Bike-Sharing Rental Demand Estimation

- Team members: Jan Faulstich, Skyler MacGowan, Yannik Suhre, Sebastian Sydow, Jacob Umland

- Presentation Date: November 14, 2020

Frankfurt School of Finance & Management

German Excellence. Global Relevance.

1 - Intro & Techstack

2 - The Dataset

3 - Frameworks

4 - Implementation

5 - Live Demo!

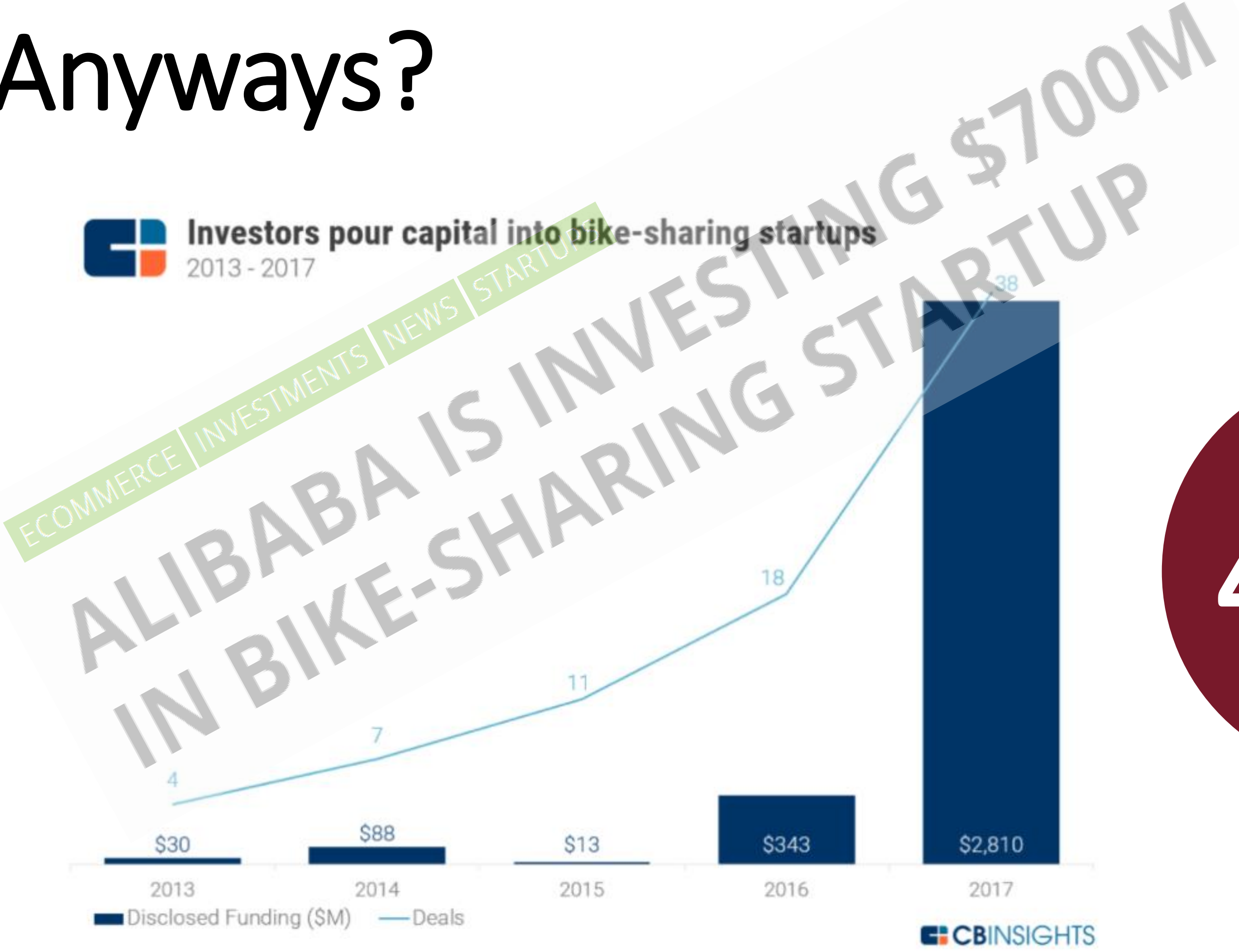
The Project

- **Objective**: build a predictive model that estimates the number of bike rentals within a specified hour-long timeframe for Washington D.C.'s *Capital Bikeshare*.
- **Dataset**: multivariate time series data with different variables one would expect to impact the demand for bike-share rentals, largely environmental and seasonal factors.
- **Timeframe**: January 1st 2011 to December 31st 2012, broken down by hour (each row represents an hour).

3

What is Bike-sharing Anyways?

- For a small fee, users rent a bike for a short period of time, usually less than 30 minutes.
- Typically these take the form of private initiatives, though at least some degree of collaboration with municipal authorities is common.



Fun fact: *as of this past July, Google Maps has started incorporating bike share stations in its route planner!*

Historical Context: Luud Schimmelpennik

- **Witkar**: limited scale but proof of concept despite very rudimentary technology and no political backing.

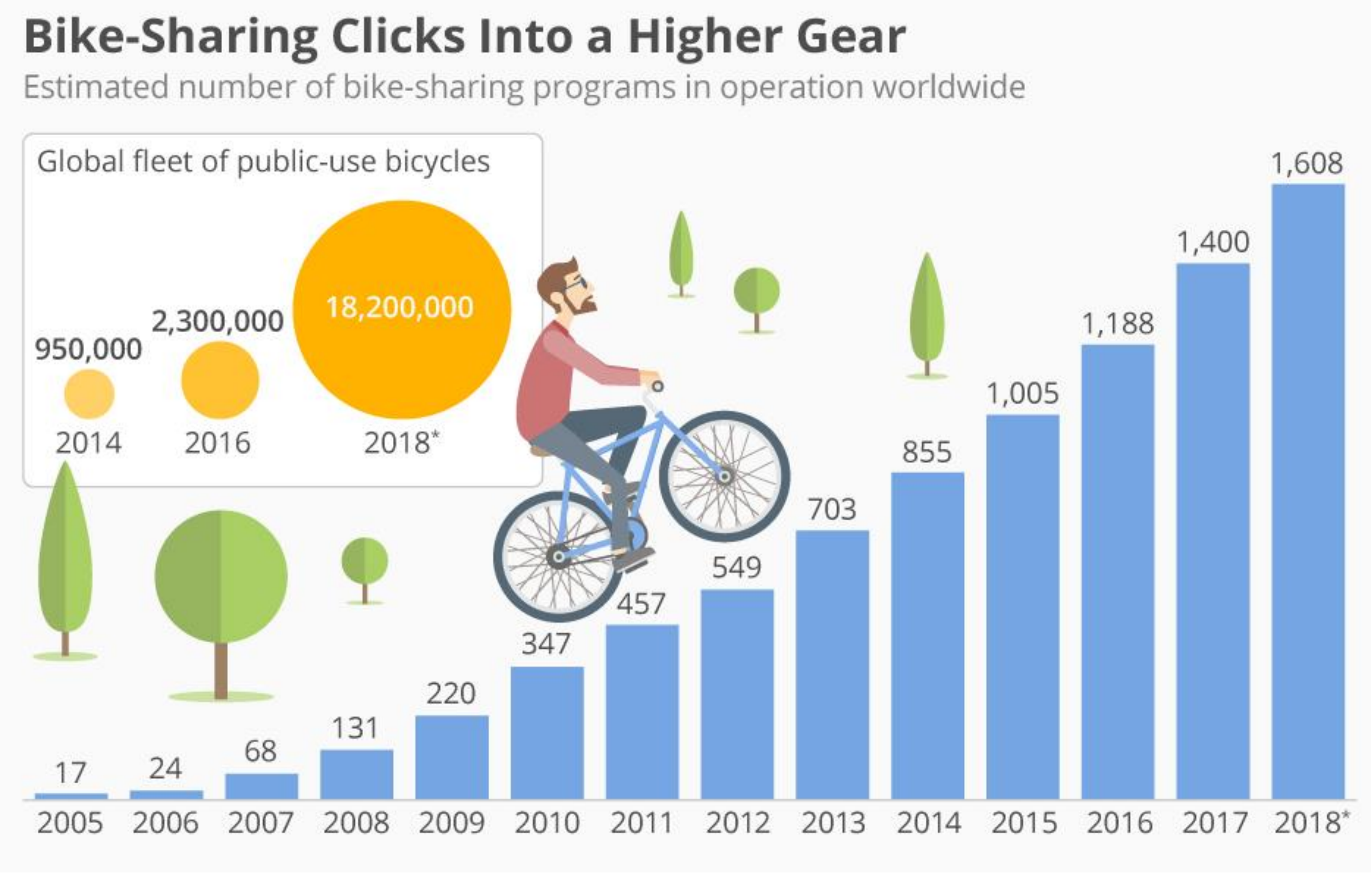
Historical Context: Luud Schimmelpennik

- Following his significant role in launching Copenhagen’s first bike-sharing system in the late 1990s, Schimmelpennik subsequently played a prominent role in the successful implementation of the first such systems in Vienna and Lyon.
- **The Game Changer**: Schimmelpennik then went on to lead the development of the *Vélib* bike-sharing system in Paris, which went live in 2007. This was an unprecedented success that catalysed a significant increase in the number of bike-sharing programs across the globe.

Fun fact: Vélib" is a portmanteau of the words vélo ("bicycle") and liberté ("freedom")!

Why were we interested in this project specifically?

- A few main factors:
  - ➢The sharing economy
  - ➢Nature of bike-sharing systems as it relates to data analysis
  - ➢Innovative approaches to urban form & design



python

asana

Overleaf

The Dataset – Basic Info & Statistics

Dataset statistics

Number of variables	17
Number of observations	17379
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%

	instant	dteday	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	2011-01-01	1	0	1	0	0	6	0	1	0.24	0.2879	0.81	0.0000	3	13	16
1	2	2011-01-01	1	0	1	1	0	6	0	1	0.22	0.2727	0.80	0.0000	8	32	40
2	3	2011-01-01	1	0	1	2	0	6	0	1	0.22	0.2727	0.80	0.0000	5	27	32
3	4	2011-01-01	1	0	1	3	0	6	0	1	0.24	0.2879	0.75	0.0000	3	10	13

Imputing NAs

Dataset statistics

Number of variables	17
Number of observations	17379
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%

Where are our NAs?

There they are...

(365 + 366) * 24 = 17544
17544 – 17379 = 165

Imputing NAs

Methods				
mean	forward fill	backward fill	interpolate	rolling window

df.ffill():

df.interpolate():

Imputing NAs

Create new Datetime index

Imputed the time series based on datetime

→ dteday, month, hour, weekday

Use forward fill function

Forward filled for more time series data

→ year, season, holiday, workingday

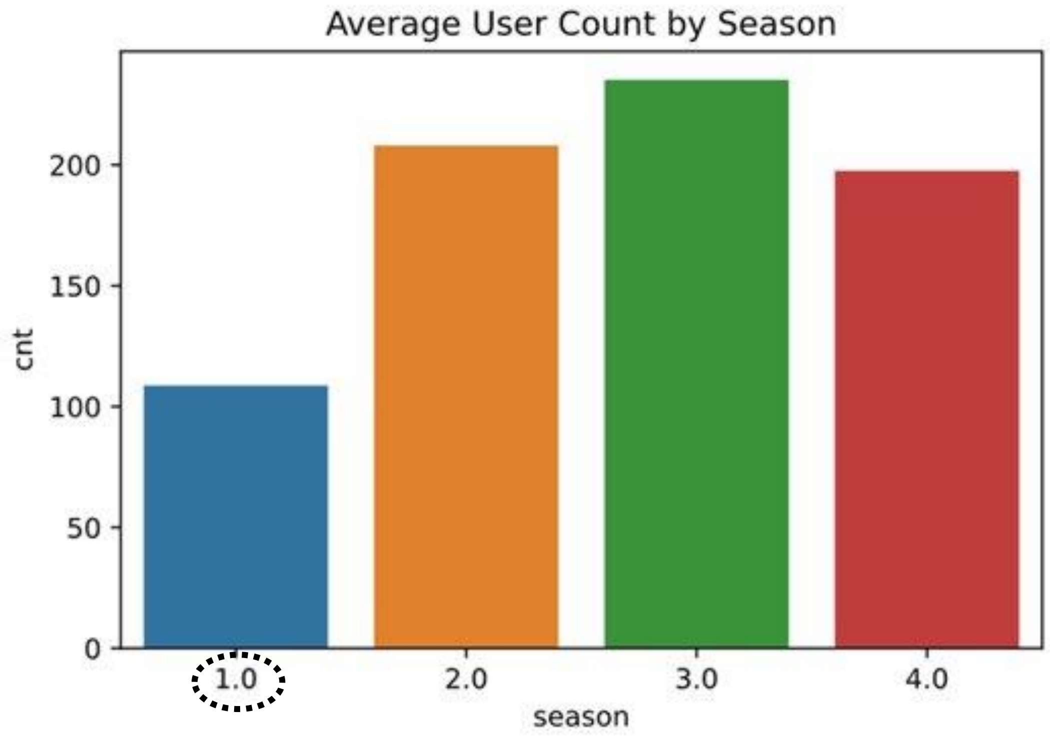
Use interpolate function

Interpoalted the weather data and the count of bike rentals

→ wheathersit, temp, atemp, hum, windspeed, casual, registered, cnt

Columns:

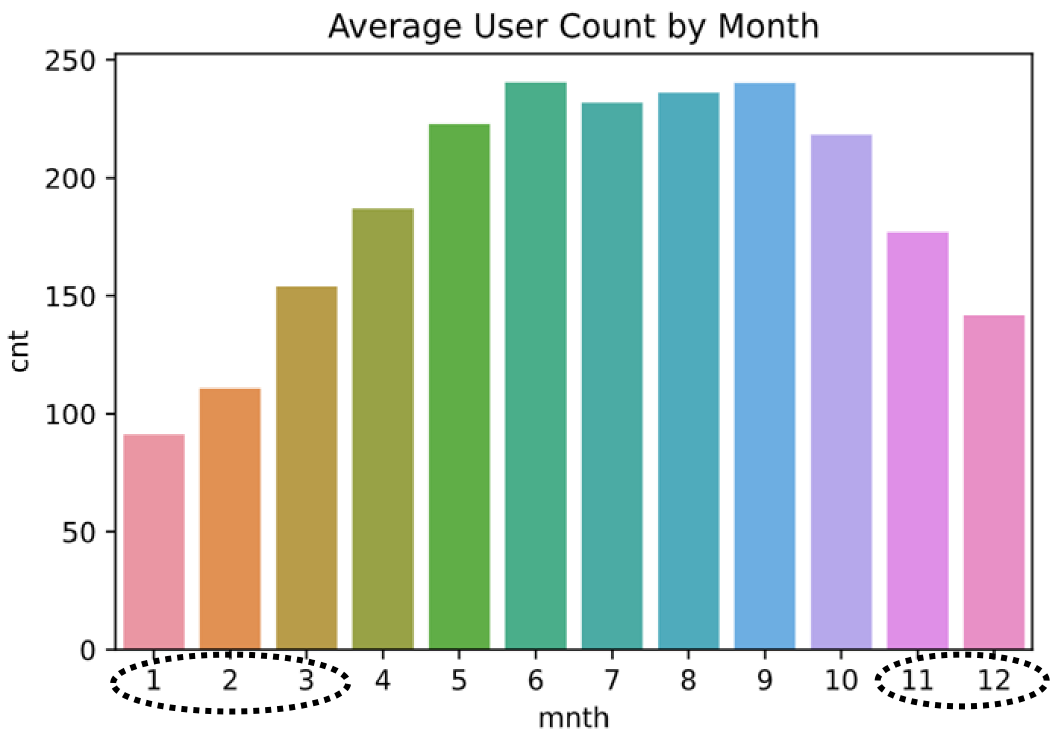
Data Visualization



Bike rentals are lowest in spring

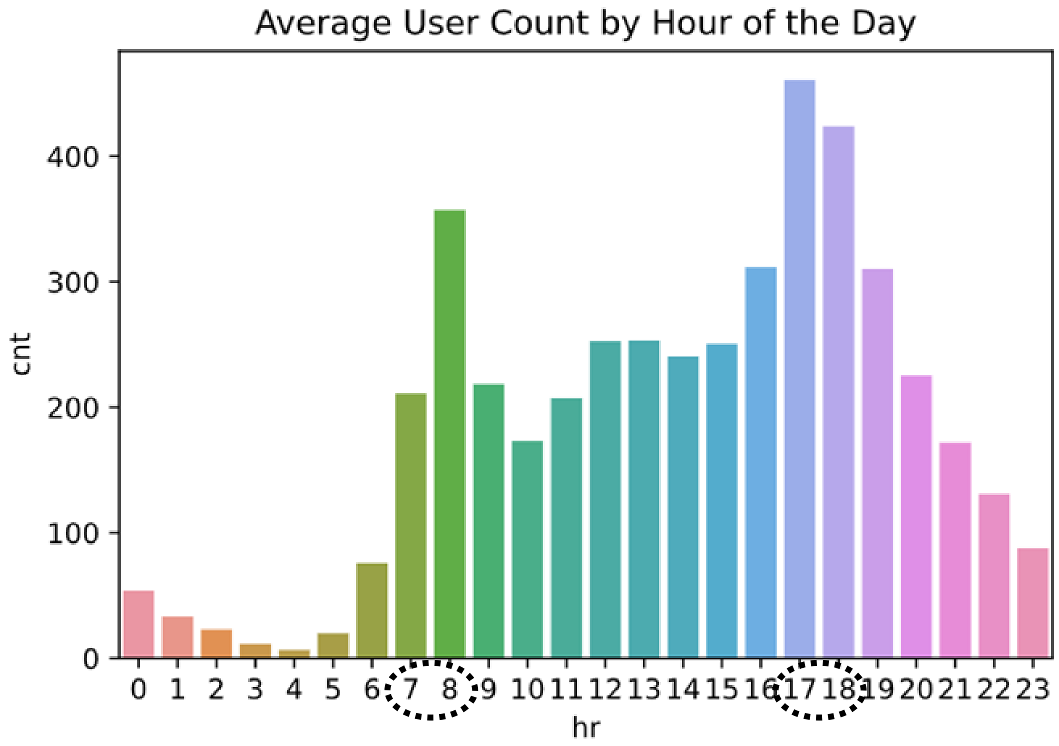
seaborn

Data Visualization



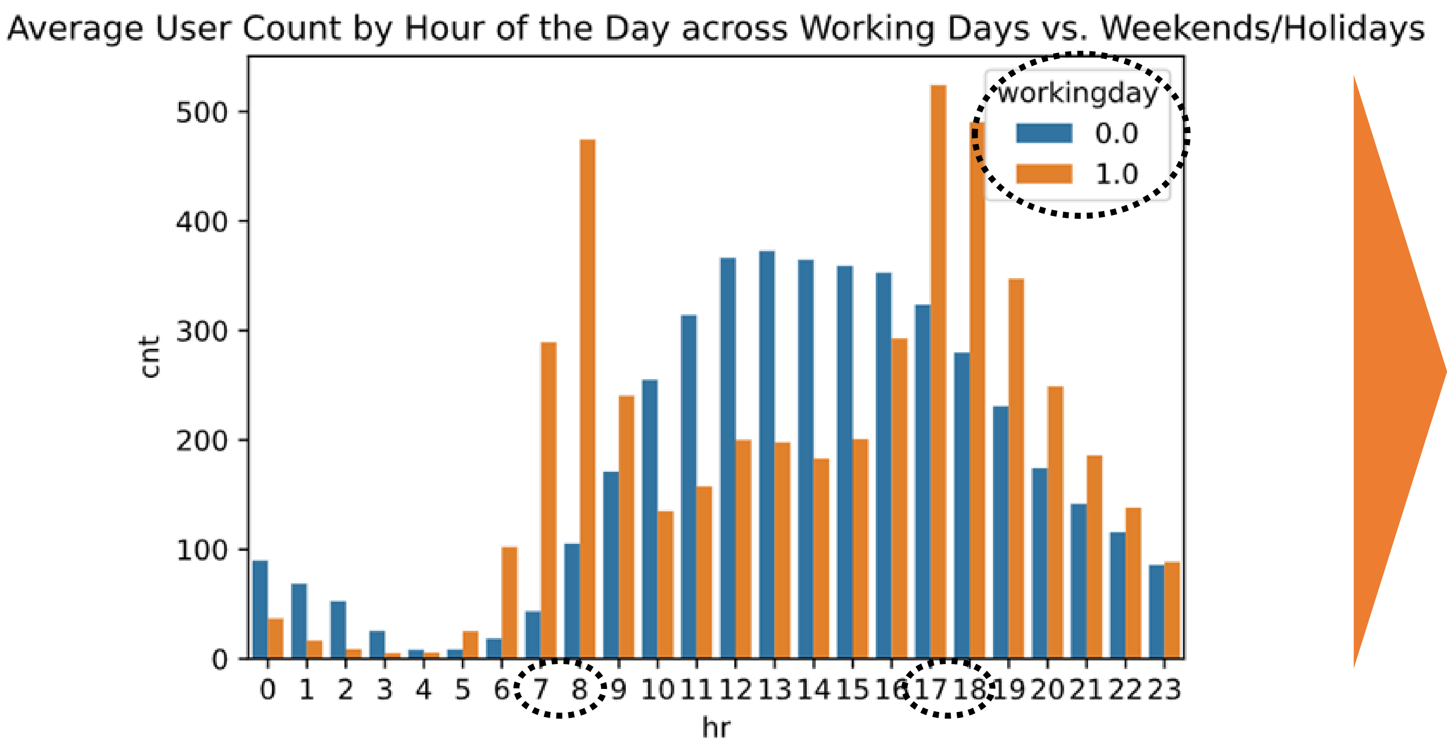
Bike rentals are low at the beginning and end of the year

Data Visualization



Demand is high during the morning and evening rush hours

Data Visualization



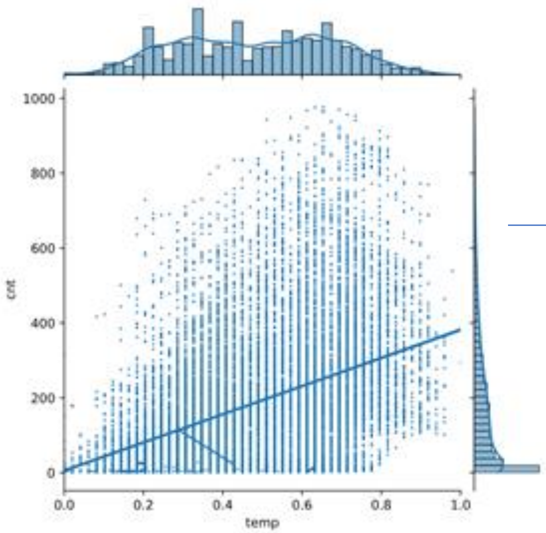
The rush hour effect is only observed on weekdays

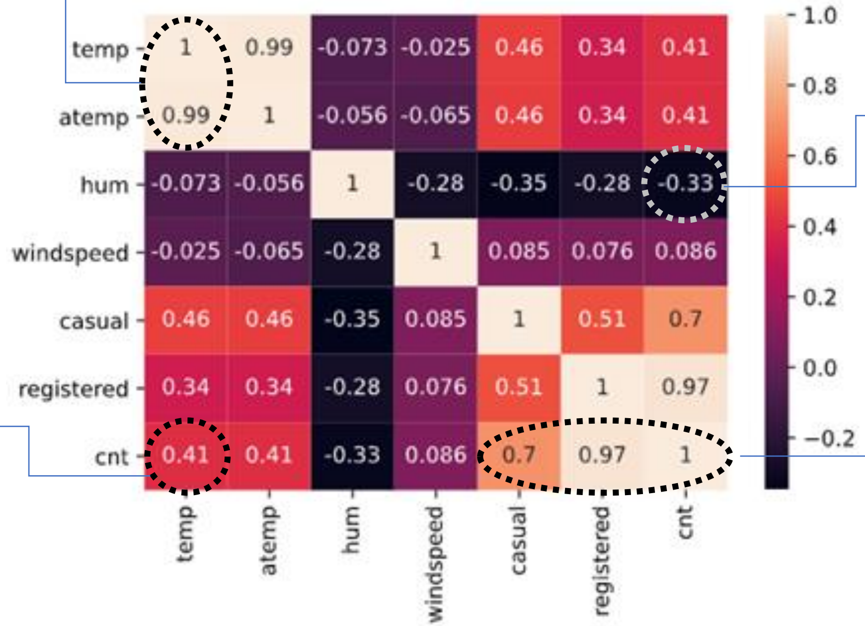
accurate models will most likely account for all these simple observations

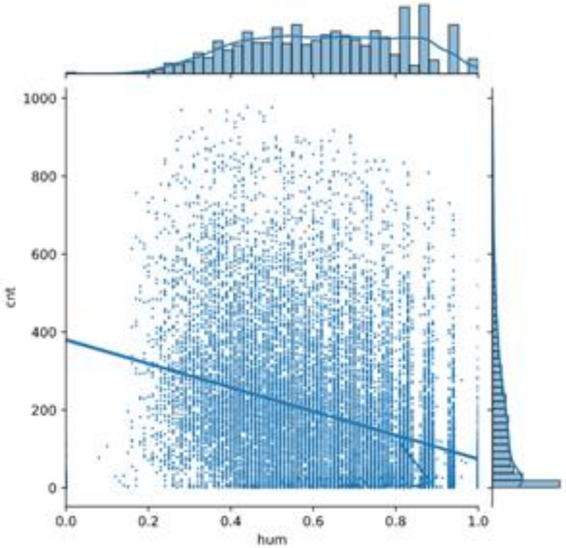
Data Visualization

almost perfect linear correlation:
- no predictive value-add
- curse of dimensionality
- Occam's razor
- multicollinearity

→ drop atemp







high correlation hints at leakage variables:
- casual + registered = cnt

→ drop casual & registered

Dropping Features & Normalization

useless index	instant	workingday	
	datetime	weathersit	
made rudundant by datetime creation	dteday	temp	
	season	atemp	*0.99 correlation with temp*
	yr	hum	
	mnth	windspeed	
	hr	registered	*leakage variable for cnt*
	holiday	casual	*leakage variable for cnt*
	weekday	cnt	

kept feature | dropped feature | *reason*

normalization of continous variables:

- fixing incorrect normalizations of temp & hum
- normalization of cnt (not necessary)

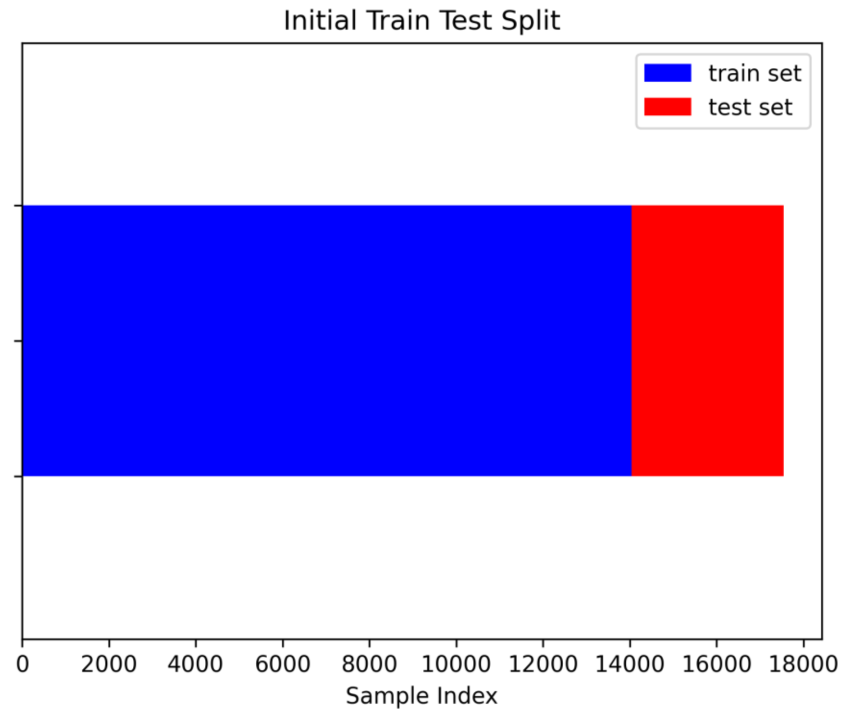
$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Data Partitioning

Key considerations:

- <u>Multivariate time series data</u>
  - ➢ observations in the dataset are not independent
  - ➢ scikit-learn's *train_test_split* does not resemble a situation in a production environment (model on past data to predict the future)

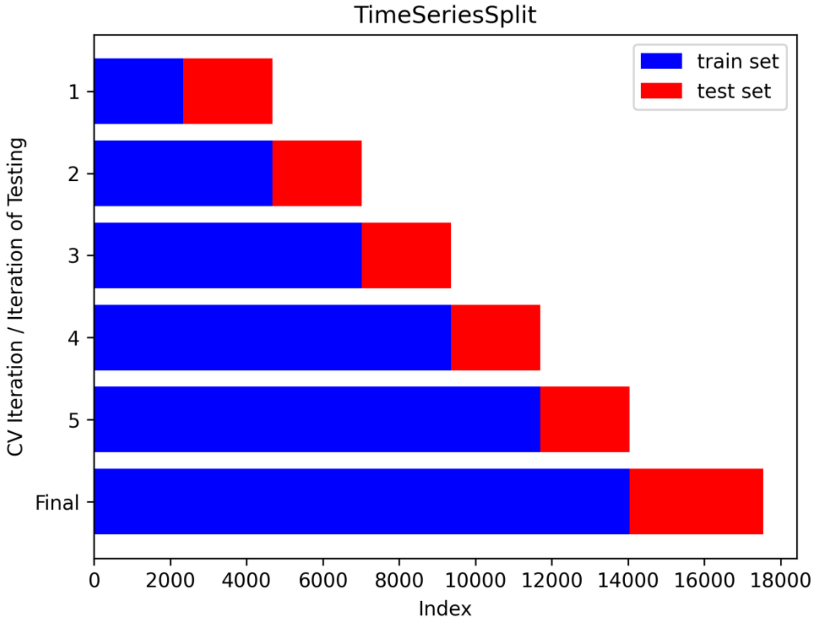
- <u>Our approach:</u>
  - ➢ initial train test split based on time

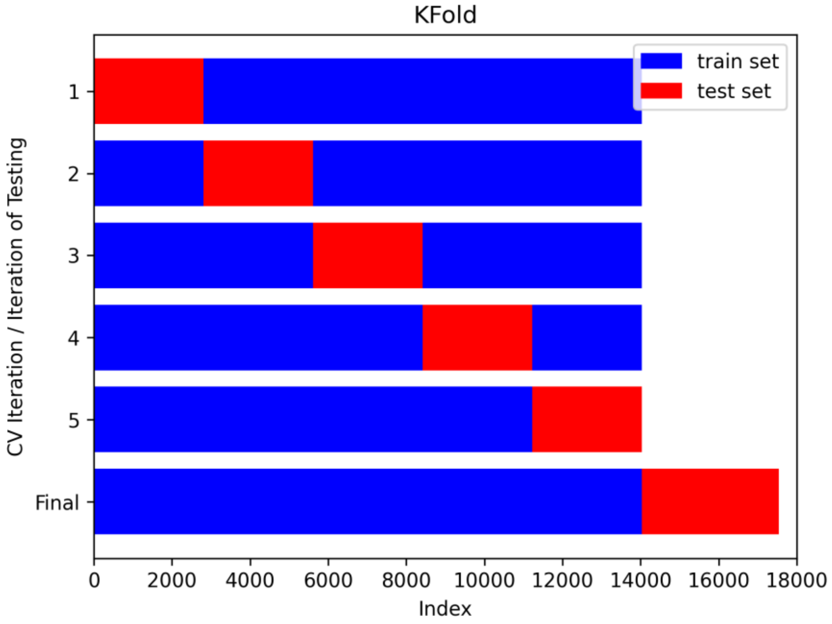


Data Partitioning

Cross Validation

- Two approaches pursued: TimeSeriesSplit and KFold





Frameworks

<u>Rationale</u>

Sktime & Darts[1]: to function properly with multivariate time series data we would have needed to edit our dataset (e.g. through concatenation and/or column ensembling); doing so was not necessary with the frameworks we did select.

TensorFlow: challenging to work with and beyond the scope of what was needed for this project.

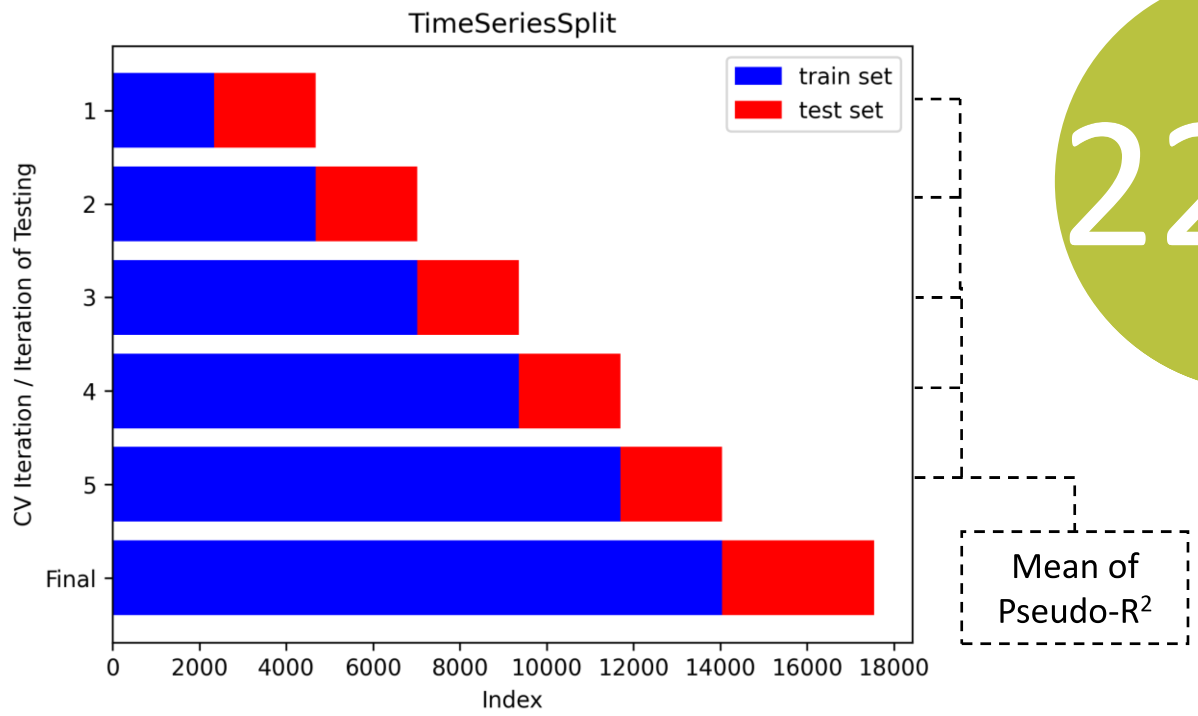
[1]Differentiable Architecture Search.

Used	Not Used
scikit learn	SKTIME
CatBoost	Darts
	TensorFlow

Scikit-Learn - RandomForestRegressor

Modelling Approach:

- Dropping unsupported datetime (→11 input features)
- Cross validation and hyperparameter tuning:
  - TimeSeriesSplit
  - Implemented through cascaded for loops
  - Criterion: mean of Pseudo-R^2 of different hyperparameter combinations across folds

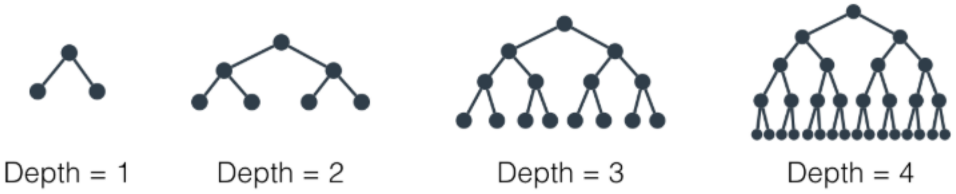


Scikit-Learn - RandomForestRegressor

Applied Hyperparameters:

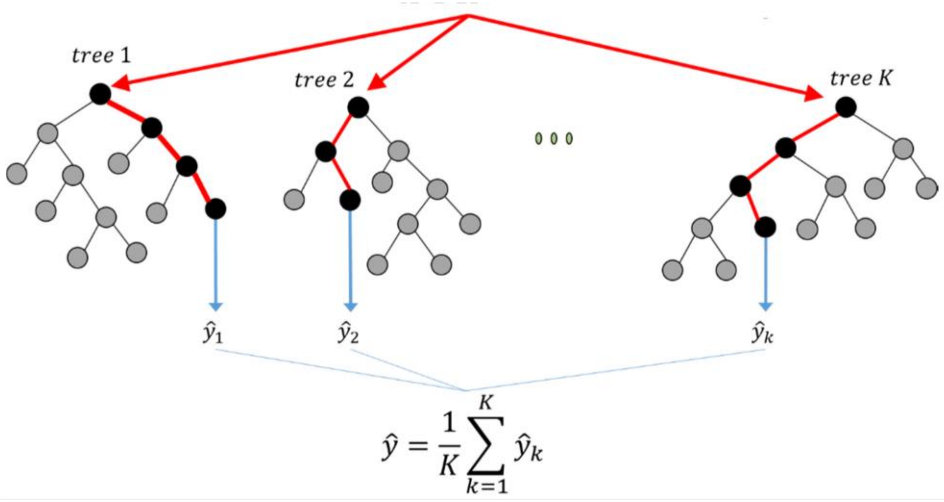
max_depth = 11

➢ maximum depth of each tree in the forest



n_estimators = 300

➢ total number of trees in the forest



Scikit-Learn - RandomForestRegressor

Applied Hyperparameters:

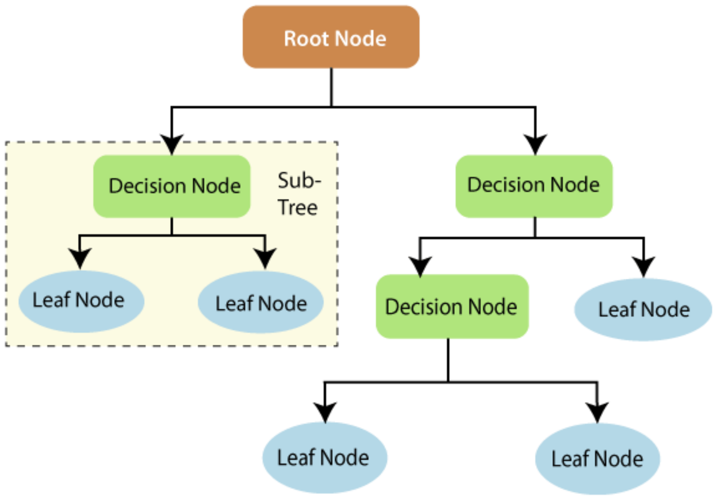
max_features = 10

➢ number of features considered when looking for the best split

- year
- month
- hour
- weathersituation
- temperature
- …

max_leaf_nodes = 80

➢ maximum number of leaf nodes in each tree → limit tree growth



Scikit-Learn - RandomForestRegressor

Summary and Results

CV-Approach
TimeSeriesSplit

Parameters	
max_depth	11
n_estimators	300
max_features	10
max_leaf_nodes	80

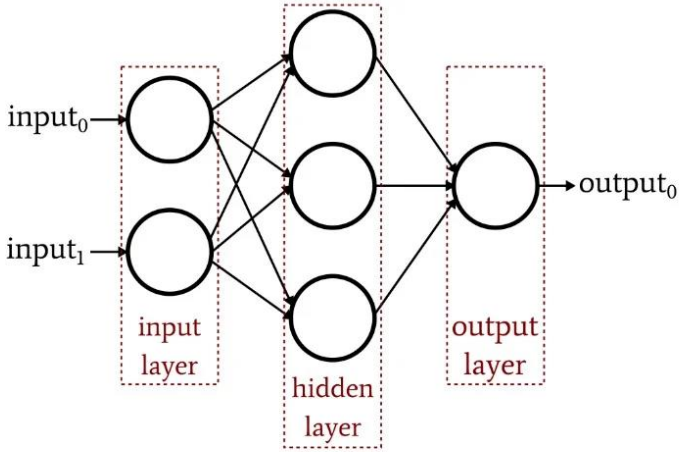
R^2	Pseudo-R^2
0.8979	0.8609

Scikit-Learn - MLPRegressor

MLP Intro and Initialization of MLPRegressor

Mulitlayer Perceptron:

- "feedforward neural network"
- ≥ 3 layers (input, hidden(s), output)



Setting up the Model:

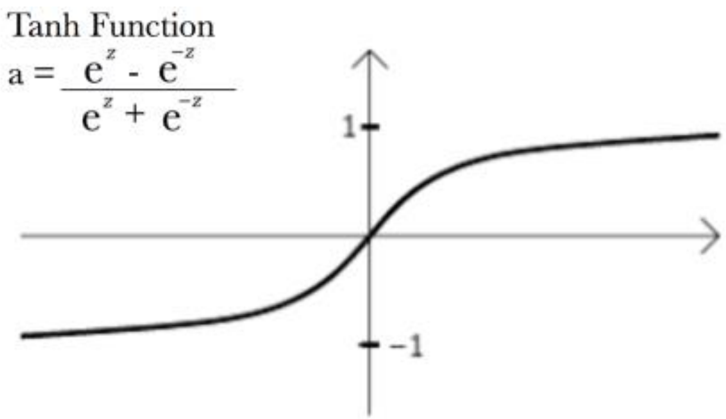
- Dropping unsupported datetime (→11 input features)
- MLPRegressor optimizes the squared-loss
- Solver: LBFGS (does not use learning rate)
- GridSearchCV

Scikit-Learn - MLPRegressor

Applied Hyperparameters

activation function:

- Logistic vs. **Tanh** vs. ReLU



alpha:

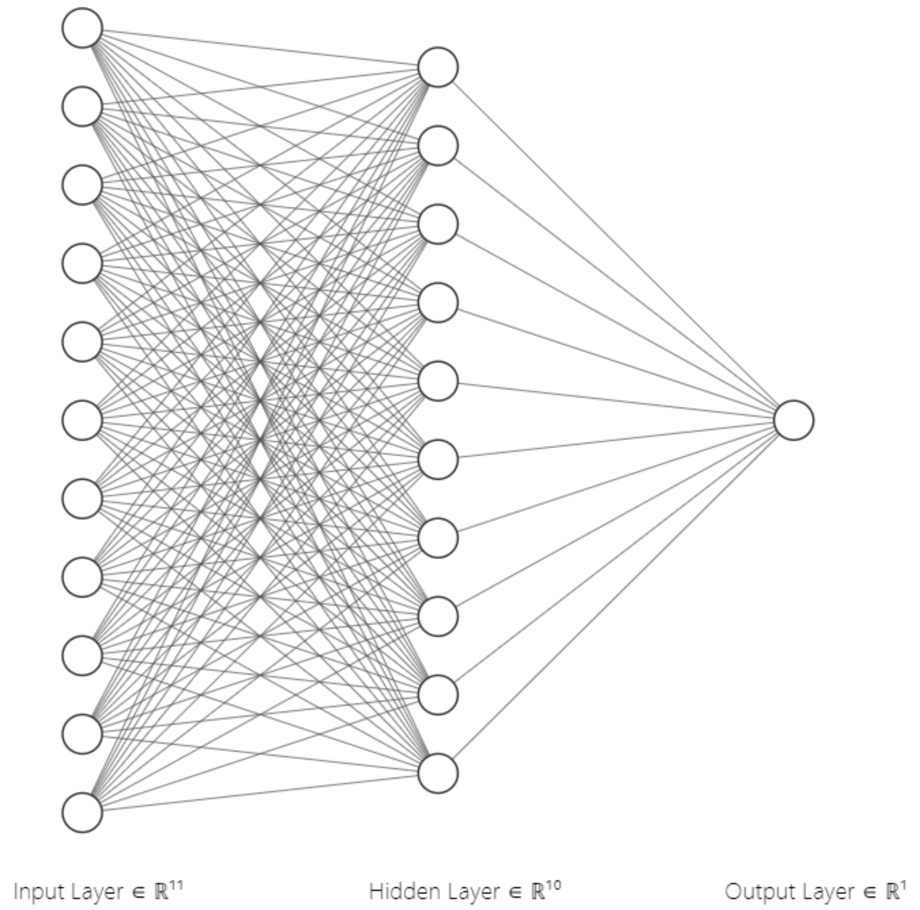
- Regularization/penalty term that combats overfitting by constraining the size of the weights
- Alpha ↗ ⇒ weights ↘ ⇒ overfitting ↘
- **Alpha = 0.1**

Scikit-Learn - MLPRegressor

Hyperparameters Applied

hidden layer size:

- **1 hidden layer**
- **10 neurons**
- Outperformed networks with 2 hidden layers & less neurons



Scikit-Learn - MLPRegressor

Summary and Results

CV-Approach
(Stratified)KFold

Parameters	
activation	tanh
alpha	0.1
hidden layers	1
number of neurons	10

R^2	Pseudo-R^2
0.888642	0.835639

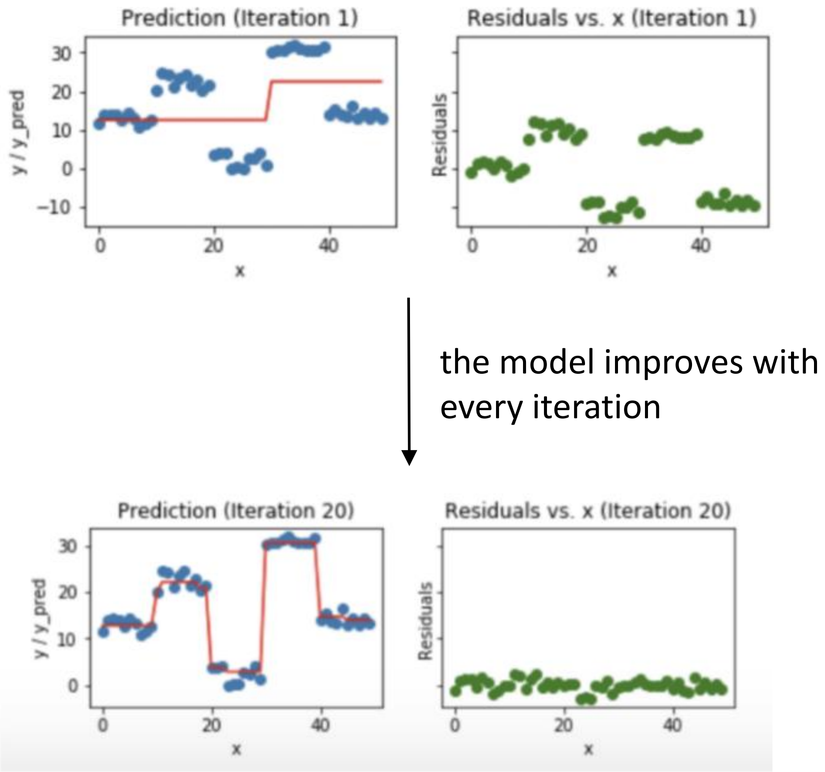
CatBoost Regressor

CatBoost

Gradient boosting on decision trees

Advantages:

1. Categorical feature support
2. Fast prediction
3. Improved accuracy



CatBoost Regressor

Model Training

Step 1: Prepare the dataset

Step 2: Use GridSearch to identify the best parameters

Parameters	
Depth	[6, 8, 10]
Learning rate	[0.01, 0.05, 0.1, 0.2, 0.3]
Iterations	[200, 400, 600, 800, 1000]

Catboost parameters:

Depth: 6
Learning rate: 0.01
Iterations: 1000

CatBoost Regressor

Model Training

<u>Step 3</u>: Fit the model to our training set

CatBoost Regressor	
Loss function	RMSE
Depth	6
Learning rate	0.01
Iterations	1000
Od_type	Iter
Od_wait	10

fit() - parameters	
cat_features	Category variables
eval_set	X_test, Y_test

Overfitting detected after 362 iterations!

CatBoost Regressor

Model Training

<u>Step 4</u>: Predict the Y_test with the model

Outcome:

R^2	Pseudo-R^2
0.9497	0.9155

5 - Live Demo!

Wrap-up

- Same software foundation is key for successful collaborative work
- Different approaches to the same problem
- Finding the needle in the haystack -> **Learn by doing!**
- Regression was the best method for us

Happy Biking!