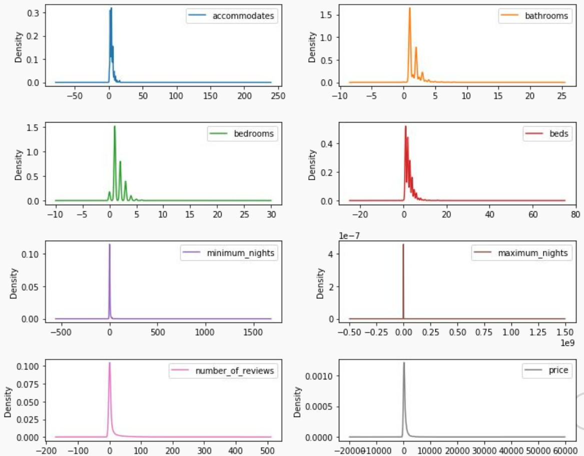
up



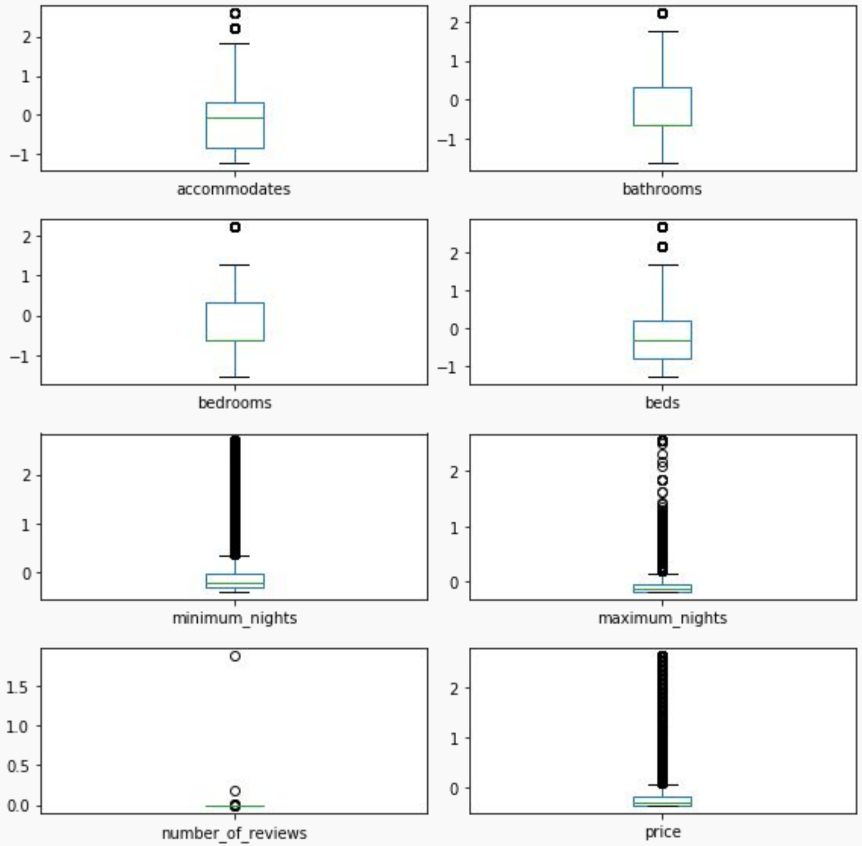

```
rio_iqr = rio_listings[target_columns].copy()
```



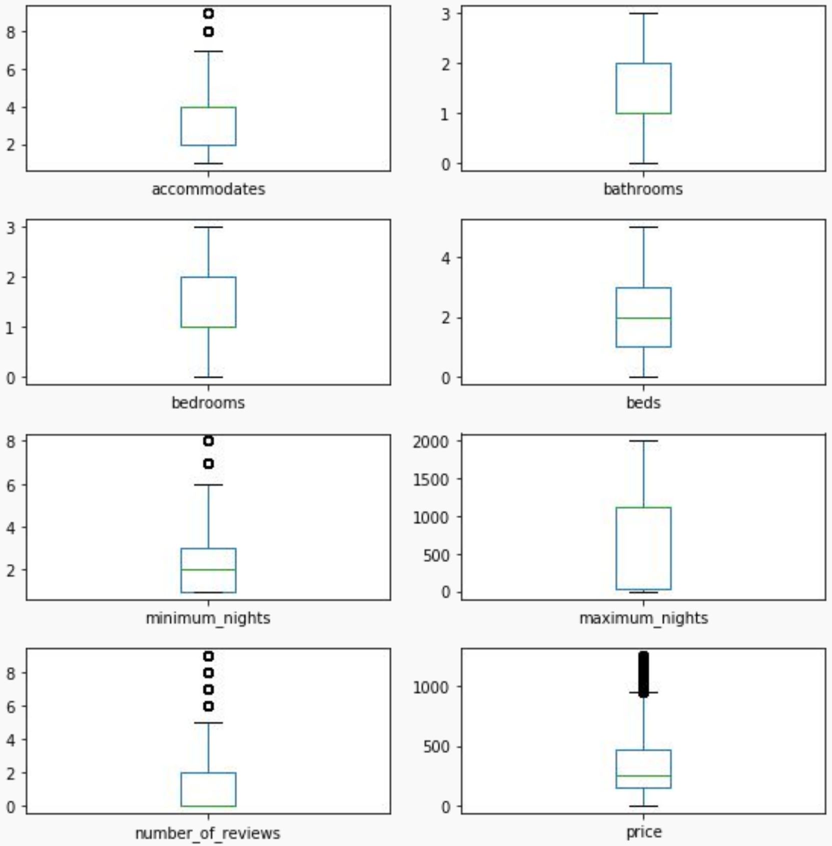
After outlier elimination (IQR method)



Z-Score Method

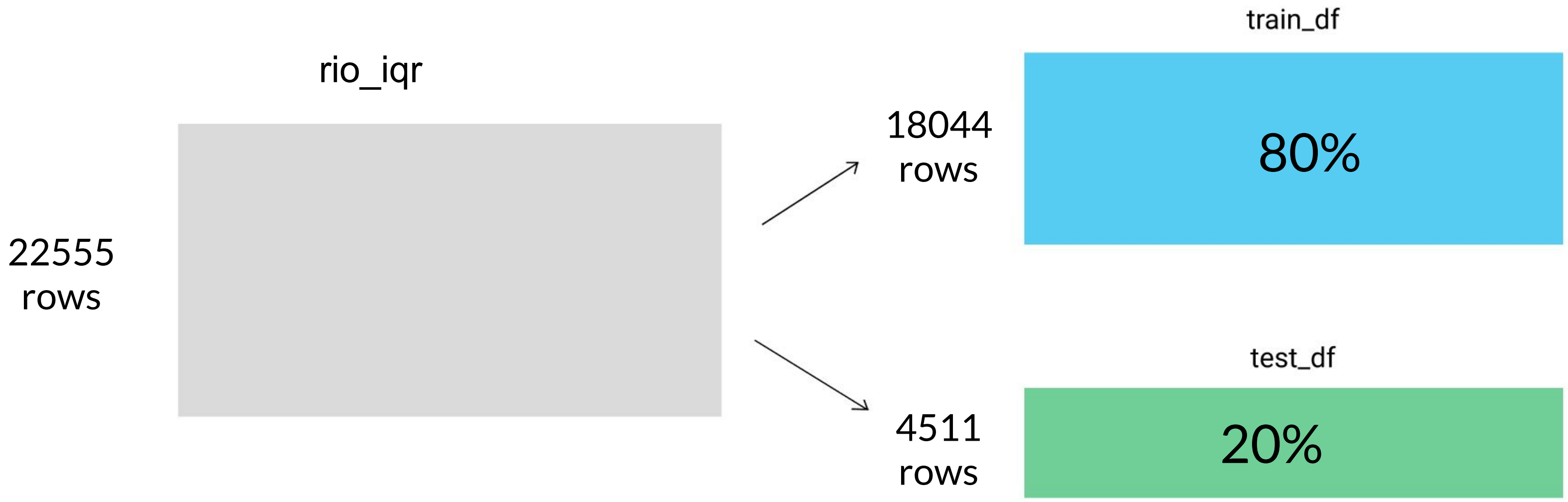


IQR Method

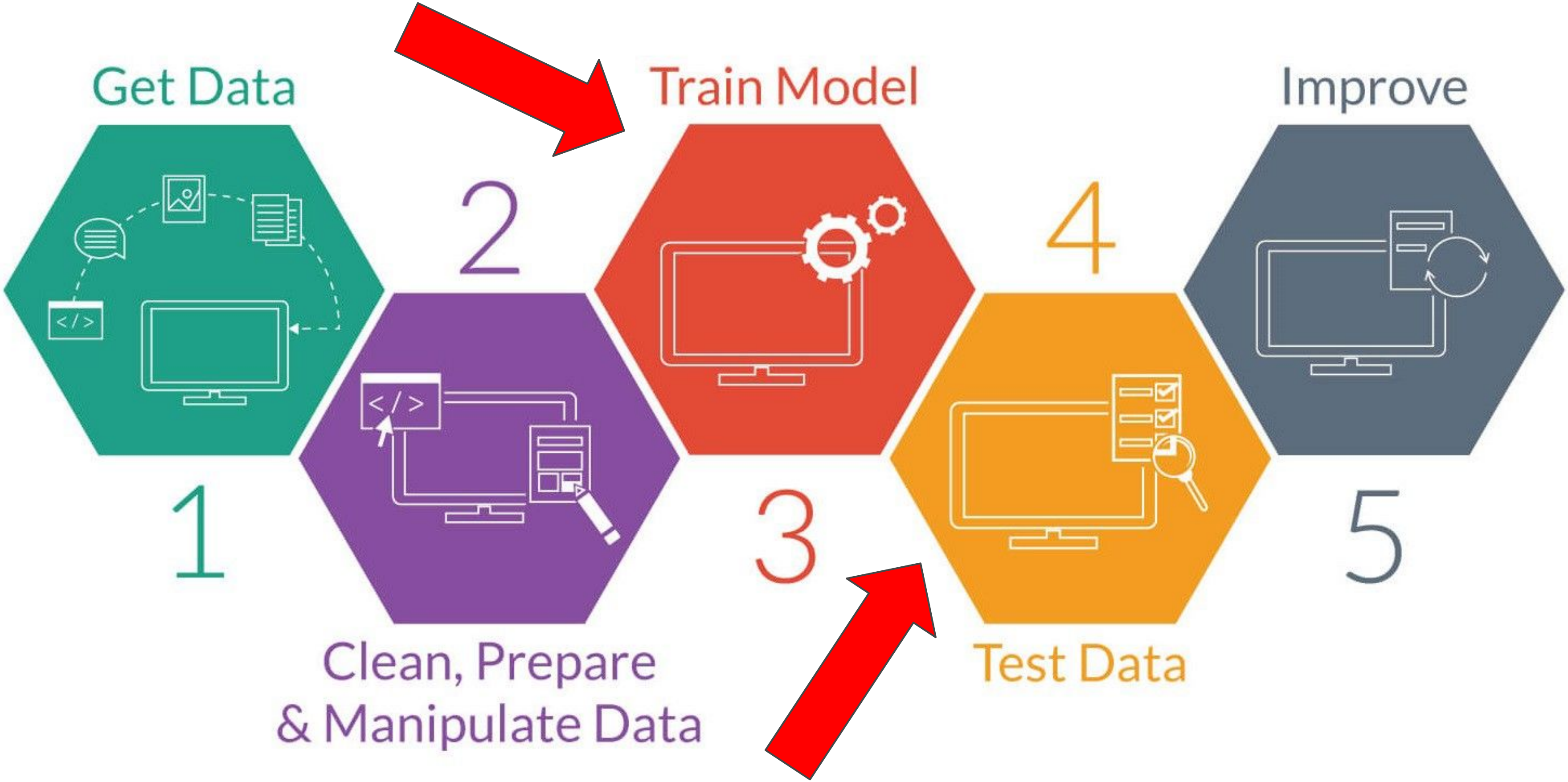


Separate data into Training and Test


```
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

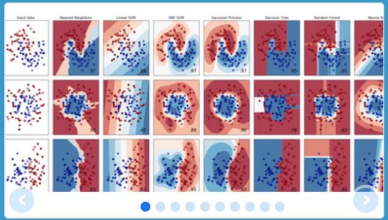


Introduction to scikit-learn

Home | Installation | Documentation | Examples

Custom Search | Search

Fork me on GitHub



scikit-learn

Machine Learning in Python

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

Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.
Algorithms: SVM, nearest neighbors, random forest, …

— Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, ridge regression, Lasso, …

— Examples

Clustering

Automatic grouping of similar objects into sets.

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

— Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: PCA, feature selection, non-negative matrix factorization.

— Examples

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning
Modules: grid search, cross validation, metrics.

— Examples

Preprocessing

Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.
Modules: preprocessing, feature extraction.

— 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))
```

249.380341654696

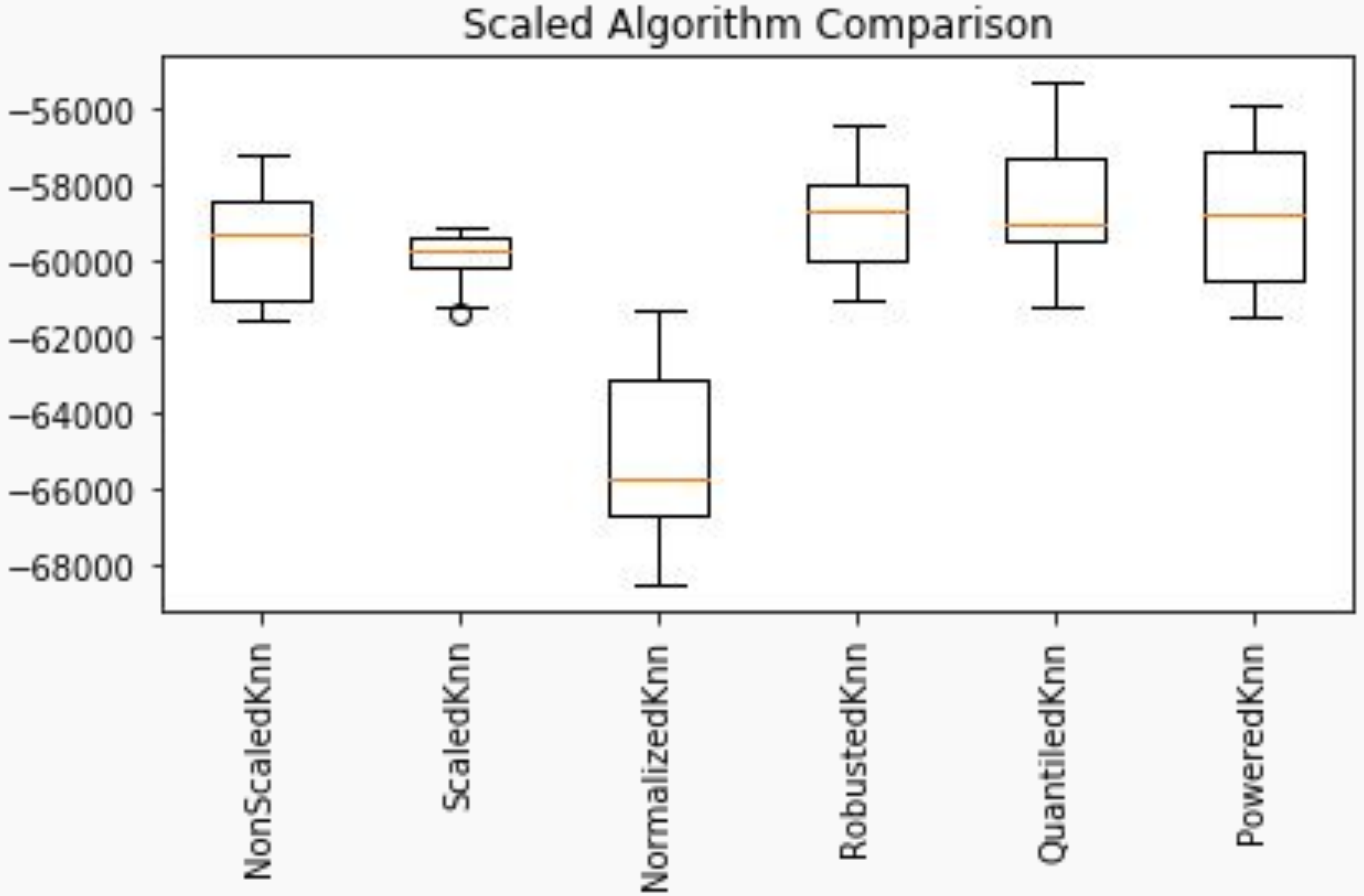
K-Fold Cross Validation

	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		

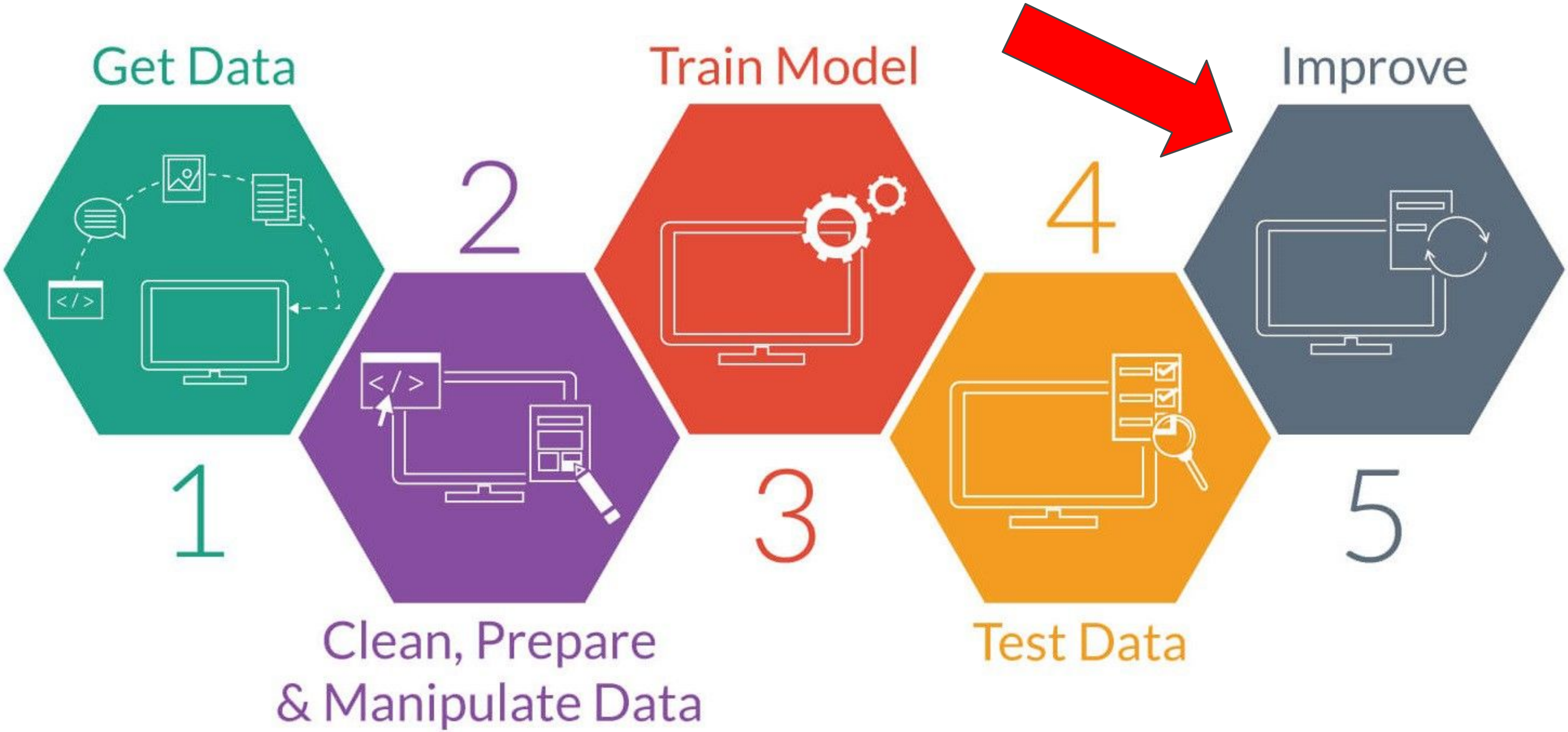
```python
# Test options and evaluation metric
num_folds = 10
seed = 7
scoring = 'neg_mean_squared_error'

# 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))])))
```

```python
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: 244.039159 Std: 38.763743
ScaledKnn Mean: 244.891344 Std: 27.143261
NormalizedKnn Mean: 254.960235 Std: 47.674465
RobustedKnn Mean: 242.588604 Std: 37.084763
QuantiledKnn Mean: 241.832053 Std: 40.831295
PoweredKnn Mean: 242.609898 Std: 43.641614
```


Scaled Algorithm Comparison
−56000
−58000
−60000
−62000
−64000
−66000
−68000
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 = RobustScaler().fit_transform(X_train)

# instantiate a model
model = KNeighborsRegressor()

# Test options and evaluation metric
num_folds = 10
seed = 20
scoring = 'neg_mean_squared_error'

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

grid_result = grid.fit(scaler, Y_train)
```

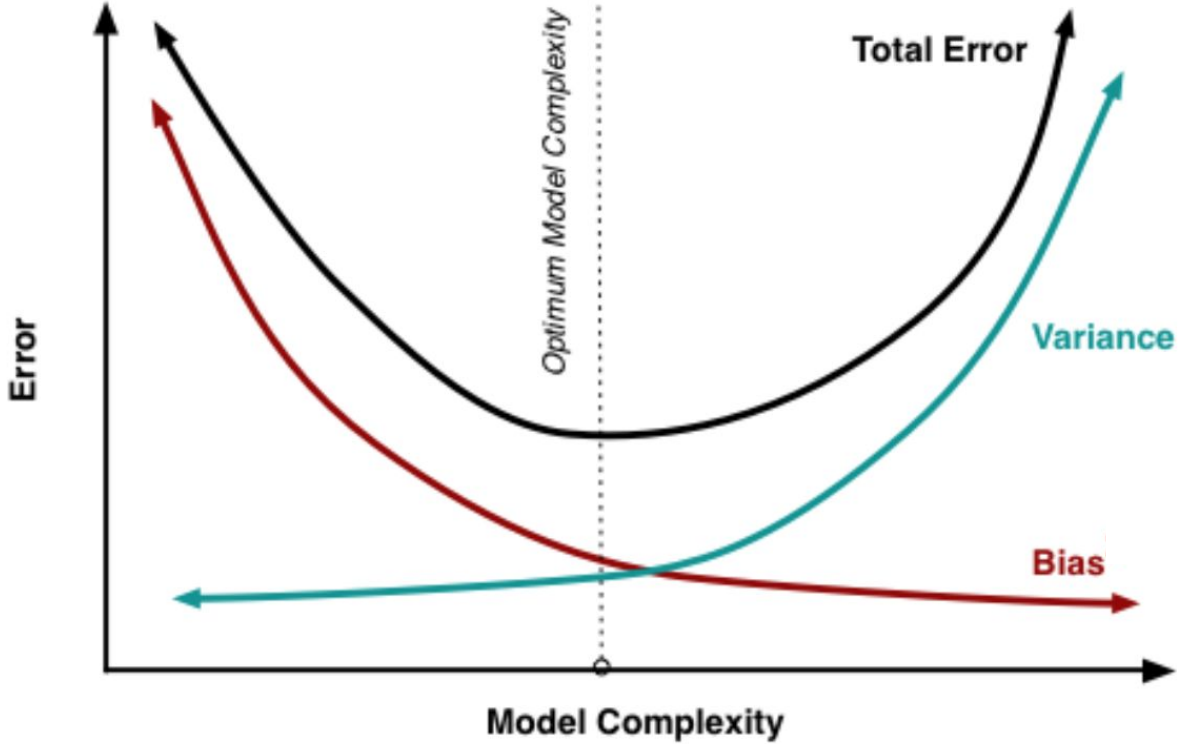
```
Best: 229.305192 using {'n_neighbors': 21}
309.371877 (70.236440) with: {'n_neighbors': 1}
254.773217 (45.432194) with: {'n_neighbors': 3}
242.588715 (37.086348) with: {'n_neighbors': 5}
237.486119 (35.105753) with: {'n_neighbors': 7}
234.829511 (33.954766) with: {'n_neighbors': 9}
232.982061 (29.816265) with: {'n_neighbors': 11}
231.647881 (27.656716) with: {'n_neighbors': 13}
230.992442 (27.038567) with: {'n_neighbors': 15}
230.429167 (28.233139) with: {'n_neighbors': 17}
229.797742 (29.420696) with: {'n_neighbors': 19}
229.305192 (29.403179) with: {'n_neighbors': 21}
```

Finalize Model

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

232.8971868832462

Bias-Variance Tradeoff




MacBook Pro