/01

# Project on Facebook Comment Volume Dataset

Python for data analysis

Martin Liger, Jean Baptiste Martin and Lamiae Maliki

# Summary

## Data overview

## Data cleaning

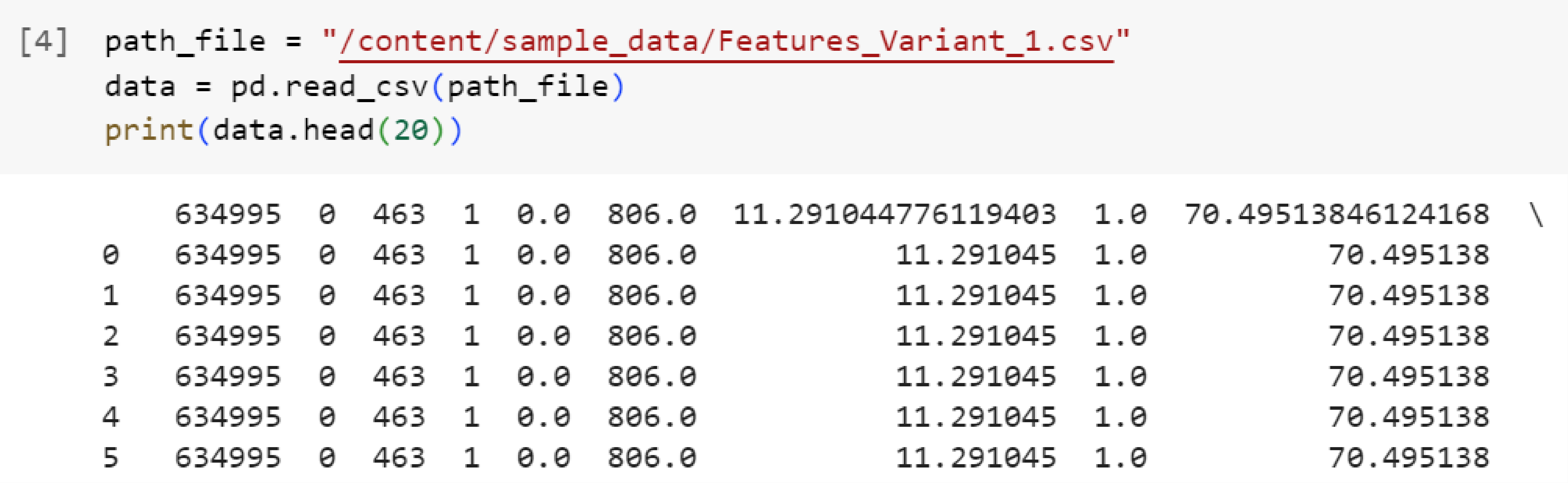
## Creation of the model

## Bonus work and Conclusion

# /03

**Goal** :  predict how many comments a post will receive

overview of the dirty dataset:



```
[4]  path_file = "/content/sample_data/Features_Variant_1.csv"
     data = pd.read_csv(path_file)
     print(data.head(20))
```

```
       634995  0  463  1  0.0  806.0  11.291044776119403  1.0  70.49513846124168  \
   0   634995  0  463  1  0.0  806.0           11.291045  1.0          70.495138
   1   634995  0  463  1  0.0  806.0           11.291045  1.0          70.495138
   2   634995  0  463  1  0.0  806.0           11.291045  1.0          70.495138
   3   634995  0  463  1  0.0  806.0           11.291045  1.0          70.495138
   4   634995  0  463  1  0.0  806.0           11.291045  1.0          70.495138
   5   634995  0  463  1  0.0  806.0           11.291045  1.0          70.495138
```

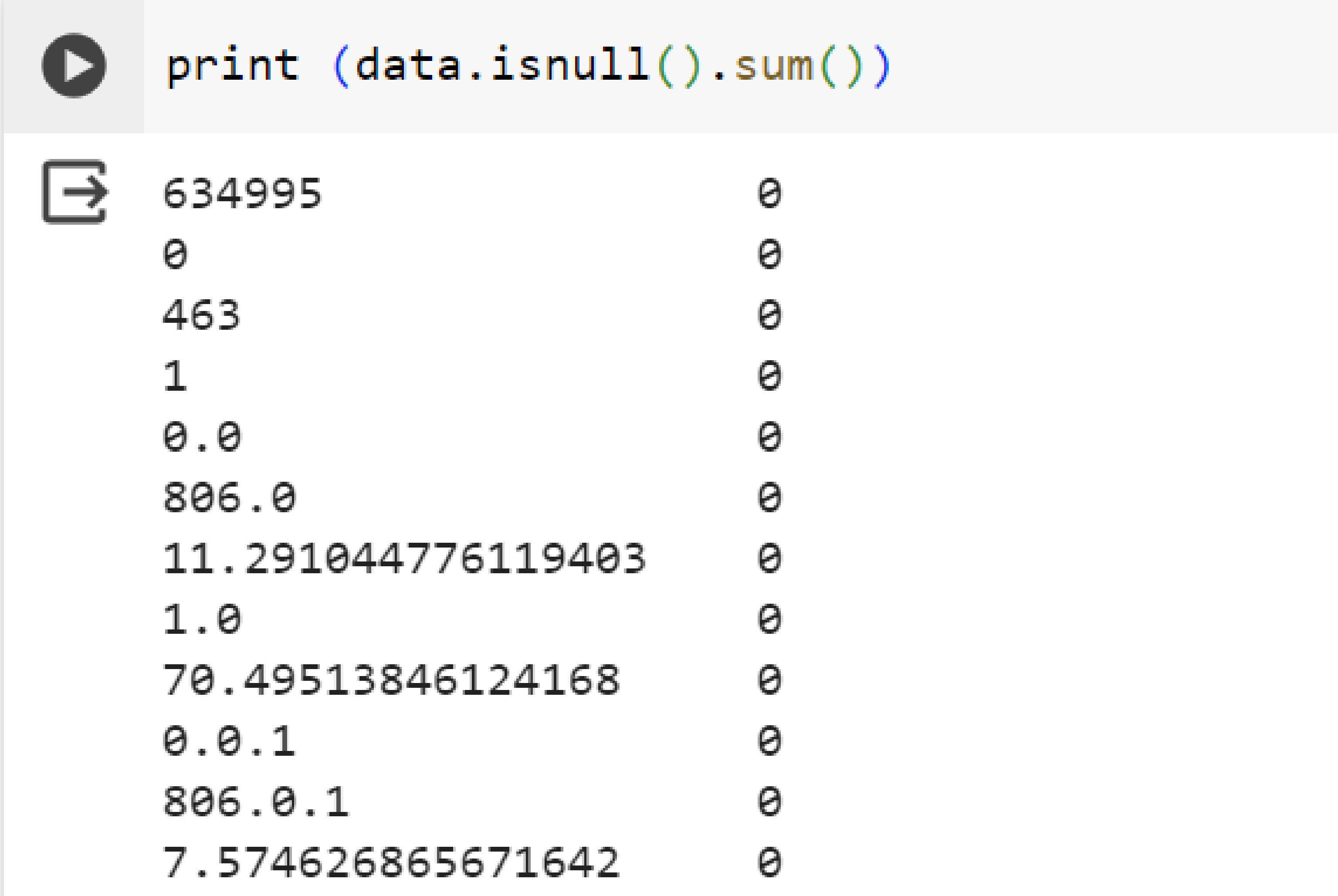
First problem :
columns are unnamed

Rapport d'Activité | 2022

**Data overview**

# /04

Second major observation :  there is no NaN or missing values :



```
print (data.isnull().sum())
```

```
634995                  0
0                       0
463                     0
1                       0
0.0                     0
806.0                   0
11.291044776119403      0
1.0                     0
70.49513846124168       0
0.0.1                   0
806.0.1                 0
7.574626865671642       0
```

but are all the 54 columns interesting?

**Data overview**

# /05

To solution this issue we did :

- spot the columns with irrelevant values with a dedicated algorithm
- plot the correlation matrix to find and drop unnecessary columns

-> 18 columns removed

issues encountered :

- the code used to drop every correlated column did not take account of the negative correlations
- one empty column was not detected by our algorithm
- some columns are correlated but still interesting (CC1 and CC4)

**Data cleaning**

Selection of the output variable :

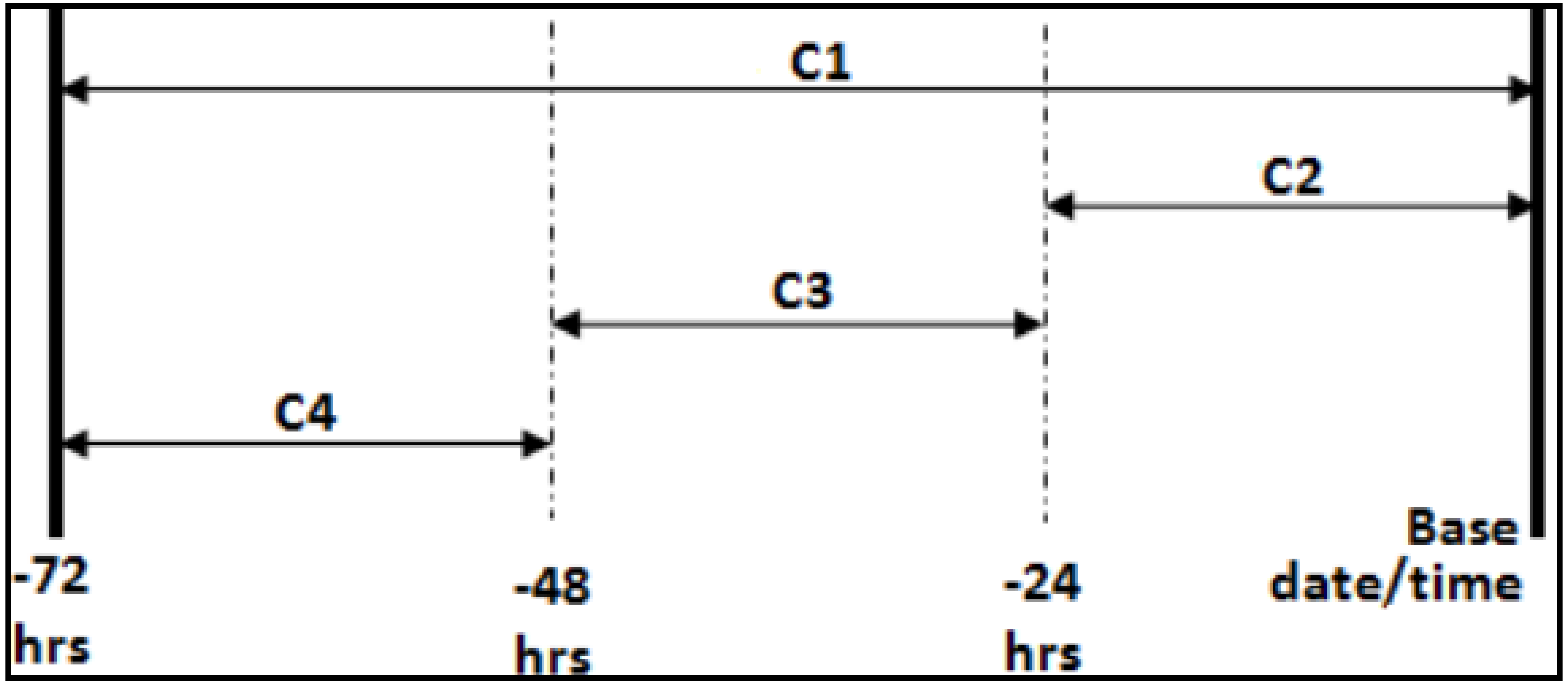
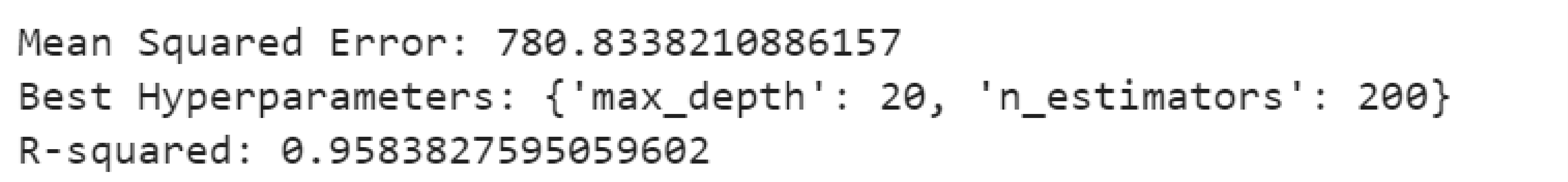


Figure 2. Demonstrating the essential feature details.

C1 was kept as the output feature

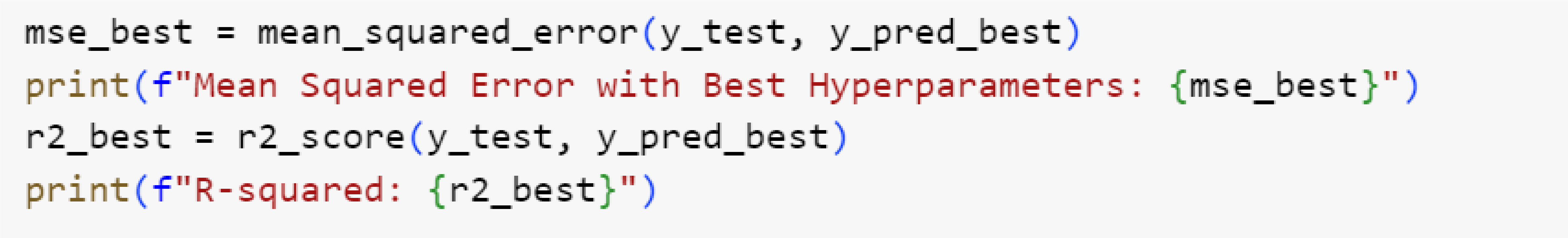
First observation : the dataset is divided in a test and a train part -> no need to make a pipeline

Results of the RandomForestRegressor without gridsearch :

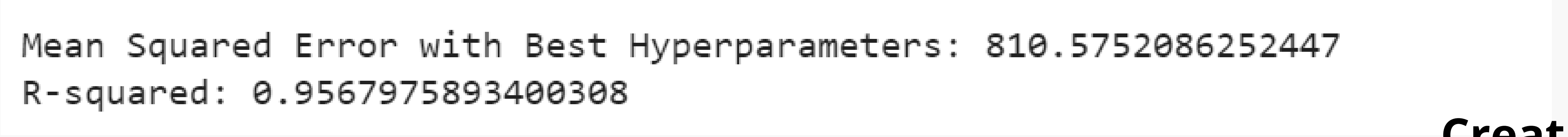


```
Mean Squared Error: 780.8338210886157
Best Hyperparameters: {'max_depth': 20, 'n_estimators': 200}
R-squared: 0.9583827595059602
```

Now with gridsearch :



```
mse_best = mean_squared_error(y_test, y_pred_best)
print(f"Mean Squared Error with Best Hyperparameters: {mse_best}")
r2_best = r2_score(y_test, y_pred_best)
print(f"R-squared: {r2_best}")
```

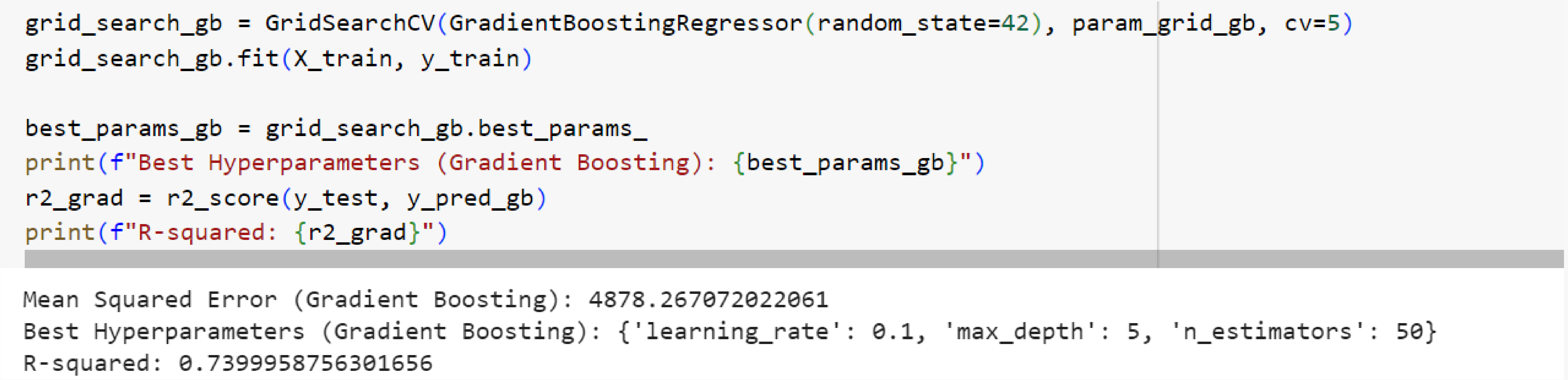


```
Mean Squared Error with Best Hyperparameters: 810.5752086252447
R-squared: 0.9567975893400308
```

**Creation of a model**

## Issue : the results obtaines without gridsearch are better

### Lets try Gradient Boosting Regressor



```
grid_search_gb = GridSearchCV(GradientBoostingRegressor(random_state=42), param_grid_gb, cv=5)
grid_search_gb.fit(X_train, y_train)

best_params_gb = grid_search_gb.best_params_
print(f"Best Hyperparameters (Gradient Boosting): {best_params_gb}")
r2_grad = r2_score(y_test, y_pred_gb)
print(f"R-squared: {r2_grad}")
```

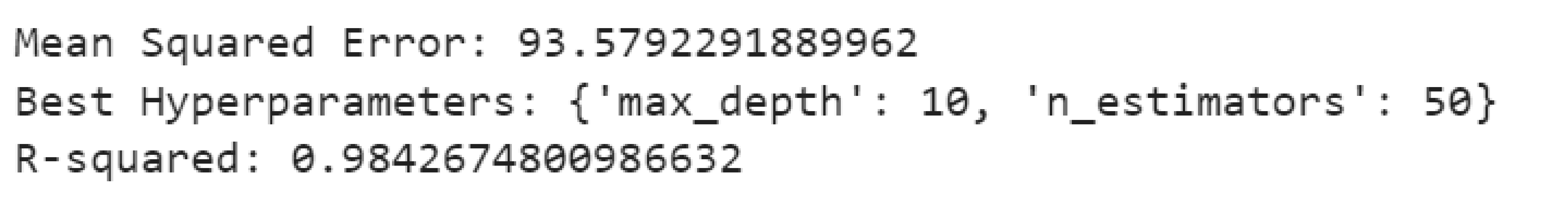
```
Mean Squared Error (Gradient Boosting): 4878.267072022061
Best Hyperparameters (Gradient Boosting): {'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 50}
R-squared: 0.7399958756301656
```

We notice that we have pretty high MSE and a r2 closed to one. We may have only a few predictions failed, but that they are failed by a lot.

**Creation of a model**

## /09

**Bonus work** predict CC2 in function of CC3 and CC4, let’s try with random forest :



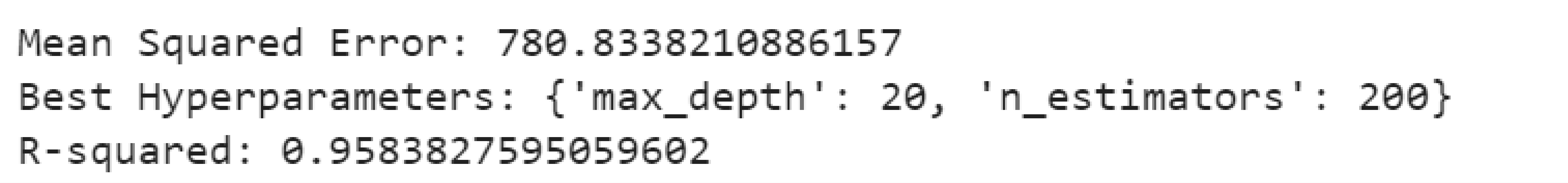
```
Mean Squared Error: 93.5792291889962
Best Hyperparameters: {'max_depth': 10, 'n_estimators': 50}
R-squared: 0.9842674800986632
```

Using gridsearch the MSE and R2 obtained are more than acceptable

Issue encountered : we tried to use a RandomForestClassifier to improve our model but the dataset is too big to be trained with this method

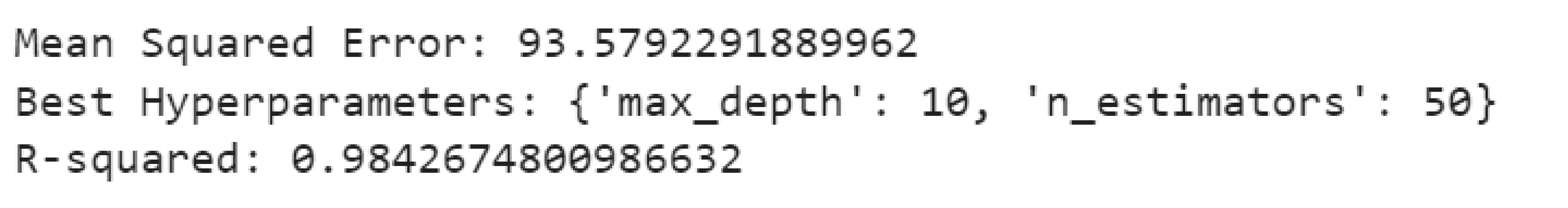
# Conclusion

For our first model predicting cc1 in function of all the data available, we selected the  **initial RandomForestRegressor**, giving us the following accuracy :



```
Mean Squared Error: 780.8338210886157
Best Hyperparameters: {'max_depth': 20, 'n_estimators': 200}
R-squared: 0.9583827595059602
```

For the bonus work predicting the number of comment obtained a day, just by looking at the number of comments the 2 days before we selected a **gridsearch optimized RandomForestRegressor** with the following accuracy :



```
Mean Squared Error: 93.5792291889962
Best Hyperparameters: {'max_depth': 10, 'n_estimators': 50}
R-squared: 0.9842674800986632
```

For both programs the results are very satisfying