Facebook Comment : Predictions on new posts

Presented by
LEMPEREUR Julien & LIU Johan

Table of Contents

01. Introduction

02. Data Preprocessing

03. Data Visualization

04. Modeling

05. Results

Introduction

- 32 million active users and the average French user spends 13 hours on Facebook
- The average Facebook Page posts 1.68 times per day
- Our goal : Predict how much commentary this post will have
- Link to our dataset : https://archive.ics.uci.edu/dataset/363/facebook+comment+volume+dataset


Data
Preprocessing

Features

```
page_features = ['likes','visitors','daily_interest','category']
derived = ['derived_{}'.format(i) for i in range(1,26)]
essential_features = ['C_{}'.format(i) for i in range(1,6)]
base_features = ['base_time','post_length','share_count','promotion_status','H_hrs']
weekday = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday']
base_DT = ['BDT_Sun', 'BDT_Mon', 'BDT_Tue', 'BDT_Wed', 'BDT_Thu', 'BDT_Fri', 'BDT_Sat']
target = ['No_of_Comments_in_H_hours']
cols = page_features+derived+essential_features+base_features+weekday+base_DT+target
```

- Page_features ==> Basic data on posts
- Derived ==> Statistical Data on different features (we do not know precisely what)
- Essential_features ==> Statistics on comments in relation to time
- Weekday ==> Date of post
- Base_DT ==> Date of sampling
- Target ==> N° of Comments

Data

	likes	visitors	daily_interest	category	derived_1	derived_2	derived_3	derived_4	derived_5	derived_6	...	Friday	Sat
0	634995	0	463	1	0.0	806.0	11.291045	1.0	70.495138	0.0	...	0	
1	634995	0	463	1	0.0	806.0	11.291045	1.0	70.495138	0.0	...	0	
2	634995	0	463	1	0.0	806.0	11.291045	1.0	70.495138	0.0	...	1	
3	634995	0	463	1	0.0	806.0	11.291045	1.0	70.495138	0.0	...	1	
4	634995	0	463	1	0.0	806.0	11.291045	1.0	70.495138	0.0	...	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
40944	7170111	70	497000	9	0.0	1881.0	497.200000	269.0	502.318385	0.0	...	0	
40945	7170111	70	497000	9	0.0	1881.0	497.200000	269.0	502.318385	0.0	...	0	
40946	7170111	70	497000	9	0.0	1881.0	497.200000	269.0	502.318385	0.0	...	0	
40947	7170111	70	497000	9	0.0	1881.0	497.200000	269.0	502.318385	0.0	...	0	
40948	7170111	70	497000	9	0.0	1881.0	497.200000	269.0	502.318385	0.0	...	0	

Data Cleaning

We had a relatively clean data with no missing values, we just had to organize it and remove the irrelevant features

C_total	16.351830
C_last_24h	36.666175
C_last_48h-24h	51.040437
C_release	16.890496
C2-C3	23.133117
base_time	1.389709
post_length	11.202529
share_count	0.000000
promotion_status	100.000000
H_hrs	0.000000
Sunday	87.764414
Monday	85.673701
Tuesday	85.020661
Wednesday	84.268005
Thursday	85.583924
Friday	85.384691
Saturday	86.304604
BDT_Sun	85.895071
BDT_Mon	86.660026
BDT_Tue	86.158255
BDT_Wed	85.452332
BDT_Thu	84.481995
BDT_Fri	85.500295
BDT_Sat	85.852027
No_of_Comments_in_H_hours	55.386659

```
# Promotion Status contains a single value and hence is
train1.drop('promotion_status', axis=1, inplace=True)
train2.drop('promotion_status', axis=1, inplace=True)
train3.drop('promotion_status', axis=1, inplace=True)
train4.drop('promotion_status', axis=1, inplace=True)
train5.drop('promotion_status', axis=1, inplace=True)
```

```
X1 = train1.iloc[:,:-1]
y1 = train1.iloc[:,-1]
X2 = train2.iloc[:,:-1]
y2 = train2.iloc[:,-1]
X3 = train3.iloc[:,:-1]
y3 = train3.iloc[:,-1]
X4 = train4.iloc[:,:-1]
y4 = train4.iloc[:,-1]
X5 = train5.iloc[:,:-1]
y5 = train5.iloc[:,-1]
```

```
y = pd.concat([y1,y2,y3,y4,y5], axis=0)
y
```

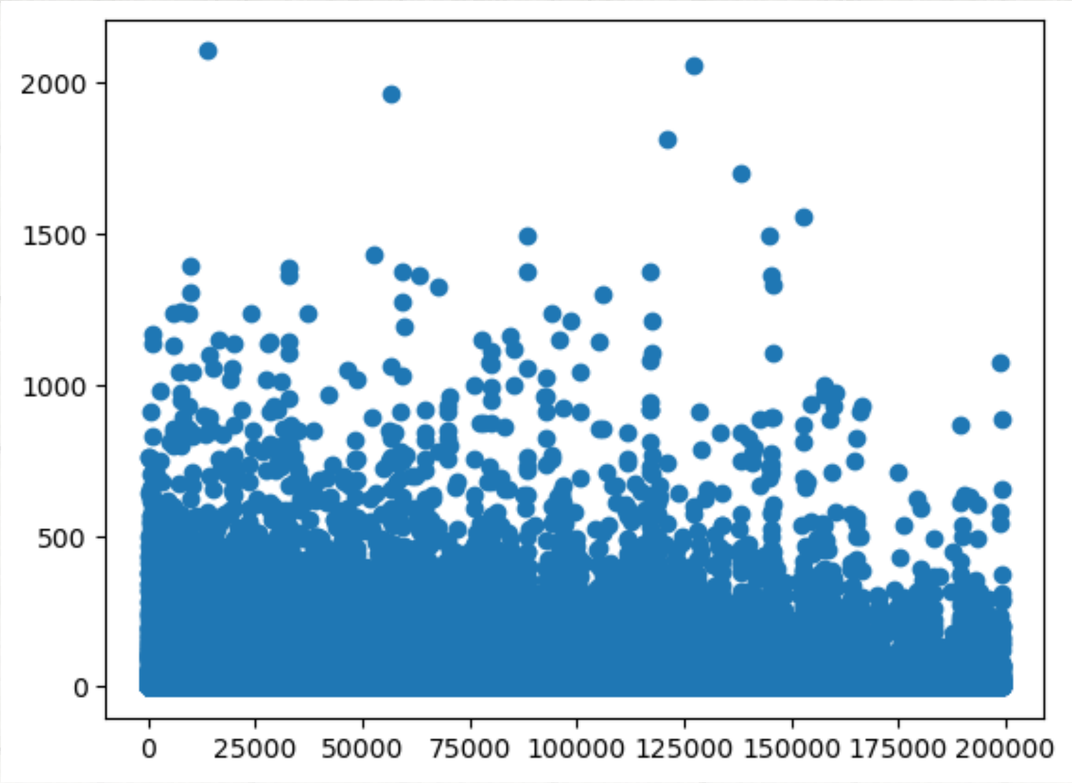
```
X = pd.concat(
    [X1,X2,X3,X4,X5],
    axis=0,
    join="outer",
    ignore_index=False,
    keys=None,
    levels=None,
    names=None,
    verify_integrity=False,
    copy=True,
)
X.shape
```

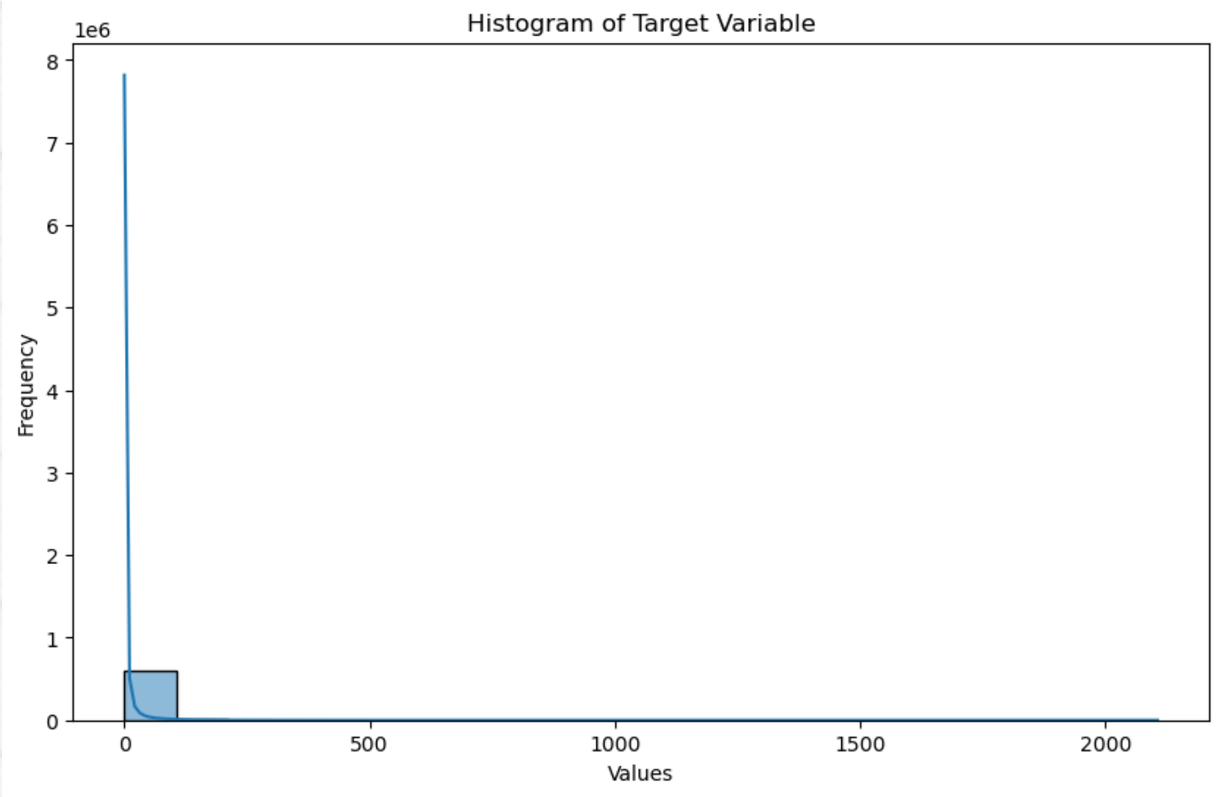
```
(602813, 52)
```


Data
Visualization

Target Variable

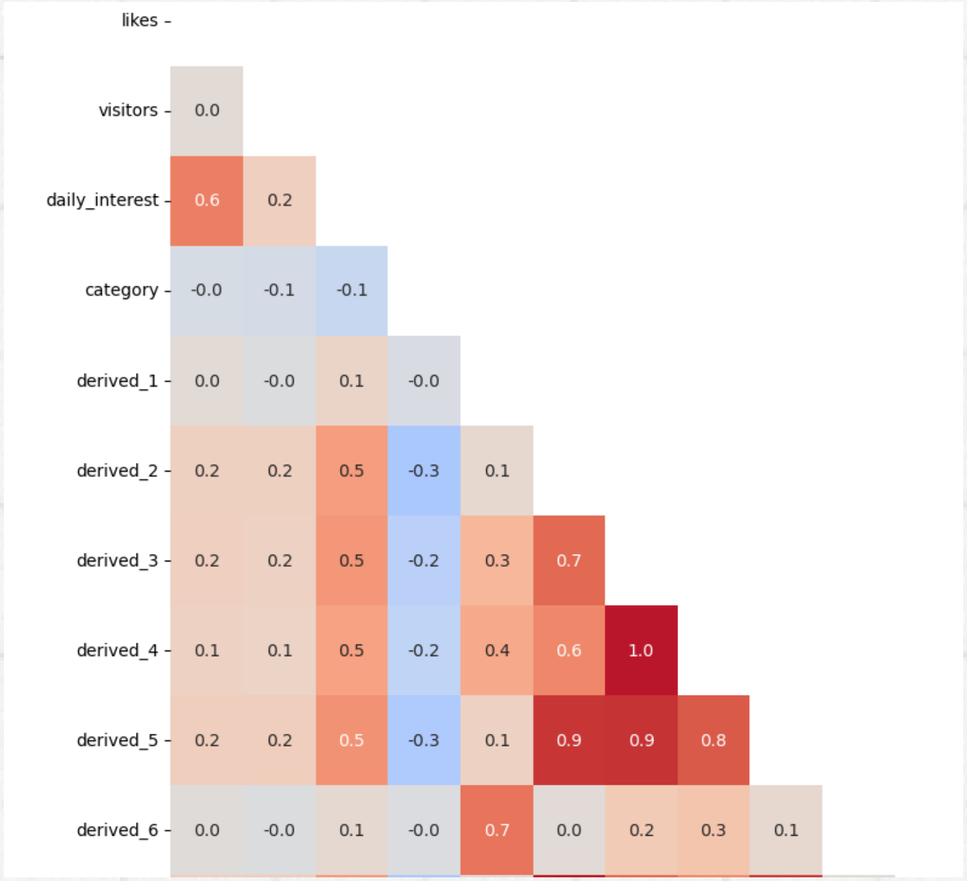
- Represent the number of commentary for each post in our training dataset
- Small amount of posts reach more than 1000 commentaries over 600 000 posts





- The dataset is realistic as it depends on the number of followers but also of the category.

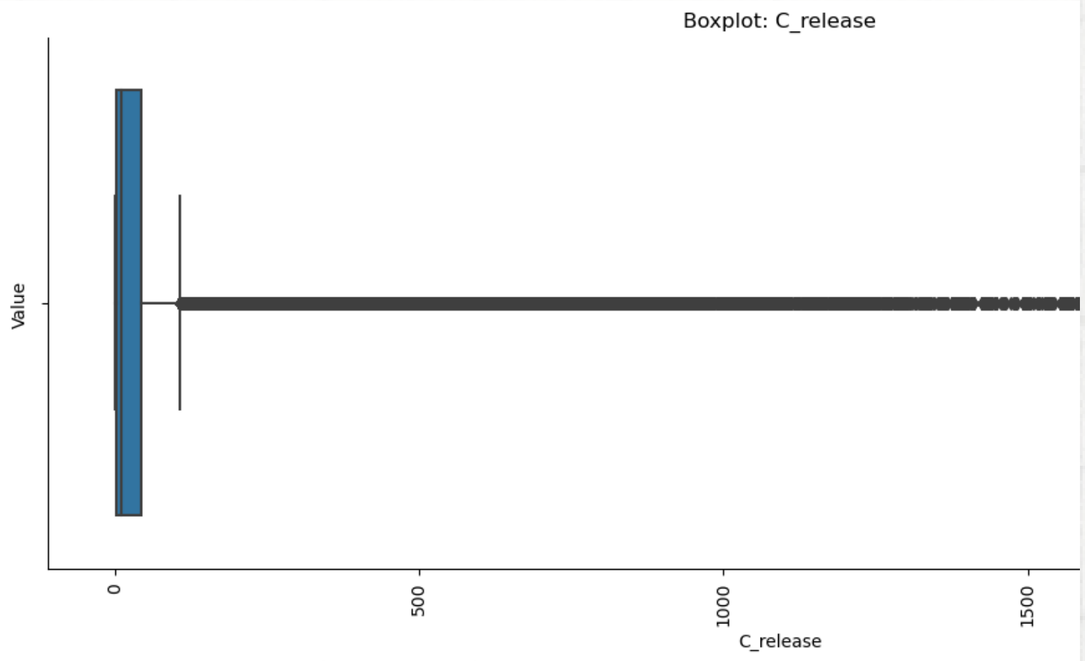
Features Columns



Correlation Matrix :

- Correlation between likes & daily_interest
- Huge correlation between our statistical features
- No real correlation between features

Features Columns

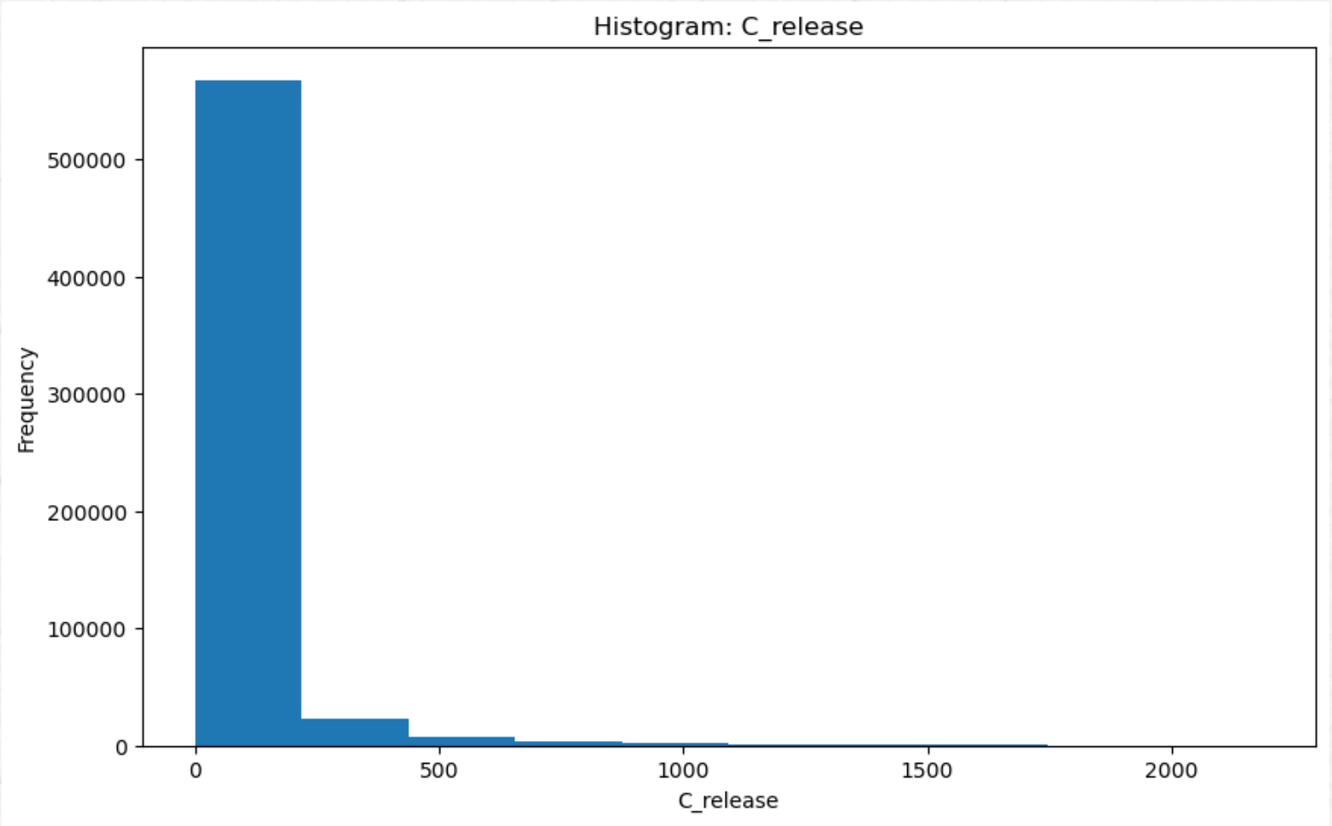


Boxplot of number of commentary 24h after release :

- Interquartile : 0 < x < 50
- Lot of outliers
- Bad distributions of the values

Histogram of number of commentary 24h after release :

- High frequency between 0 & 200ish over 600 000 posts




290
271
270
256
250
250
240
240
290
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Nov
Dec

Modeling

Modeling

Issues with GridSearch and Cross Validation :

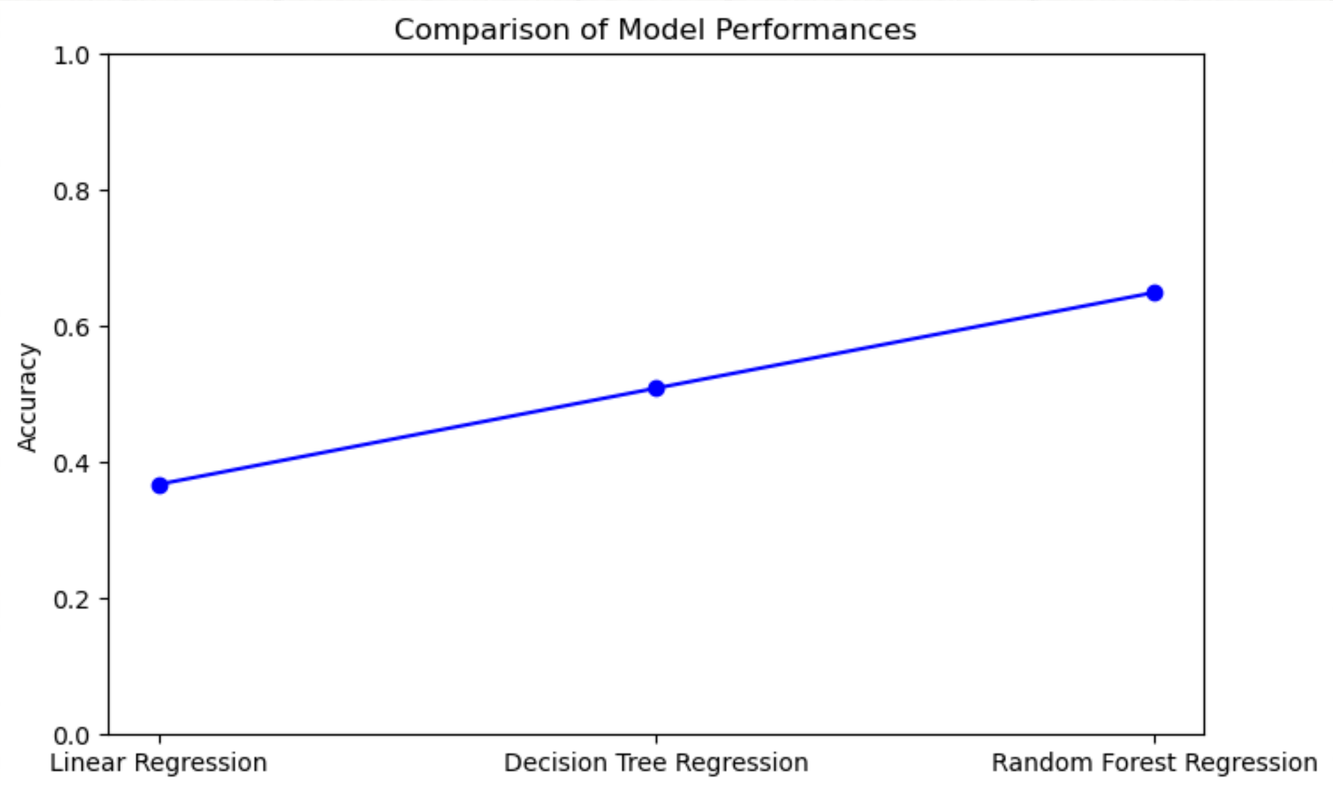
Limited to using smaller hyperparameters and unability to use CV

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train,y_train)
lr_pred = lr.predict(X_test)
```

```
from sklearn.tree import DecisionTreeRegressor
dr = DecisionTreeRegressor()
dr.fit(X_train,y_train)
dr_pred = dr.predict(X_test)
```

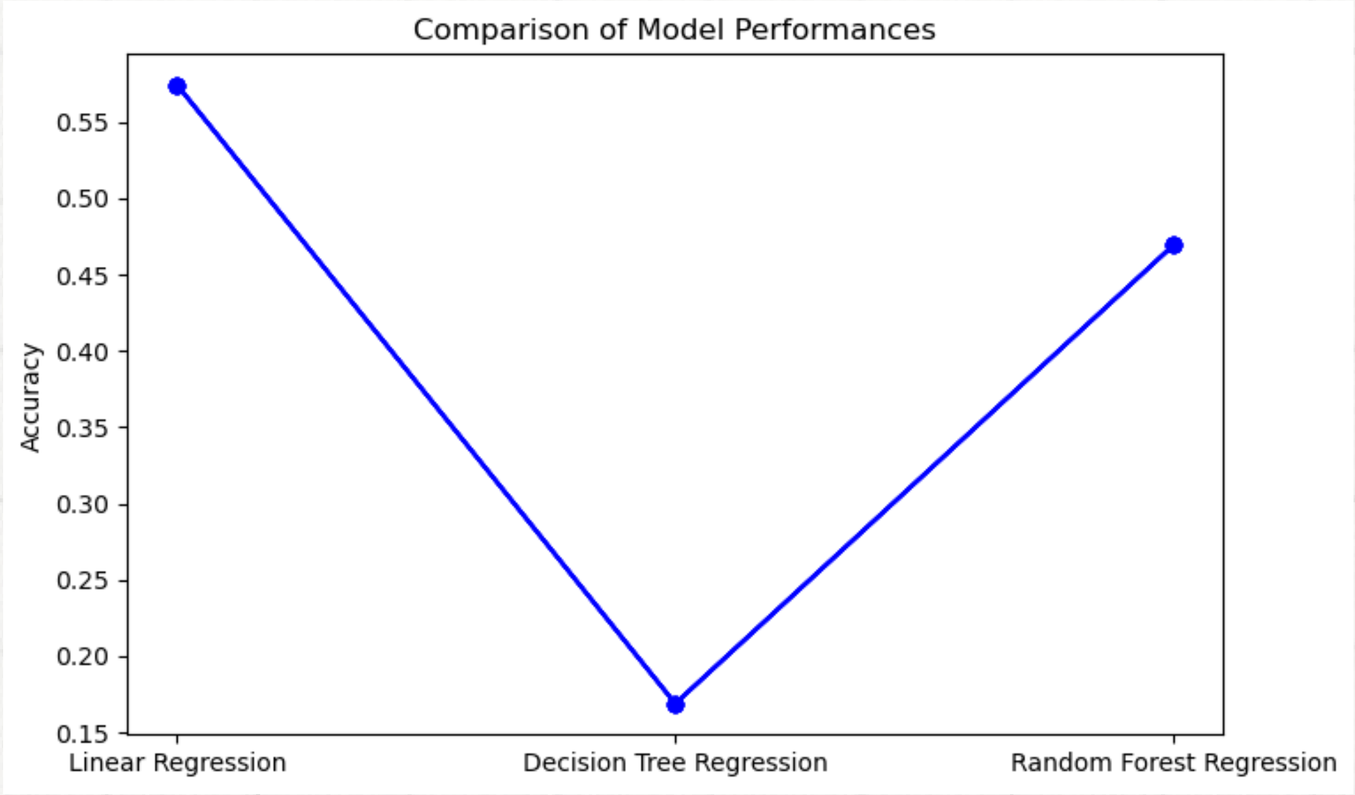
Modeling

Analyzing the accuracy of each model using score ：



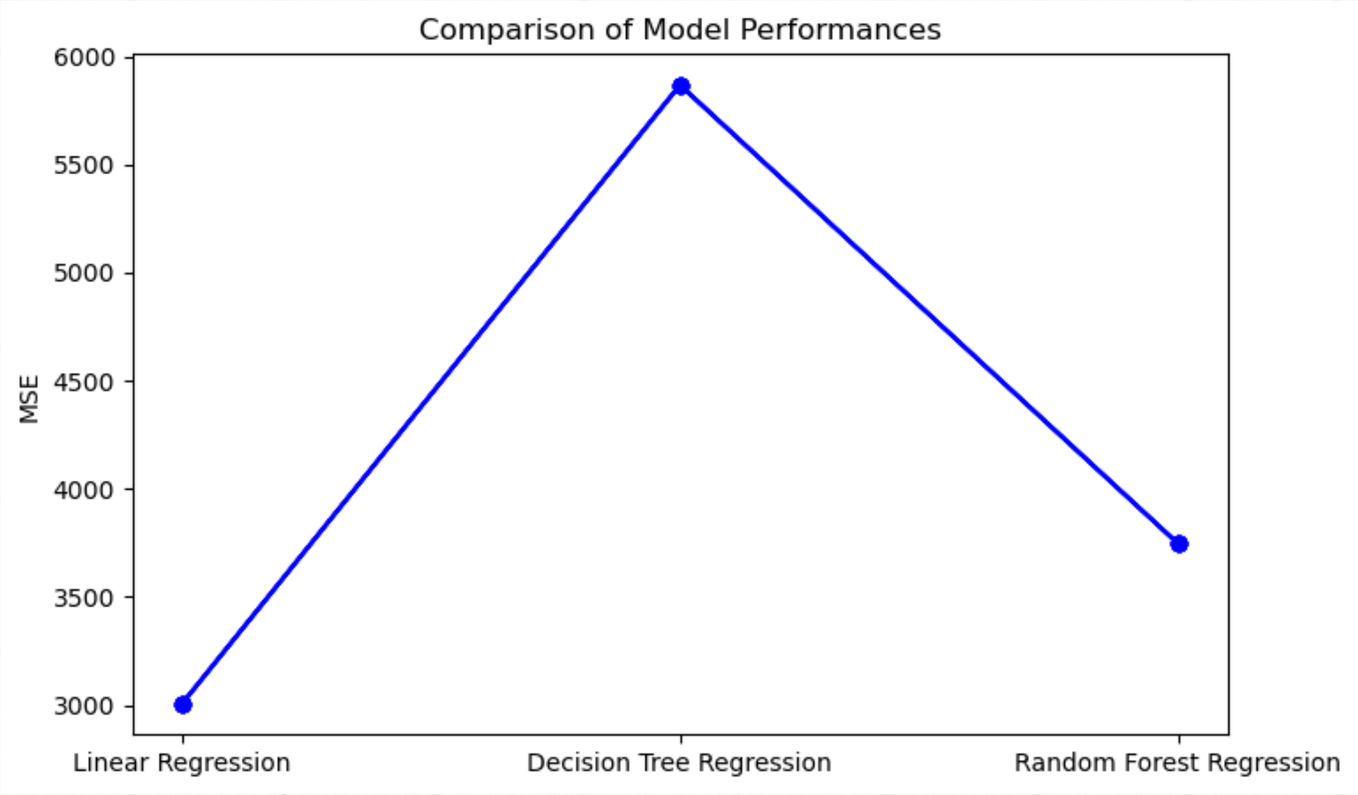
Modeling

Applying our models to our testing Data : Comparing Accuracy



Modeling

Applying our models to our testing Data : Comparing MSE(mean squared error)



Conclusion

- Dataset with bad distributions in the values (sample techniques used ?)
- From the TestSet, Linear Regression has the best performance in the prediction
- Better results with Cross-Validation for Random Forest & Decision Tree
- GitHub Link : https://github.com/Fritober/Python-for-DA

Thank tou for listening

Presented by
LEMPEREUR Julien & LIU Johan