UFRN

Open Data Day
Natal, Brazil

01 a 03 de março de 2018

Minicourse

Machine Learning Fundamentals in Python - Improve the Accuracy & Scikit-Learn

Speaker:
Prof. Ivanovitch Silva (ivan@imd.ufrn.br)

Improve the accuracy (multivariate knn)

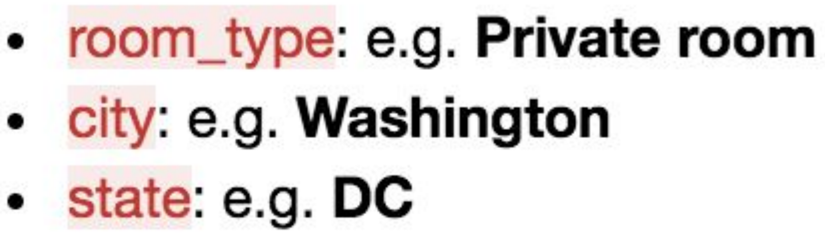
- Increase the number of attributes the model uses to calculate similarity when ranking the closest neighbors
- Increase k, the number of nearby neighbors the model uses when computing the prediction

Improve the accuracy (multivariate knn)

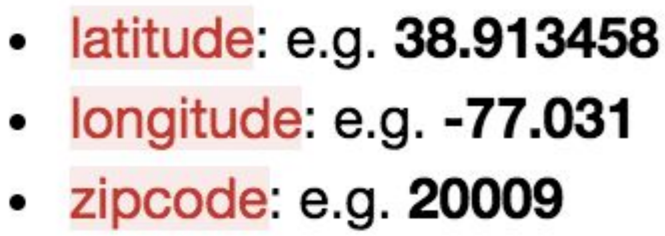
When selecting more attributes to use in the model, we need to watch out for columns that don't work well with the distance equation.

- non-numerical values (e.g. city or state)
  - Euclidean distance equation expects numerical values
- missing values
  - distance equation expects a value for each observation and attribute
- non-ordinal values (e.g. latitude or longitude)
  - ranking by Euclidean distance doesn't make sense if all attributes aren't ordinal

Removing features



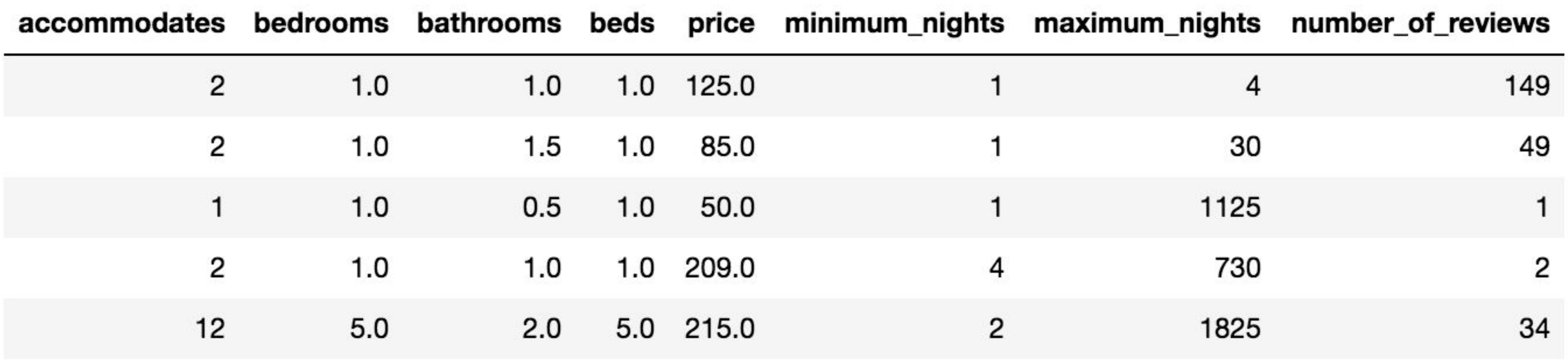
- room_type: e.g. **Private room**
- city: e.g. **Washington**
- state: e.g. **DC**



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

- host_response_rate
- host_acceptance_rate
- host_listings_count

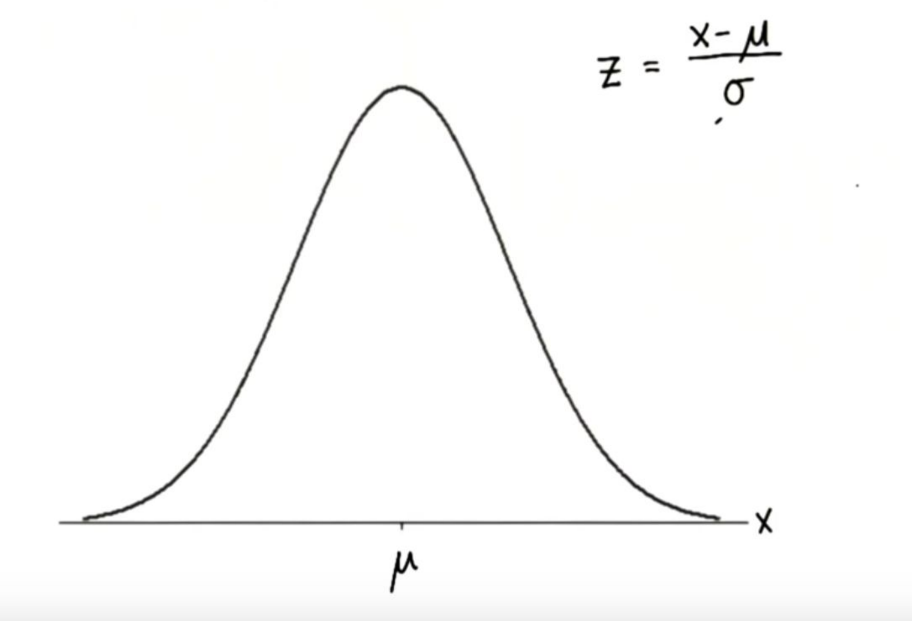
Normalize columns

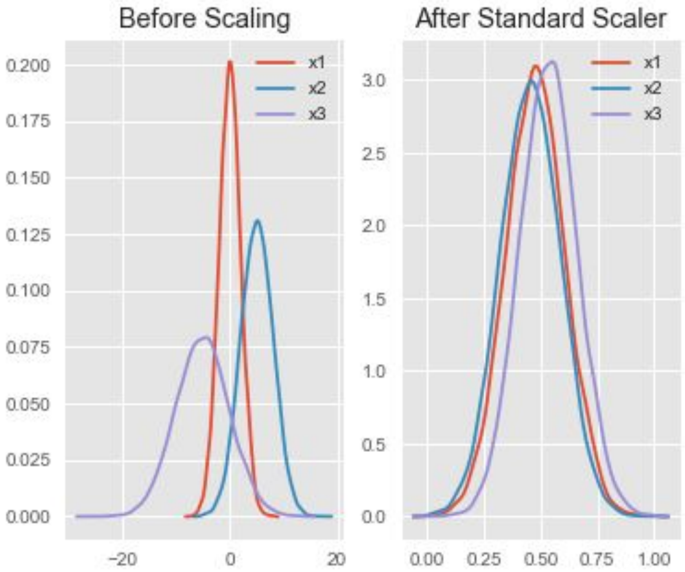


accommodates	bedrooms	bathrooms	beds	price	minimum_nights	maximum_nights	number_of_reviews
2	1.0	1.0	1.0	125.0	1	4	149
2	1.0	1.5	1.0	85.0	1	30	49
1	1.0	0.5	1.0	50.0	1	1125	1
2	1.0	1.0	1.0	209.0	4	730	2
12	5.0	2.0	5.0	215.0	2	1825	34

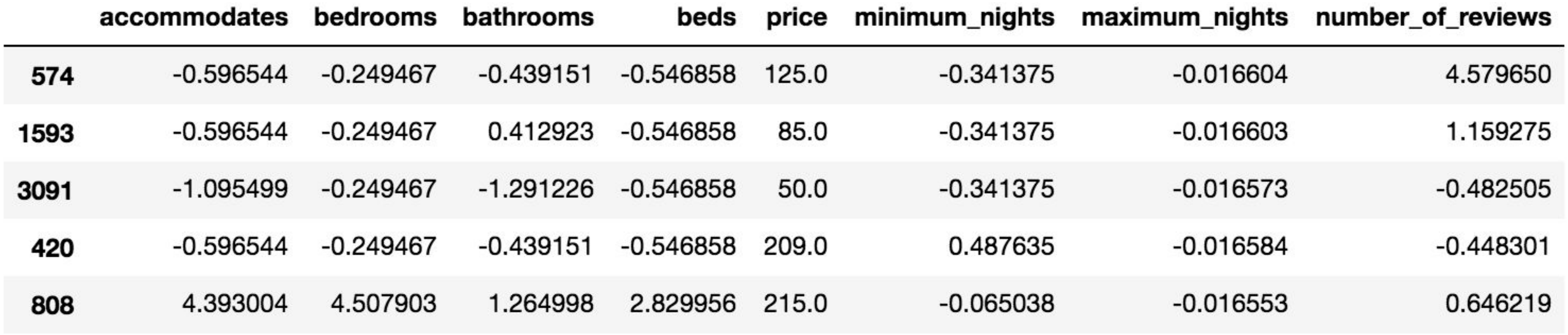
Normalize columns

```
normalized_listings = (dc_listings - dc_listings.mean()) / (dc_listings.std())
```





Normalize columns



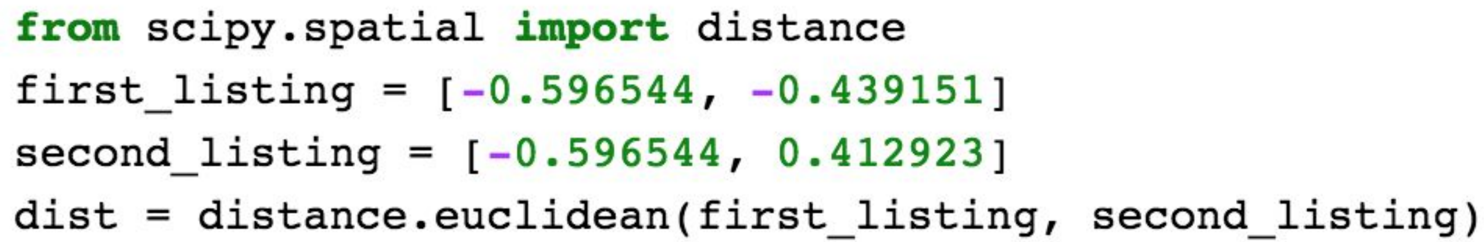
	accommodates	bedrooms	bathrooms	beds	price	minimum_nights	maximum_nights	number_of_reviews
574	-0.596544	-0.249467	-0.439151	-0.546858	125.0	-0.341375	-0.016604	4.579650
1593	-0.596544	-0.249467	0.412923	-0.546858	85.0	-0.341375	-0.016603	1.159275
3091	-1.095499	-0.249467	-1.291226	-0.546858	50.0	-0.341375	-0.016573	-0.482505
420	-0.596544	-0.249467	-0.439151	-0.546858	209.0	0.487635	-0.016584	-0.448301
808	4.393004	4.507903	1.264998	2.829956	215.0	-0.065038	-0.016553	0.646219

Euclidean distance for multivariate case

accommodates	bathrooms
-0.596544	-0.439151
-0.596544	0.412923

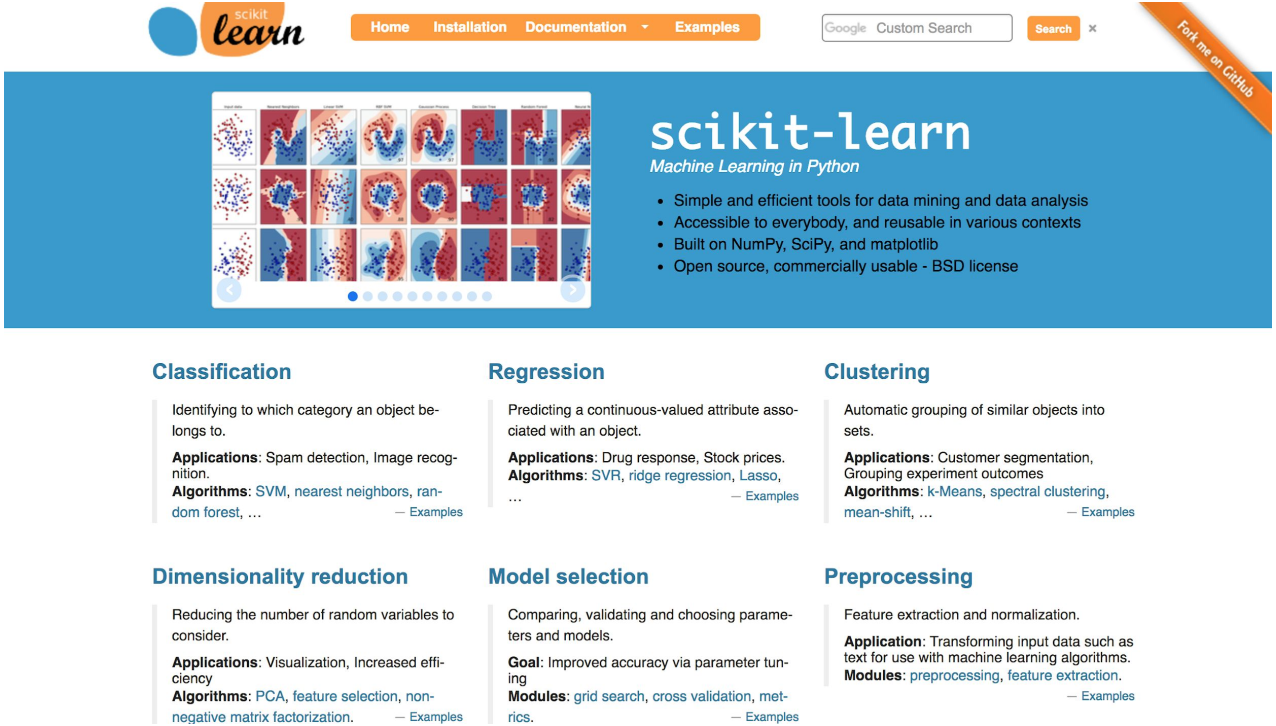
$(q_1 - p_1) + (q_2 - p_2) + \ldots + (q_n - p_n)$	differences	$(-0.596544 + 0.596544) + (-0.439151 - 0.412923)$
$(q_1 - p_1)^2 + (q_2 - p_2)^2 + \ldots + (q_n - p_n)^2$	squared differences	$(0)^2 + (-0.852074)^2$
$\sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \ldots + (q_n - p_n)^2}$	Euclidean distance	$= \sqrt{0 + 0.72603}$ $= 0.852074$

Euclidean distance for multivariate case



```
from scipy.spatial import distance
first_listing = [-0.596544, -0.439151]
second_listing = [-0.596544, 0.412923]
dist = distance.euclidean(first_listing, second_listing)
```

Introduction to scikit-learn



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.

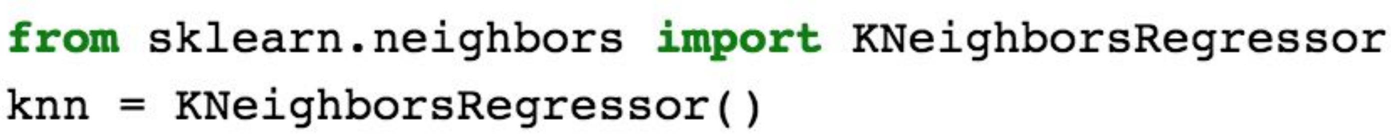
Application: Transforming input data such as text for use with machine learning algorithms.
Modules: preprocessing, feature extraction. — Examples

Scikit-learn workflow

The scikit-learn workflow consists of 4 main steps:

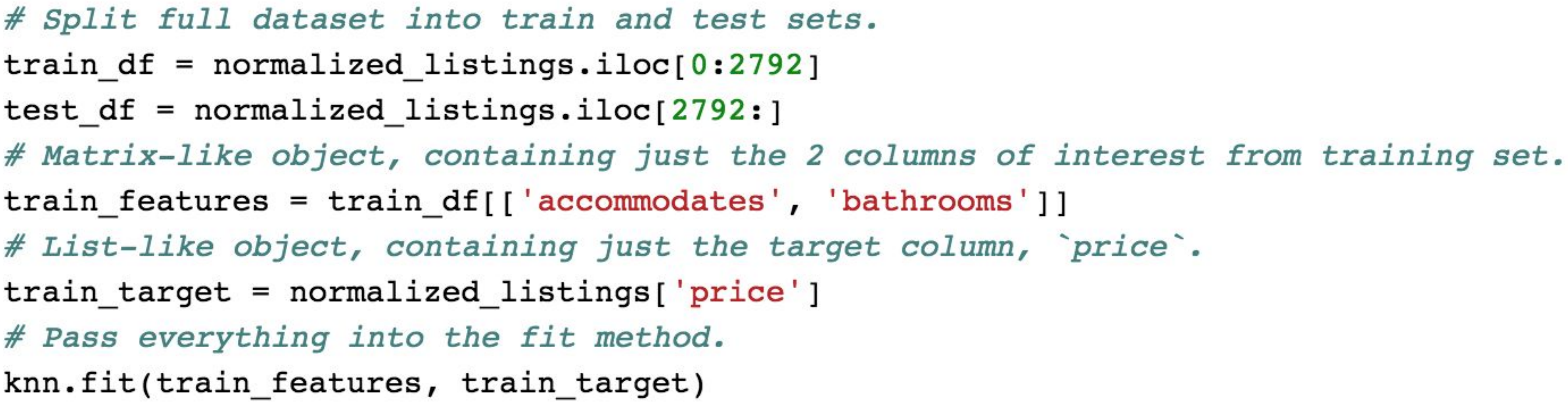
- instantiate the specific machine learning model you want to use
- fit the model to the training data
- use the model to make predictions
- evaluate the accuracy of the predictions

Instantiate the machine learning model



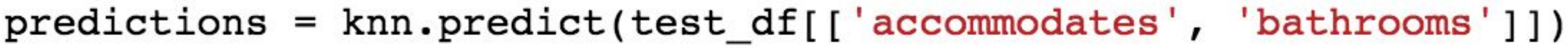
```
from sklearn.neighbors import KNeighborsRegressor
knn = KNeighborsRegressor()
```

Fitting the model to the training data



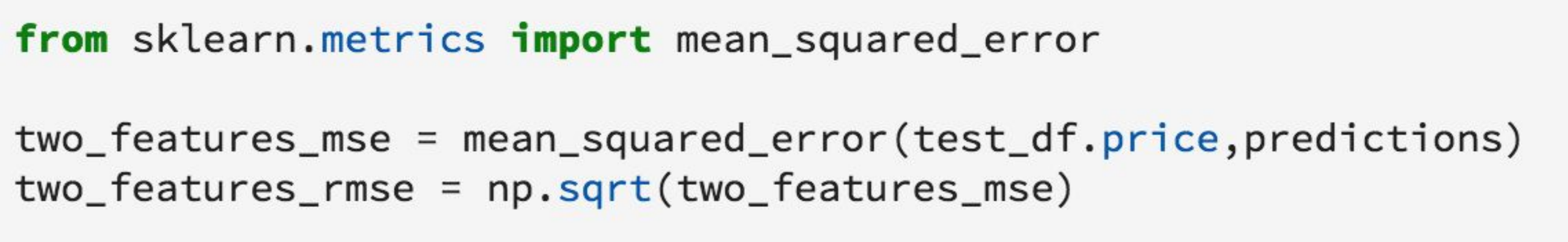
```
# Split full dataset into train and test sets.
train_df = normalized_listings.iloc[0:2792]
test_df = normalized_listings.iloc[2792:]
# Matrix-like object, containing just the 2 columns of interest from training set.
train_features = train_df[['accommodates', 'bathrooms']]
# List-like object, containing just the target column, `price`.
train_target = normalized_listings['price']
# Pass everything into the fit method.
knn.fit(train_features, train_target)
```

Make predictions



```
predictions = knn.predict(test_df[['accommodates', 'bathrooms']])
```

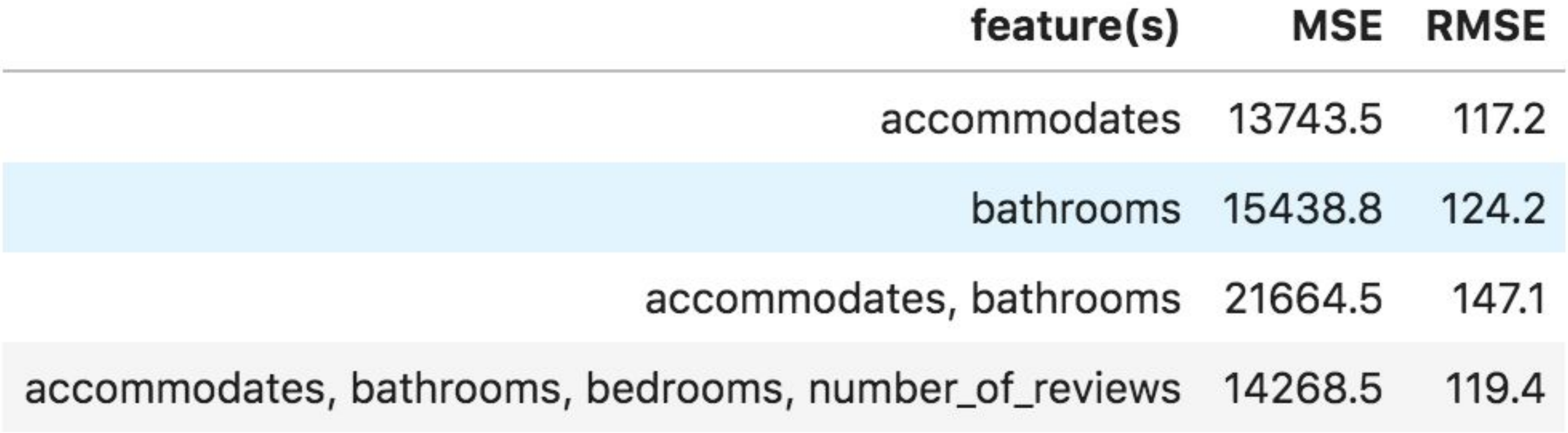
Calculating MSE/RMSE using scikit-learn



```
from sklearn.metrics import mean_squared_error

two_features_mse = mean_squared_error(test_df.price,predictions)
two_features_rmse = np.sqrt(two_features_mse)
```

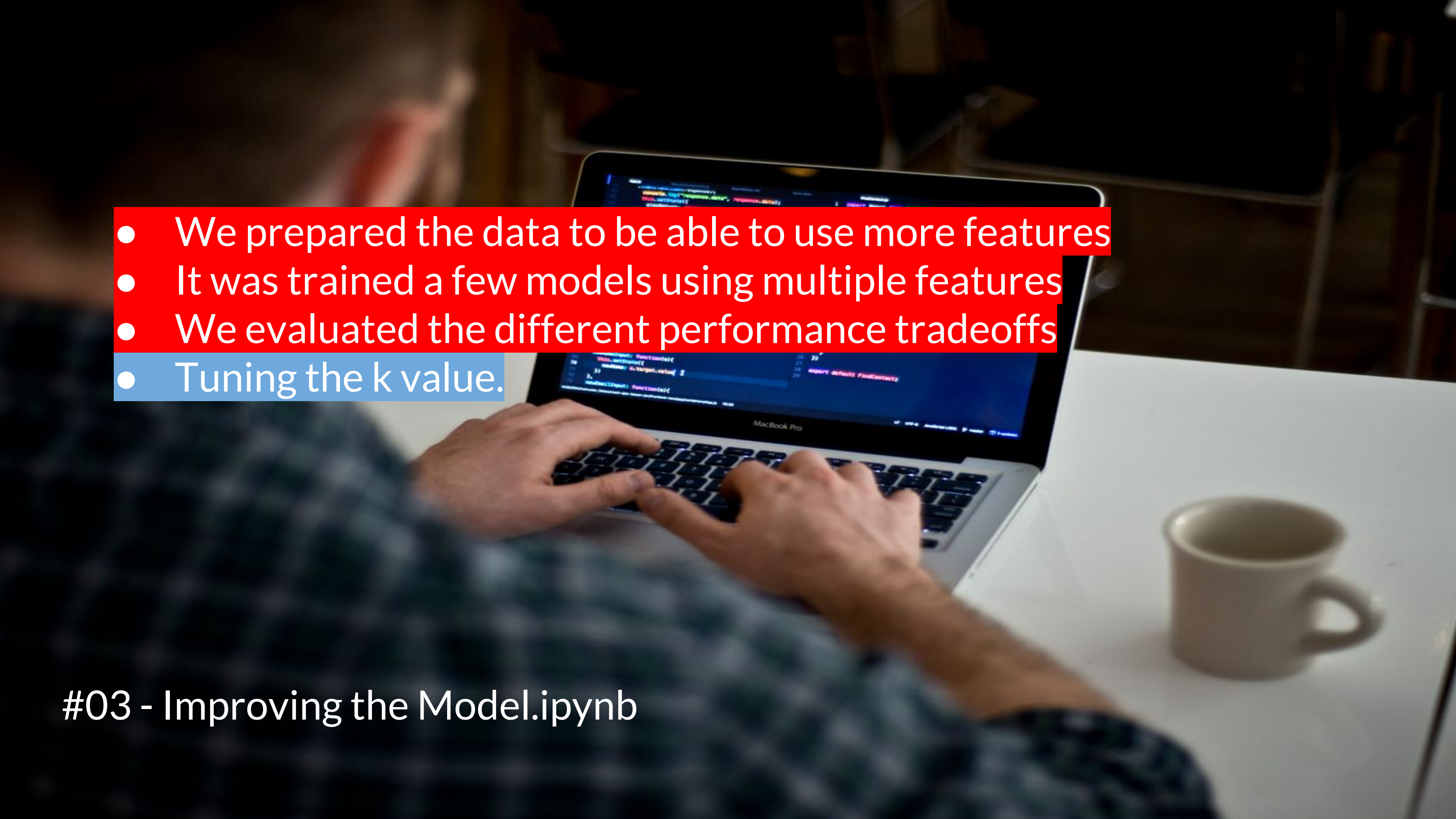
Using more features



feature(s)	MSE	RMSE
accommodates	13743.5	117.2
bathrooms	15438.8	124.2
accommodates, bathrooms	21664.5	147.1
accommodates, bathrooms, bedrooms, number_of_reviews	14268.5	119.4

Using all features

????



- We prepared the data to be able to use more features
- It was trained a few models using multiple features
- We evaluated the different performance tradeoffs
- Tuning the k value.

#03 - Improving the Model.ipynb