

# Python for Data Science

---

## kaggle Challenge - Bike count in Paris

**Adrien SENGHOR - Victor SOTO**

*“attaquants comme bappé” team*

December 20<sup>th</sup>, 2024

APM\_51460\_EP (2024-2025)



# THE GOAL AND POTENTIAL CHALLENGES AT FIRST GLANCE

## THE GOAL 🎯

- A prediction...
  - **the log hourly traffic at each bike counter in Paris**, for a given period (2020-09-01 01:00 to 2021-09-09 23:00)
- ...with accuracy
  - **measured by the root-mean-square error (RMSE)** between the predicted log-count and the evaluated target

## THE CHALLENGES at first glance

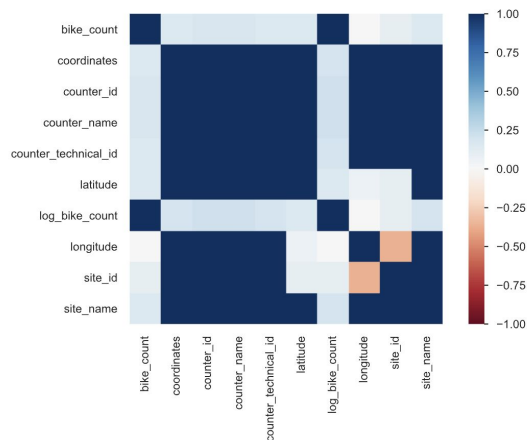
- Data exploration (missing values, etc.)
- External data integration
- Model selection and finetuning

-> EASY!!

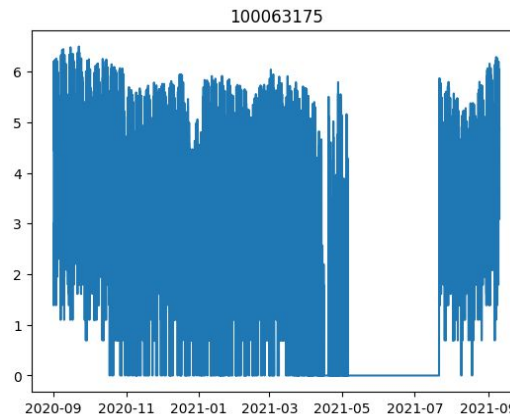


# EDA - TRAINING SET

Heatmap for X features



Log count problem with counters

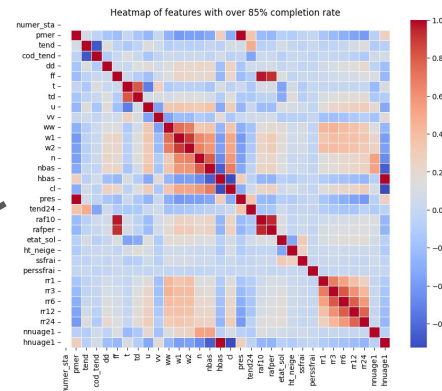
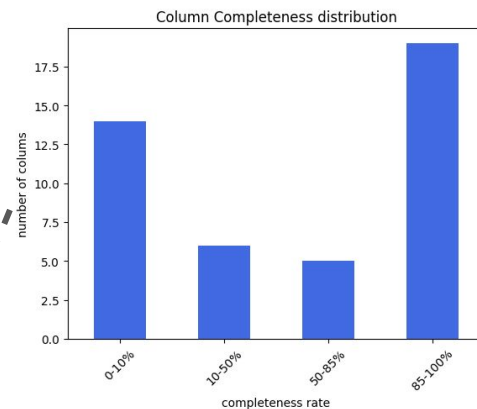


## KEY INSIGHTS

- Clean set with **redundant features** (counter location information): overfitting risk
- y data with **missing values** for identified counters, probably due to problems with the sensors on a certain period

# EDA - EXTERNAL DATA

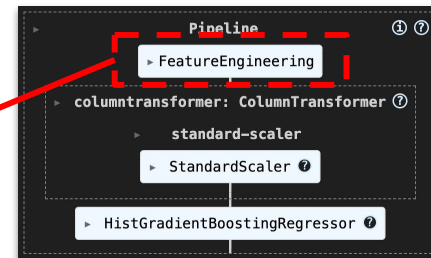
- Large dataset
  - 59 columns
  - open dataset sourced from SYNOP
- Varying data quality across columns
  - Presence of empty and incomplete columns
  - Significant number of missing values across features
- Independent features
  - Minimal correlation between features
  - Substantial amount of data requiring preprocessing



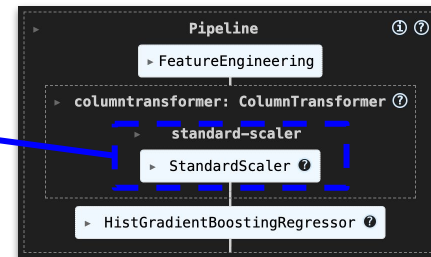
# FEATURE ENGINEERING



- Training data
  - **X["date"] column**
    - **New datetime features**
    - **Sine/cosine transformation<sup>1</sup>**
  - "Plugged-in" external data
    - Public holidays and Covid dates added
- External data -> **difficult to select the right features**
  - Attention to the chronology of the 2 datasets
  - Attention to the missing values (most frequent modality filling)



- Encoding -> **no useful categorical features spotted**
  - Only **numerical features** kept (no OneHot/OrdinalEncoder)
  - **StandardScaler<sup>2</sup>** applied



<sup>1</sup>[Three approaches to encoding time informations as features for ML models](#), Nvidia developer blog

<sup>2</sup>[StandardScaler module from scikitlearn](#) is used to normalize and scale the values from a column

# MODEL AND HYPERPARAMETERS SELECTION

- Naive approach for **feature selection**

- **Model comparison**

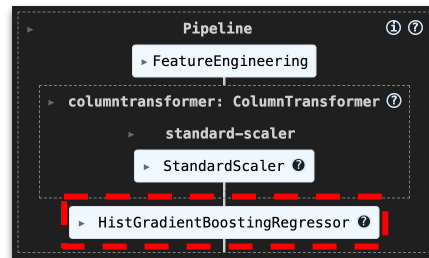
- Complete FeatureEngineering pipeline (previous slide)
- Default parameters
- **5-fold Cross-Validation**
- RMSE comparison

- **Model finetuning**

- **HistGradientBoostingRegressor**
- Hyperparameter optimization with  **OPTUNA**
- difficult to select the right ranges to explore



Model	RMSE
Ridge	0.90
RF	0.98
CatBoost	0.71
HGB	0.72



# RESULTS

- Hyperparameter finetuned
  - on learning\_rate, max\_iter, max\_depth and min\_samples\_leaf
- **BEST PREDICTION - 0.61 RMSE**  
(🏆 13th position on the Kaggle on the private Leaderboard<sup>1</sup>)
- GitHub structuration
  - FINAL\_model\_HGB\_Kaggle.py using the FINAL\_FeatureEngineering.py file



<sup>1</sup>[MSDB 2024](#), Kaggle

# Thank You!

Adrien SENGHOR - Victor SOTO





# Appendix

# OUR GITHUB

[https://github.com/victorsotofr/bike\\_counters\\_adrien\\_SENGHOR\\_vector\\_SOTO.git](https://github.com/victorsotofr/bike_counters_adrien_SENGHOR_vector_SOTO.git)

The screenshot shows the GitHub interface for the repository 'bike\_counters\_adrien\_SENGHOR\_vector\_SOTO'. At the top, there's a header with the repository name, a 'Public' badge, and buttons for 'Pin' and 'Unwatch'. Below this, a navigation bar shows 'main' as the selected branch, '4 Branches', and '0 Tags'. A search bar and 'Add file' button are also present. The main content area displays a list of files and folders. A red dashed box highlights three files: 'FINAL\_FeatureEngineering.py', 'FINAL\_model\_HGB.py', and 'FINAL\_model\_HGB\_Kaggle.py'. The file list includes folders like '.github/workflows', '\_Old', 'data', and 'external\_data', as well as files like '.DS\_Store', '.gitignore', 'LICENSE', and 'README.md'. Each entry shows a brief description and the time since the last commit.

File/Folder	Description	Last Commit
.github/workflows	simpler starter kit	2 weeks ago
_Old	Create bike_counters_starting_kit.ipynb	last week
data	Add files via upload	2 weeks ago
external_data	Instruction fichier README	2 weeks ago
.DS_Store	dernière réorganisation	last week
.gitignore	Update .gitignore	2 weeks ago
FINAL_FeatureEngineering.py	FICHIERS FINAL + réorganisation du GITHUB	last week
FINAL_model_HGB.py	ajout du fichier SUMBISSION Kaggle	last week
FINAL_model_HGB_Kaggle.py	ajout du fichier SUMBISSION Kaggle	last week
FINAL_model_HGB_test.ipynb	ajout du fichier SUMBISSION Kaggle	last week
LICENSE	Initial commit	2 weeks ago
README.md	simpler starter kit	2 weeks ago

# FEATURE ENGINEERING (1/2)

---

## DIFFICULTIES

- Select the right features
- Spot the useful correlations
- Not over-engineer our model (stack features and hope for the best incorporation within the model)

## IDEAS (1/2) 💡

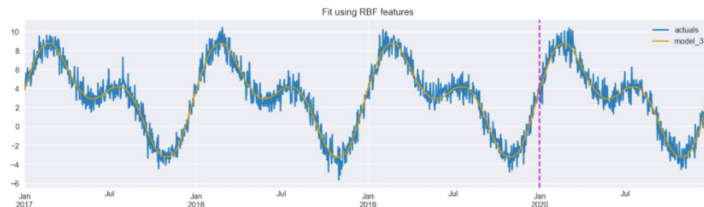
- Add more time to explore the external data
- Do **L1-regularization**, ie Lasso regression, before applying the final model
  - It enables a preselection of the most useful features

# FEATURE ENGINEERING (2/2)

## IDEAS (2/2) 💡

- Datetime feature
  - **Radial basis encoding** - not fully understood, difficult application and integration in the pipeline

```
model_3 = LinearRegression().fit(X_3.iloc[:TRAIN_END],  
                                y.iloc[:TRAIN_END])  
  
results_df["model_3"] = model_3.predict(X_3)  
results_df[["actuals", "model_3"]].plot(figsize=(16,4),  
                                           title="Fit using RBF features")  
plt.axvline(date(2020, 1, 1), c="m", linestyle="--");
```



<sup>1</sup>[Three approaches to encoding time informations as features for ML models](#), Nvidia developer blog

# MODEL SELECTION

---

## DIFFICULTIES

- Have a first idea of what could be the best model right away
- Training without spending too much time
- Finetune the final model's hyperparameters with the right ones and right ranges for them

## IDEAS

- **AutoML**<sup>1</sup>

<sup>1</sup>[AutoML](#)