# BIKE COUNT PROJECT

*Final presentation 12.12.21 – Martin Quievre – Charles Proye*

# INTRODUCTION

# SUMMARY

1. EXPLORING THE DATASET
2. ADDING NEW FEATURES
3. PREPROCESSING
4. CHOOSING THE MODEL
5. TUNING THE PARAMETERS

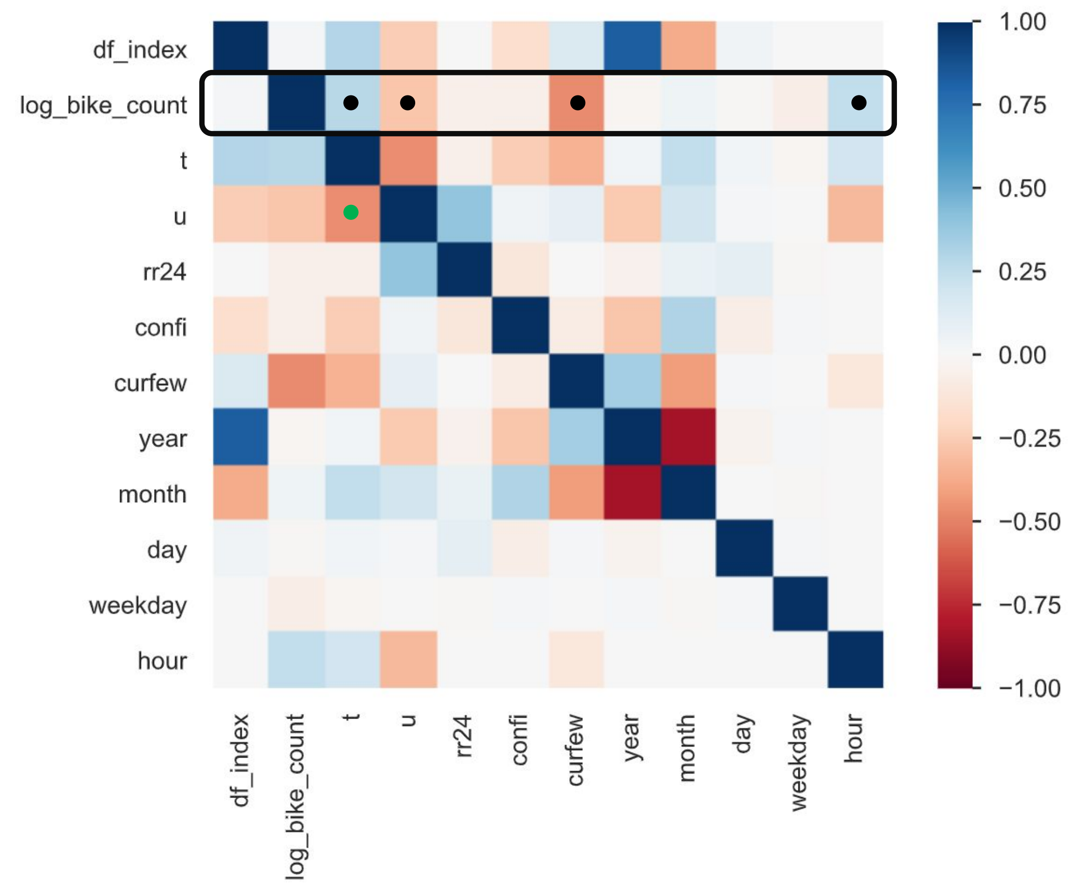
# EXPLORING THE DATASET

# SPOT THE CORRELATION

- Features significatively correlated with *log_bike_count* :
  - *curfew*
  - *hour*
  - *temperature*
  - *humidity*

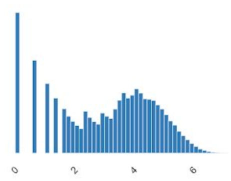
- We denote a strong correlation between :
  - *temperature / humidity*

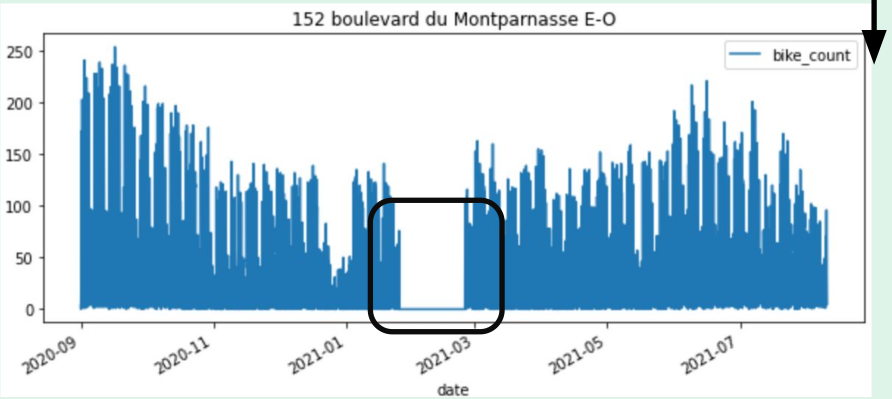
The ***counter_name*** feature is not on the heatmap but is also strongly correlated to the target value



# SPOT THE MISSING VALUES

| | | | |
|---|---|---|---|
| Distinct | 977 | Minimum | 0 |
| Distinct (%) | 0.2% | Maximum | 7.151485464 |
| Missing | 0 | Zeros | 40104 |
| Missing (%) | 0.0% | Zeros (%) | 8.8% |
| Infinite | 0 | Negative | 0 |
| Infinite (%) | 0.0% | Negative (%) | 0.0% |
| Mean | 3.048589299 | Memory size | 3.5 MiB |






**Observation** :
Large number of zeros,
might be suspicious

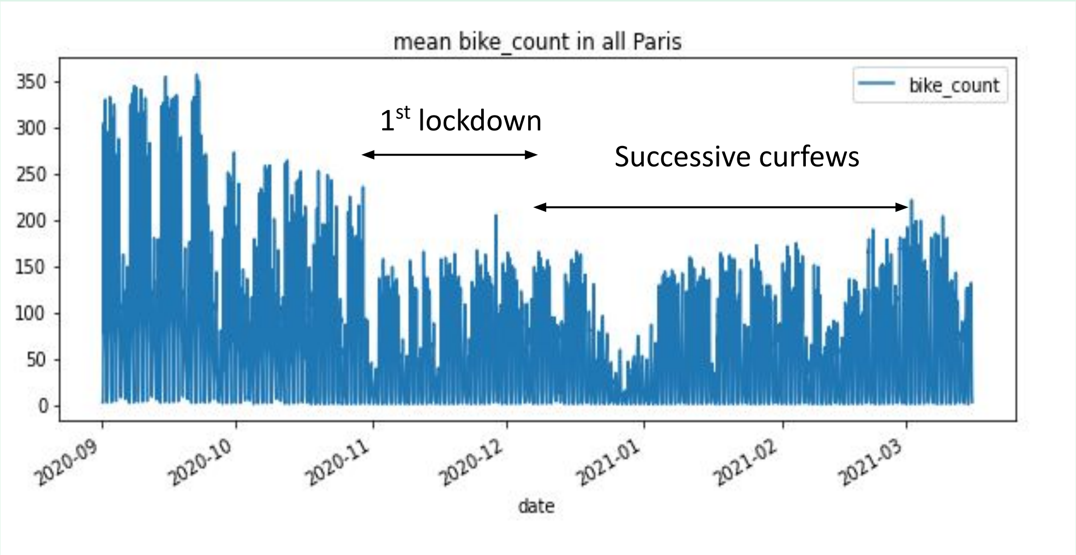
**Mask** :
To find the potential
sequences of zero

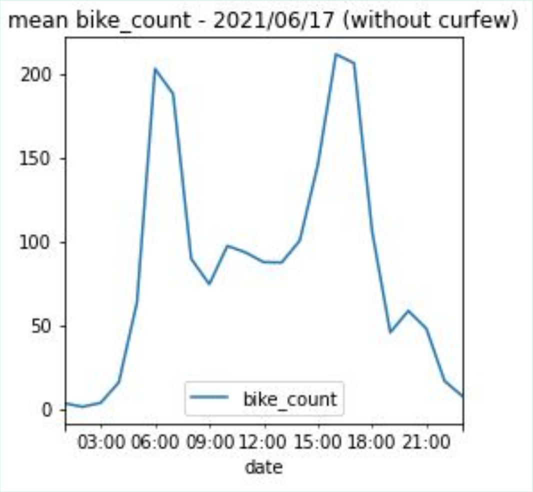
**Plot** :
To visualize those
sequences

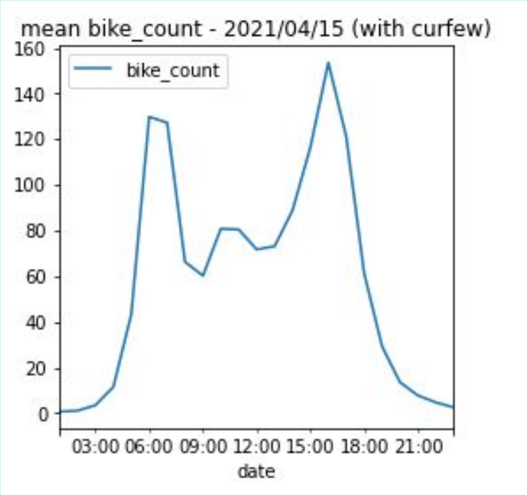
- Those unusual values are possibly due to **public work or a counter breakdown.**
- Creating a mask to get rid of those unusual values

# ADDING NEW FEATURES

# COVID IMPACT ?




**Impact of lockdown measures**

**Impact of curfew measures**

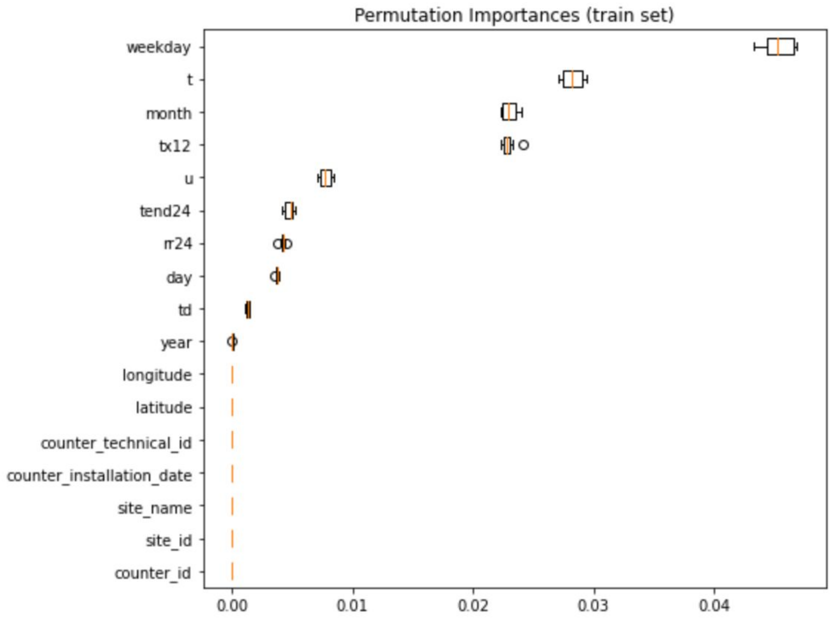
We encoded **new binary features** using a column transformer and adding it at the beginning of our pipeline to take into account these two external phenomenons.

# OTHER IDEAS

**External data file** :

> Permutation importance analysis on the variables of the external data file

> Choice to keep **u, tend24 and rr24.**




> u : humidité
> tend24 : tendance atmosphérique 24 dernières heures
> rr24 : précipitation dans les 24 dernières heures

---

**Other binary features - failed for bad results**

> Public Holidays

> Weekends

**Ideas to explore**

> [opendata.paris.fr](opendata.paris.fr)

> Other datas : construction sites ? how to merge it ? Geographically ? with coordinates?

# PREPROCESSING

# ENCODING DATES

**1st method**

**Code**

```python
def _encode_dates(X):
    X = X.copy()
    X.loc[:, 'year'] = X['date'].dt.year
    X.loc[:, 'month'] = X['date'].dt.month
    X.loc[:, 'day'] = X['date'].dt.day
    X.loc[:, 'weekday'] = X['date'].dt.weekday
    X.loc[:, 'hour'] = X['date'].dt.hour

    return X.drop(columns=["date"])
```

**Output**

| | year | month | day | weekday | hour |
|---|---|---|---|---|---|
| 48321 | 2020 | 9 | 1 | 1 | 2 |
| 48324 | 2020 | 9 | 1 | 1 | 3 |
| 48327 | 2020 | 9 | 1 | 1 | 4 |
| 48330 | 2020 | 9 | 1 | 1 | 15 |
| 48333 | 2020 | 9 | 1 | 1 | 18 |

**2nd method**

**Code**

```python
def _encode_dates_2(X):
    X = X.copy()
    X.loc[:, 'new_date'] = X['date'] - min(data['date'])
    X.loc[:, 'nb_of_seconds'] =X.loc[:,'new_date'].dt.total_seconds()

    return X.drop(columns=["date", 'new_date'])
```

**Output**

| | date | nb_of_seconds |
|---|---|---|
| 48321 | 2020-09-01 02:00:00 | 3600.0 |
| 48324 | 2020-09-01 03:00:00 | 7200.0 |
| 48327 | 2020-09-01 04:00:00 | 10800.0 |
| 48330 | 2020-09-01 15:00:00 | 50400.0 |
| 48333 | 2020-09-01 18:00:00 | 61200.0 |

# PREPROCESSING

After feature selection, merge of external and date encoding,  we have the following dataset :

Ordinal Encoder

Polynomial Features :
degree = 2

Standard Scaler

One Hot Encoder

passthrough

| year | month | day | weekday | hour | t | u | confi | curfew | counter_name |
|---|---|---|---|---|---|---|---|---|---|
| 2020 | 9 | 1 | 1 | 1 | 285.75 | 81 | 0 | 0 | 152 boulevard du Montparnasse E-O |
| 2020 | 9 | 1 | 1 | 1 | 285.75 | 81 | 0 | 0 | 152 boulevard du Montparnasse O-E |
| 2020 | 9 | 1 | 1 | 1 | 285.75 | 81 | 0 | 0 | 18 quai de l'Hôtel de Ville NO-SE |
| 2020 | 9 | 1 | 1 | 1 | 285.75 | 81 | 0 | 0 | 18 quai de l'Hôtel de Ville SE-NO |
| 2020 | 9 | 1 | 1 | 1 | 285.75 | 81 | 0 | 0 | 20 Avenue de Clichy NO-SE |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 2021 | 8 | 9 | 0 | 23 | 291.45 | 72 | 0 | 0 | Totem 85 quai d'Austerlitz SE-NO |
| 2021 | 8 | 9 | 0 | 23 | 291.45 | 72 | 0 | 0 | Totem Cours la Reine E-O |
| 2021 | 8 | 9 | 0 | 23 | 291.45 | 72 | 0 | 0 | Totem Cours la Reine O-E |

# CHOOSING THE MODEL

# TESTED MODELS

We have tested those **eight models** on the dataset :

**Model without result**

PROPHET

**Model with result**

RIDGE

HIST GRAD BOOST

RANDOM FOREST

TensorFlow Keras

**Model with result that we tuned**

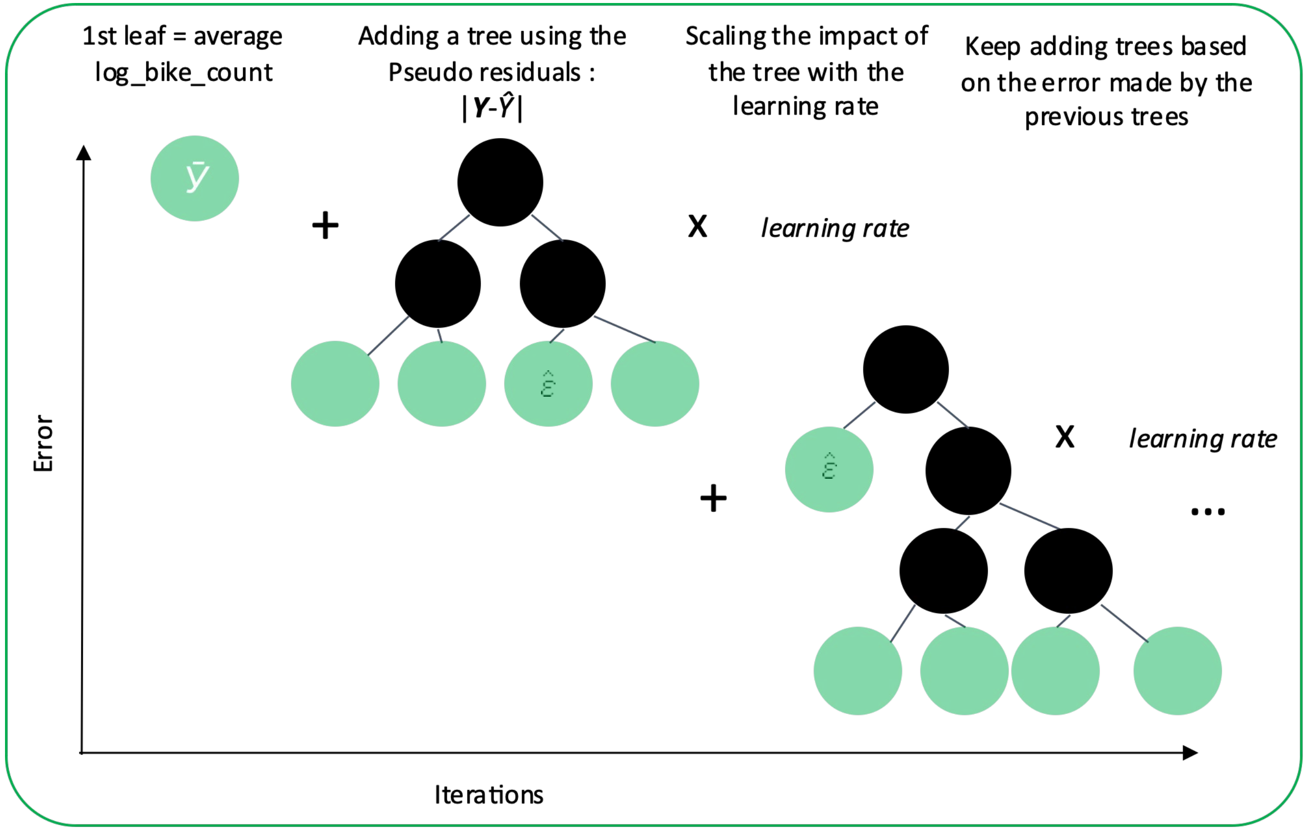
LightGBM

XGBoost

CatBoost

# THE GRADIENT BOOSTING METHOD FOR REGRESSION

The Principle :




Mathematically :

Input :

- Training set $(x_i, y_i)_{i=1}^{n}$, here $x$ is the different features and $y$ = log_bike_count
- Differentiable loss function :
  Here XGBoost built-in parameters choose squared error :
  $L(y_i, F(x)) = (y_i - F(x_i))^2$
- The number of iteration $M = n_{iter}$

Step 1 :

1. Initialize model with a constant value :
2. For $m = 1$ to $M$ :
   - Compute pseudo residuals :
     $r_{im} = -[\frac{\partial L(y_i, F_{m-1}(x_i))}{\partial F_{m-1}(x_i)}] = 2(y_i - F_{m-1}(x_i)) \quad i = 1, .., n$
   - Fit a tree $h_m(x)$ to pseudo-residuals with the training set : $(x_i, r_{im})_{i=1}^{n}$
   - Solve the following optimization problem :
     $\gamma_m = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i))$
     This is often solved using SGD (XGBoost)
   - Update the model : $F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$
3. Output $F_M(x)$

# NEURAL NETWORK

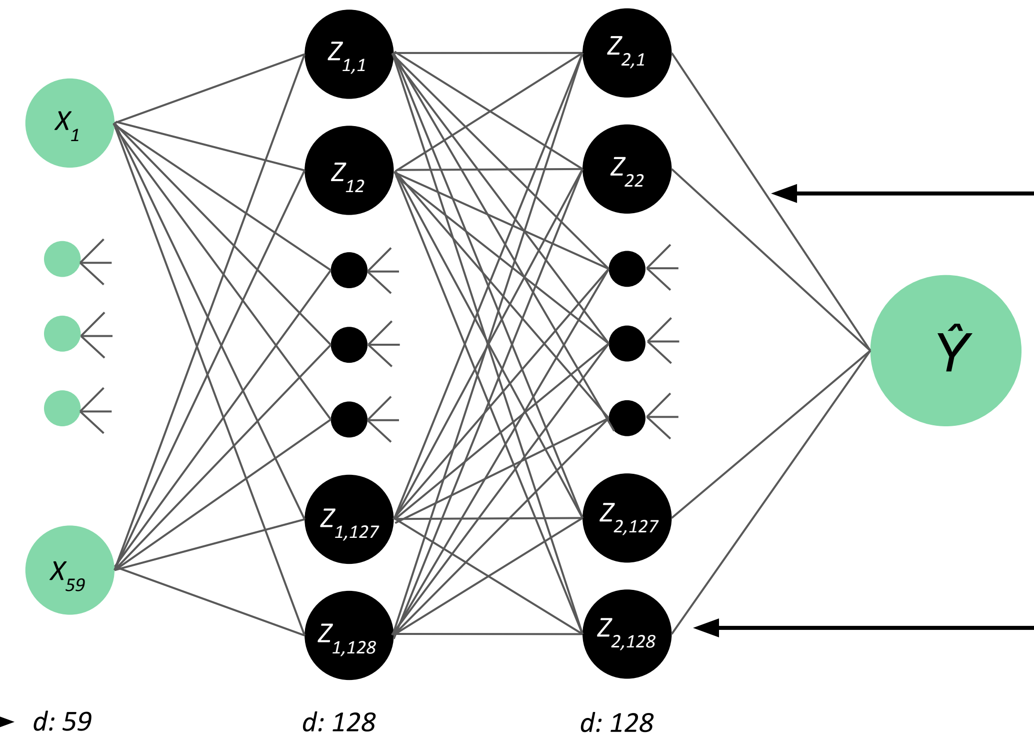
Using **TensorFlow** and **Keras** we tried to implement a Neural Network :

**Structure**

- One input layer – 64 inputs
- Two hidden layers – 128 neurons each
- One dimension output layer
- Batch size : 50
- Epochs : 15

**59 ?**

- *counter_name :* 56 features as one hot encoded
- *datetime* : 2 features
- *meteo* : 1 features




**Parameters Θ**

- 59 * 128 + 128 = 7 680
- 128 * 128 + 128 = 16 512
- 128 + 1 = 129

***24 321 parameters***

**Activation function ?**

After a few tests, it appears that Exponential linear unit (ELU) activation function was better than Sigmoid, RELU and SELU.

$$\mathrm{ELU}(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(e^x - 1) & \text{if } x < 0 \end{cases}$$

# TUNING THE PARAMETERS

# PARAMETERS TUNING

To tune our parameters while dealing with the trade-off between time and precision of parameter tuning,  we tried the two following methods :

**# 1 By trial and error and by 'parameter category'**

Extract from Notebook – Method 1

```
# 1ST PART CONTROL OVER TREE STRUCTURE

params = {
'lgbmregressor__max_depth': [6,24],
'lgbmregressor__num_leaves': [2**6, 2**10],
'lgbmregressor__min_data_in_leaf':[40, 200],
'lgbmregressor__n_estimators': [130, 200],
}
grid_search = GridSearchCV(pipe, param_grid=params,
n_jobs=-1, cv=5)
result_gs = grid_search.fit(X_train, y_train)

print(f"Best params : {result_gs.best_params_}")
                    ...
```

**# 2 Randomized Grid Search**

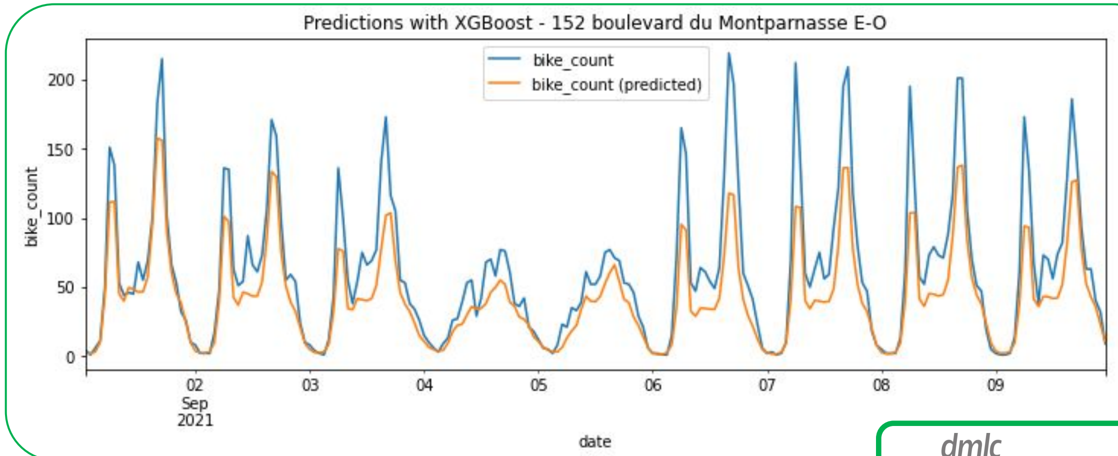
Extract from Notebook – Method 1

```
# RANDOMIZE GRID SEARCH

params = {
'lgbmregressor__learning_rate' : [0.01, 0.02, 0.03, 0.04, 0.05,
0.08, 0.1, 0.2, 0.3, 0.4],
'lgbmregressor__n_estimators' : [100, 200, 300, 400, 500, 600,
800, 1000, 1500, 2000],
                        ...
'lgbmregressor__reg_alpha': [0, 1e-1, 1, 2, 5, 7, 10, 50, 100],
'lgbmregressor__reg_lambda': [0, 1e-1, 1, 5, 10, 20, 50, 100]
}

grid_search = RandomizedSearchCV(pipe,
param_distributions=params, n_jobs=4, cv=5, n_iter=200)
result_gs = grid_search.fit(X_train, y_train)
print(f"Best params : {result_gs.best_params_}")
```

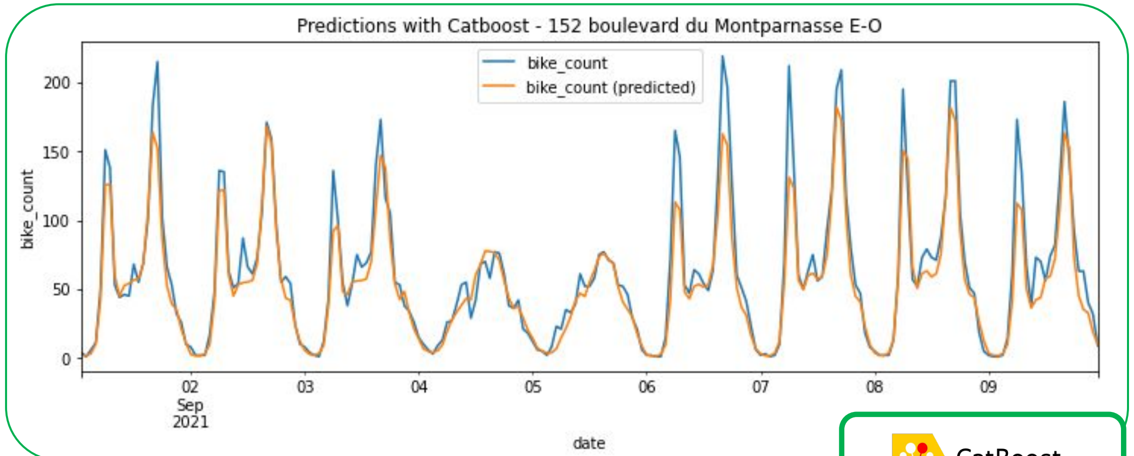
# RESULTS

| MODELE | RMSE - RAMP | RMSE - LOCAL | TRAIN TIME | VALIDATION TIME |
|---|---|---|---|---|
| dmlc XGBoost | 0.743 | 0.703 | 367 | 92 |
| LightGBM | 0.750 | 0.738 | 146 | 295 |
| CatBoost | 0.741 | 0.741 | 257 | 47 |

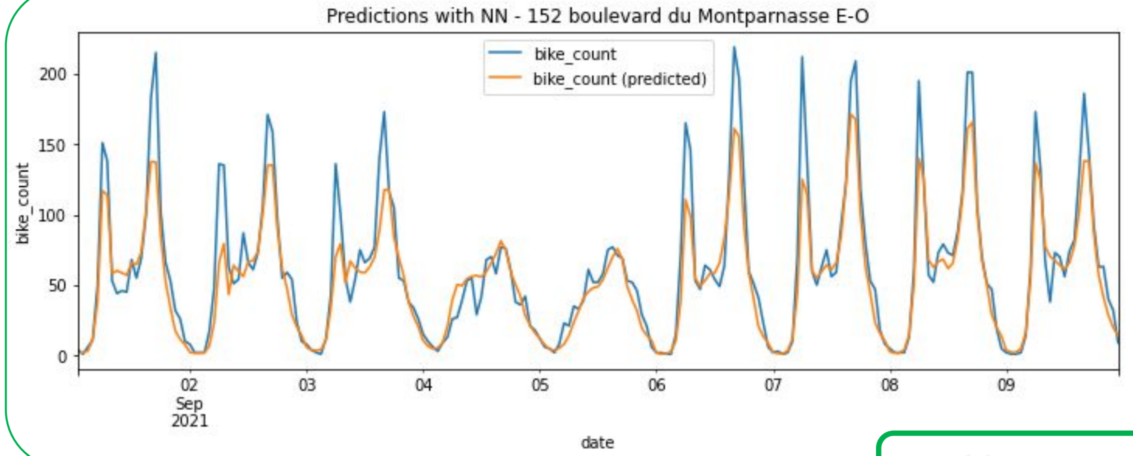
# VISUALIZING THE RESULTS



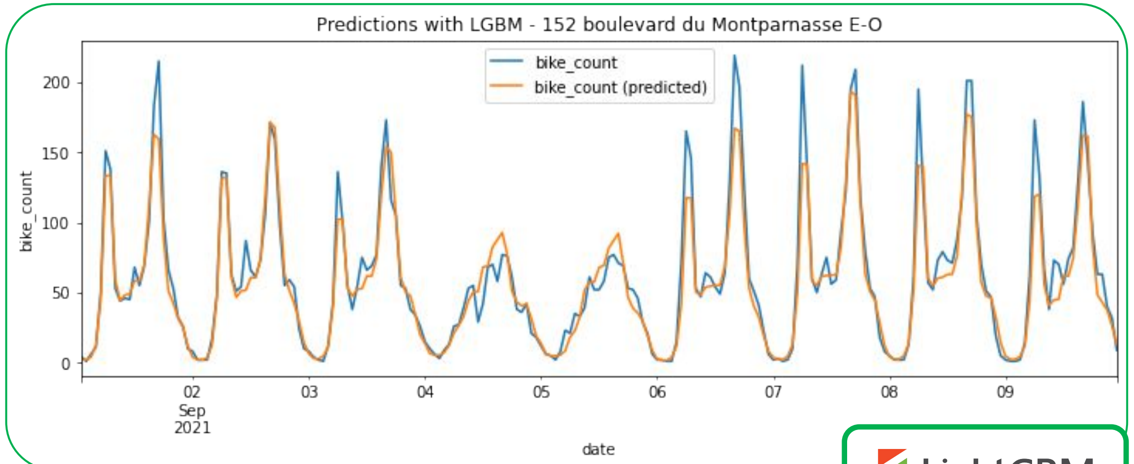

dmlc XGBoost




CatBoost




TensorFlow Keras




LightGBM