ENSIIE

Report

New York : Prediction of taxi trip duration

Imane Alla

Ghada Ben Said

Nisrine Moummou

Mame Diarra Toure

Enseignants :

M. Hanczar

M. Janodet

14/03/2020

23 mars 2020

Table des matières

1 Introduction

We were challenged to build a model that predicts the total ride duration of taxi trips in New York City. Our primary dataset is the one released by the NYC Taxi and Limousine Commission, which includes pickup time, geo-coordinates, number of passengers, and several other variables. The data set is divided in two files : a training set composed of 1458644 observations of 11 initial variables with the target variable being **the trip duration** and a test data set composed of 625134 observations of 10 initial variables (the target variable was removed since it's the one we want to predict. We are going to proceed as follows : First we will start by cleaning up our data set. That's mean getting ride of any observation that seem like an outlier. Secondly we will start with getting acquainted with the data by looking at the different relation between the feature variables and the target variable. thirdly we are going to compute various machine learning algorithms going from a simple linear model to more sophisticated algorithms such as gradient descent or K-nearest neighbors Finally we will add additional features that we think is relevant to our data set and see if they improve our predictions.

2 Data modification and visualisation

In this section our aim is to extract as much information as we can from the data in order to have a better understanding.

2.1 First Glimpse of the data and first cleaning

```
     id              vendor_id     pickup_datetime    dropoff_datetime   passenger_count
Length:1458644     Min.   :1.000   Length:1458644     Length:1458644     Min.   :0.000
Class :character   1st Qu.:1.000   Class :character   Class :character   1st Qu.:1.000
Mode  :character   Median :2.000   Mode  :character   Mode  :character   Median :1.000
                   Mean   :1.535                                         Mean   :1.665
                   3rd Qu.:2.000                                         3rd Qu.:2.000
                   Max.   :2.000                                         Max.   :9.000
pickup_longitude  pickup_latitude dropoff_longitude dropoff_latitude store_and_fwd_flag
Min.   :-121.93   Min.   :34.36   Min.   :-121.93   Min.   :32.18    Length:1458644
1st Qu.: -73.99   1st Qu.:40.74   1st Qu.: -73.99   1st Qu.:40.74    Class :character
Median : -73.98   Median :40.75   Median : -73.98   Median :40.75    Mode  :character
Mean   : -73.97   Mean   :40.75   Mean   : -73.97   Mean   :40.75
3rd Qu.: -73.97   3rd Qu.:40.77   3rd Qu.: -73.96   3rd Qu.:40.77
Max.   : -61.34   Max.   :51.88   Max.   : -61.34   Max.   :43.92
trip_duration
Min.   :      1
1st Qu.:    397
Median :    662
Mean   :    959
3rd Qu.:   1075
Max.   :3526282
```

FIGURE 1 – Summary of the training data set

Thanks to the summary function we have a first glimpse into our data set . We can see for example that the longest ride last 3526282 seconds which corresponds to approximately 979 hours (that's definitely too long)

and the shortest ride last 1s (which is too short). We also see tat some of the feature variables are characters so we will have to modify them. In this first cleaning we are going to get rid of extremes values. We decide to remove all observation with a trip duration shorter than 1 minute and longer than 6 hours. We are also going to modify the datetime format for a better manipulation.

2.2 Visualization of the different features and their relation with the target variable

Visualisations of feature distributions and their relations are really helpfuf for better understanding a data set, and they often open up new lines of inquiry. Examinig the data in multiple forms enables us to see which feature variables are going to be relevent to our model. It also helps to notice even subtle trends and correlations.

2.2.1 distributions of each individual feature

In this section we will begin by having a look at the distributions of the individual data features.

2.2.1.1 Localisation

We start with a map of New York to see where the rides take place and distances in question. We used the leaflet package of R for this visualization.

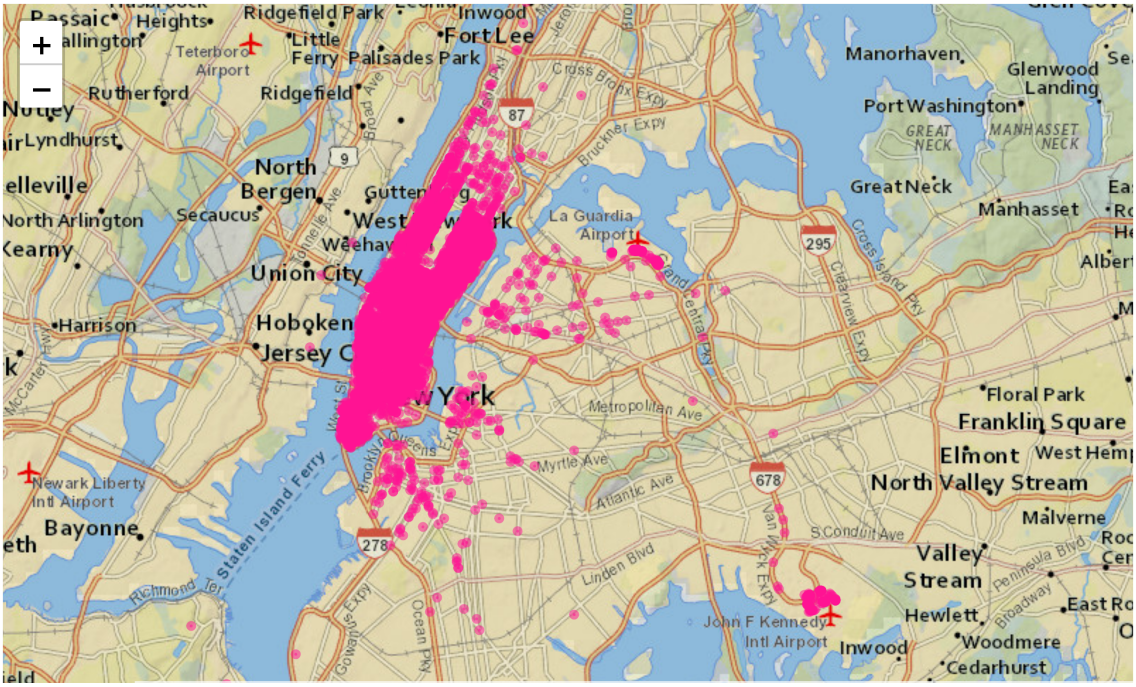

FIGURE 2 – maps of the location of the trips

In turns out that almost all of our trips were in fact taking place in Manhattan only. Another notable hot-spot is JFK airport towards the south-east of the city. The map gives us an idea what some of our distributions could look like.

2.2.1.2 Distribution of the target variable

For our first look into the target variable , we've plotted the frequencies of the trip duration and the log of the the trip duration. The majority of rides follow a rather smooth distribution that looks almost log-normal with a

peak just short of 1000 seconds, i.e. about 27 minutes.

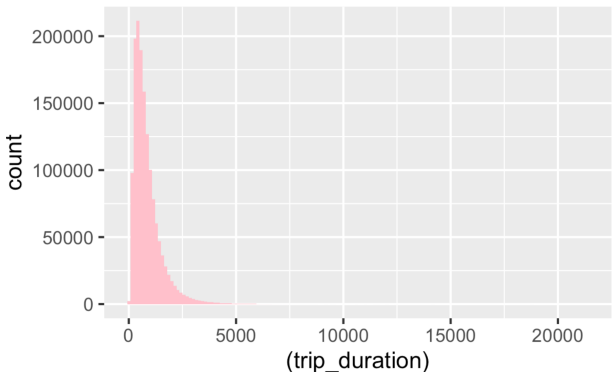

FIGURE 3 – Distribution of the target variable

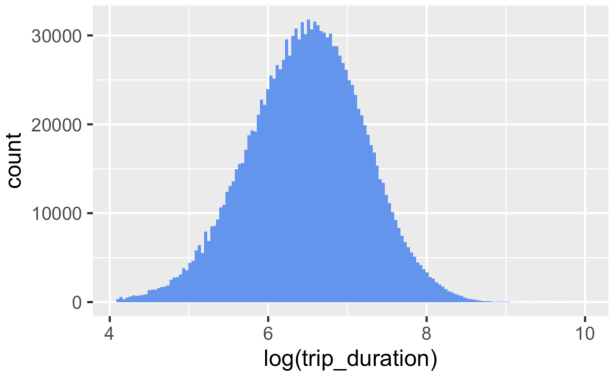

FIGURE 4 – Distribution of the log of target variable

In the plot below we can already see some daily and weekly modulations in the number of trips. Let's investigate these variations.

2.2.1.3 number of trips according to the number of passengers , the flag and the vendor

We can see that the majority of rides concern only one customers. There are few courses with 0 customers and 8 or 9 customers. The store-and-fwd-flag values, indicating whether the trip data was sent immediately to the vendor ("N") or held in the memory of the taxi because there was no connection to the server ("Y"), show that there was almost no storing taking place .

We also the number of courses realised by each company. We can see that the vendor two realised approximately 200000 courses more than the vendor 1.

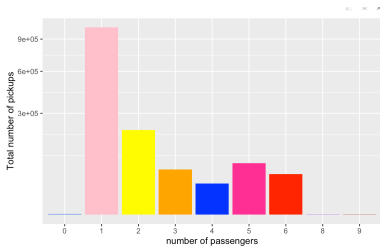

FIGURE 5 – number of trips according to the number of passengers

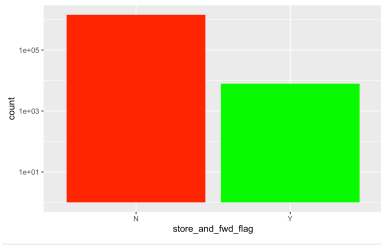

FIGURE 6 – number of trips according to the flag

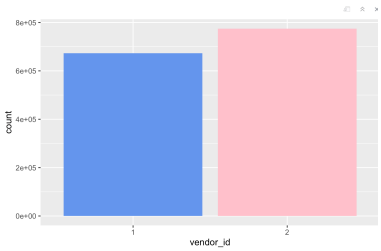

FIGURE 7 – number of trips according to the vendor

2.2.1.4 number of trips according to the day of the week and the hour of the day

We notice that the distribution of the number of rides according to the week day is similar for the 2 vendors. The most busy day is friday with about 120000 rides for the vendor 2 and 105000 rides for the vendor 1. Monday is the calmest day with the least number of courses with around 85000 rides for the vendor 1 and 100000 courses for the vendor 2 For the number of pick ups according to the hour, again the two vendors have very similar

distributions . The busiest hour seems to be between 5 and 6 pm and the calmest around 5 am. As one would intuitively expect, there is a strong dip during the early morning hours. We find another dip around 4pm and then the numbers increase towards the evening.

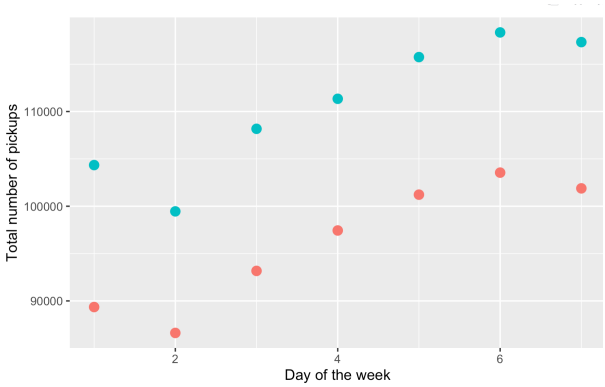

FIGURE 8 – number of trips according to the day of the week

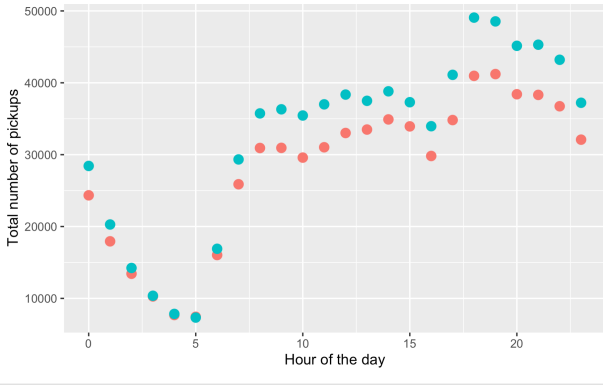

FIGURE 9 – number of trips according to the hour of the day

2.2.1.5 number of trips according to the hour for each day of the week and each month of the year

We find that January and June have fewer trips, whereas March and April are busier months. This tendency is observed for both vendor-ids.We can also notice that the distribution of trips for each month are very similar. The distribution of the number of trips according to the hour of the day is quite different for each day. The weekend (Saturday and Sunday, plus Friday to an extend) have higher trip numbers during the early morning ours but lower ones in the morning between 5 and 10, which can most likely be attributed to the contrast between NYC business days and weekend night life. In addition, trip numbers drop on Sunday evening/night. Now that we've looked into the different distribution of the features variables we are going to study their relation with the target variable

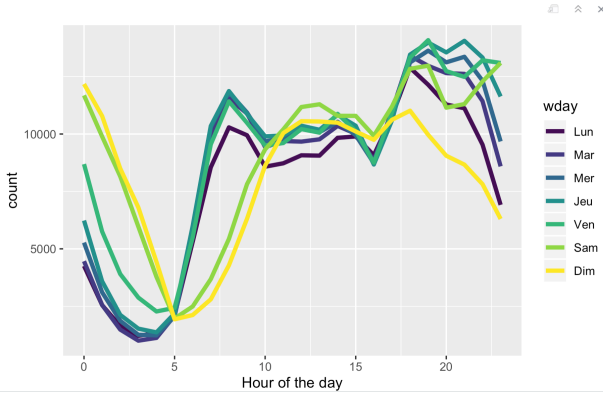

FIGURE 10 – superimposition of number of trips according to the day of the week

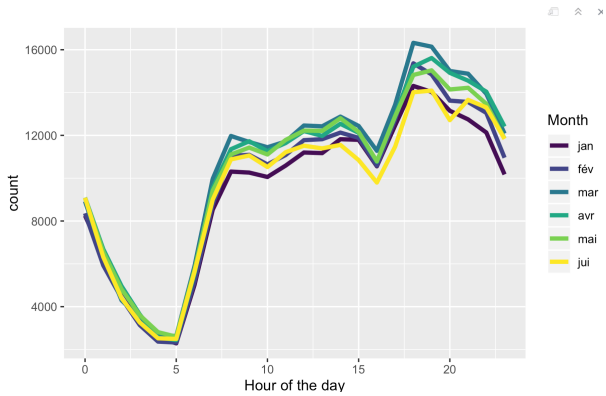

FIGURE 11 – superimposition of number of trips according to the hour of the day

2.2.2 Relation between the features variables and the target variable

While the previous section looked primarily at the distributions of the individual features, here we will examine in more detail how those features are related to each other and to our target trip-duration.

2.2.2.1 Pickup date/time vs trip-duration

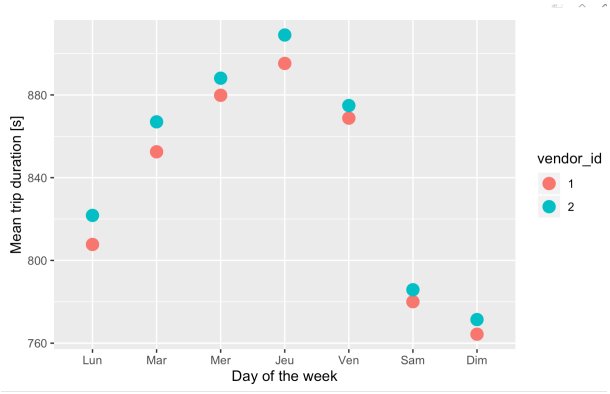

FIGURE 12 – mean duration of the trips according to the day of the week

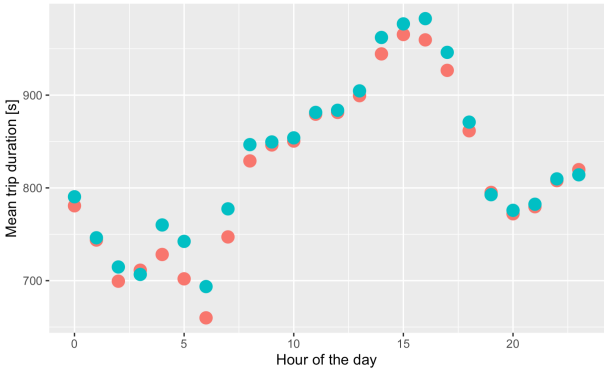

FIGURE 13 – mean duration of the trips according to the hour of the day

Previously we've seen that Thursdays were the second busiest day for both vendors they also have longer trips in average. Mondays were the calmest days however they don't have the fastest trips in average. This could mean that the calmer day doesn't lead to faster trips or in the opposite busiest days does not necessarily mean longer trips. We also notice that the vendor 2 which have the more frequent trips has also the longest trip duration . Over the course of a typical day we find a peak in the early afternoon and dips around 5-6am and 8pm. According to these two plots we can conclude that The weekday and hour of a trip appear to be important features for predicting its duration and should be included in a successful model. Also since the vendor 2 has higher trip duration. It will be worth adding the vendor-id feature to a model to test its predictive importance.

2.2.2.2 Passenger count and Vendor-id vs trip-duration

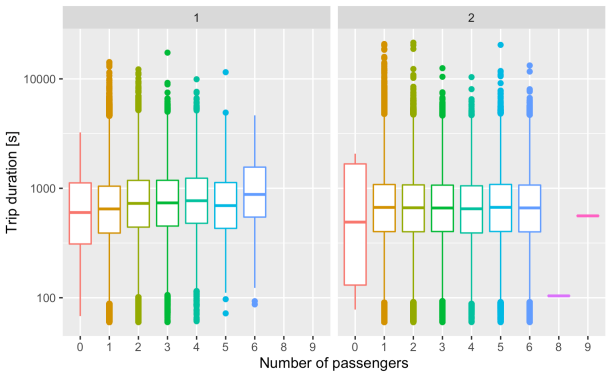

FIGURE 14 – trip duration of the trips according to the number of passenger and the vendor

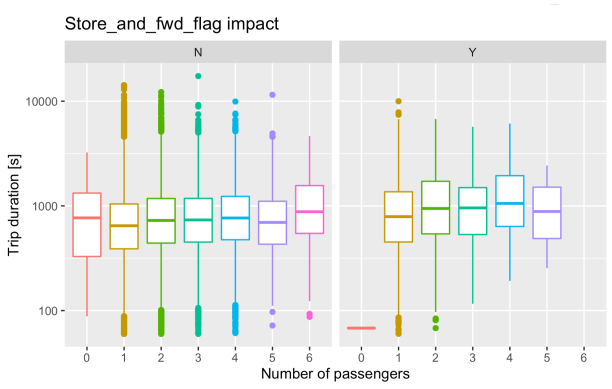

FIGURE 15 – trip duration of the trips according to the number of passenger and flag status

First of all. we notice that only the vendor 2 realise courses with more than 6 passengers. Between 1 and 6 passengers the median of trip durations are remarkably similar, in particular for vendor 2. There might be differences for vendor 1, but they are small The number of passengers doesn't seem to have a big effect on the trip duration. We also find that there is no overwhelming differences between the stored and non-stored trips. The stored ones might be slightly longer, though.

2.2.3 Adding relevant Feature : the direct distance of the trip

The trip duration is very likely to be correlated with the distance between the pick up and drop off location so we are goin to add a feature corresponding to the distance using the geolocalisation cooridnates and the R package geosphere As one would have expected the duration of the trip increases with the direct distance since we have a log-normal distribution for our target,We decided to use to transform our target into its log and do the same thing for the distance since there seems to be a linear relation between the two of them.

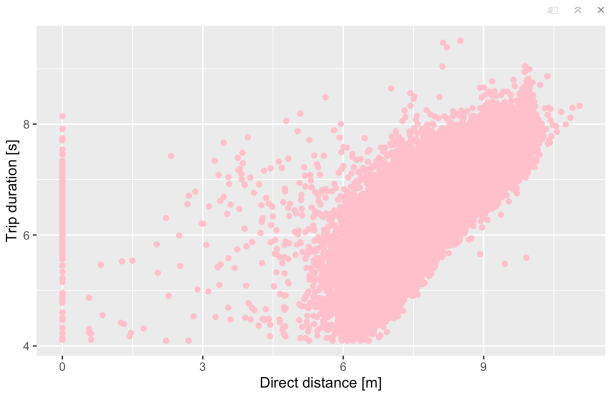

FIGURE 16 – The log(trips duration) according to the log(distance)

2.2.4 Last cleaning , final training dataset

According to all the remark we have made previously here are all the modification we did on the dataset

1. It seems a little unbelievable that a trip can last more than a day so we decide to set the threshold for trip duration to 6 hours. We also got rid of trips lasting 0 seconds
2. We've seen that the trip durtion follows a log normal distribution and that scaling th evaraible by taking its log give us a better representation of the correlation that exist in the data set so we we have decided to work with the log of trip duration as our target variable
3. Since we've seen that there seem to be a linear relation between the distance and the trip duration we've also transformed the distance variable accordingly .
4. We have seen that the flag status doesn't seem to have any effect on the target variable so we took it off the dataseet.

5. Since there only few trips with more than 6 passengers we've also set the threshold for the passenger count to 6

6. We've seen that the hour of the day, the day of the week and th months have an impact on our target variable so we decide to extract those information from the Pick-up date time and add a colum for each of them.

2.2.5 correlation plot

On the correlation plot of the remainign variables we can see that the target variable(trip duration) has the biggest correlation with the distance and the geolalisation coordinates.

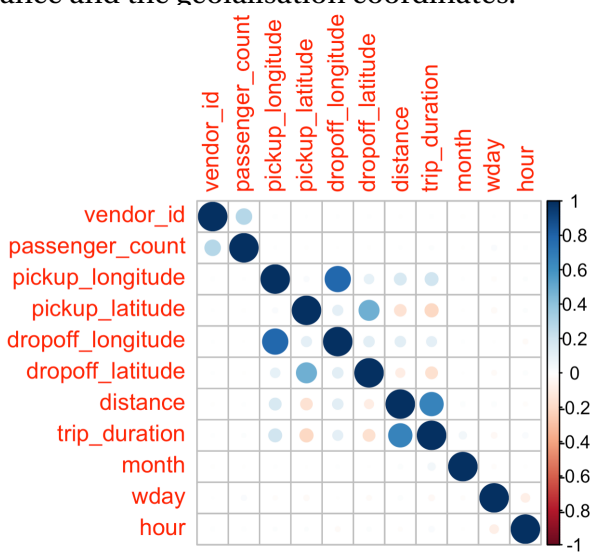

FIGURE 17 – Correlation plot of our final training dataset

Modelization Part

On this section we are goign to compute various machine learning algorithm in order to find the best prediction for our target variable i.e the one who gives the smallest score.

3 Baseline model : Ordinary Least square method

In statistics, ordinary least squares (OLS) is a type of linear least squares method for estimating the unknown parameters in a linear regression model. OLS chooses the parameters of a linear function of a set of explanatory variables by the principle of least squares : minimizing the sum of the squares of the differences between the observed dependent variable (values of the variable being observed) in the given dataset and those predicted by the linear function. Consider an overdetermined system

$\sum_{j=1}^{p} X_{ij}\beta_j = y_i,\ (i = 1,2,\ldots,n)$, of n linear equations in p unknown coefficients, 1,2,...,p, with n > p. (Note : for a linear model as above, not all of

X contains information on the data points. The first column is populated with ones,

$X_{i1} = 1$, only the other columns contain actual data, so here p = number of regressors + 1.) This can be written in matrix form as

$$\mathbf{X\beta} = \mathbf{y}, \text{where } \mathbf{X} = \begin{bmatrix} X_{11} & X_{12} & \cdots & X_{1p} \\ X_{21} & X_{22} & \cdots & X_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ X_{n1} & X_{n2} & \cdots & X_{np} \end{bmatrix}$$

$$\boldsymbol{\beta} = ^{\mathrm{T}} (\beta_1 \beta_2 \dots \beta_p) \qquad \mathbf{y} = ^{\mathrm{T}} (y_1 y_2 \dots y_n)$$

Such a system usually has no exact solution, so the goal is instead to find the coefficients $\boldsymbol{\beta}$ which fit the equations "best", in the sense of solving the quadratic minimization problem $\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} S(\boldsymbol{\beta})$, where the objective function S is given by $S(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left| y_i - \sum_{j=1}^{p} X_{ij}\beta_j \right|^2 = \|\mathbf{y} - \mathbf{X\beta}\|^2$. This minimization problem has a unique solution, provided that the p columns of the matrix X are linearly independent, given by solving the normal equations $(\mathbf{X}^{\mathrm{T}}\mathbf{X})\hat{\boldsymbol{\beta}} = \mathbf{X}^{\mathrm{T}}\mathbf{y}$. That formula enables us to compute the prediction for our test data set Here is the score we obtained

FIGURE 18 – Score :OLS method

4 Feature selection : Backward and stepwise regression

We've seen previoulsy in the corrplot that there some feature variables which are correlated to each other Leading us to think that we should coupure feature selection. Stepwise regression is a method of selecting independent variables in order to choose a set of predictors that have the best relationship with the dependent variable. Suppose that we have a statistical model of some data. Let k be the number of estimated parameters in the model. In the R package the step function uses the AIC criterion for weighing the choices, which takes proper account of the number of parameters fit; at each step an add or drop will be performed that minimizes the AIC score. Let $\hat{L}$ be the maximum value of the likelihood function for the model. Then the AIC value of the model is the following $AIC = 2k - 2\ln(\hat{L})$ Given a set of candidate models for the data, the preferred model is the one with the minimum AIC value. We've performed Stepwise and backward feature Selection. However, the feature regression didn't perform a variable selection for both bakward and stepwise. The lowest AIC was equal to -2304382 and was obtain with a model containing all the variables like the baseline model

```
Start:  AIC=-2304382
train$trip_duration ~ vendor_id + passenger_count + pickup_longitude +
    pickup_latitude + dropoff_longitude + dropoff_latitude +
    distance + month + wday + hour

                   Df Sum of Sq    RSS      AIC
<none>                          292630 -2304382
- pickup_latitude    1         4 292634 -2304366
- vendor_id          1         4 292634 -2304366
- pickup_longitude   1        20 292650 -2304285
- passenger_count    1        47 292677 -2304154
- dropoff_longitude  1       171 292801 -2303541
- month              1       996 293626 -2299478
- wday               1      1470 294100 -2297149
- hour               1      1672 294302 -2296159
- dropoff_latitude   1      2132 294762 -2293904
- distance           1    423905 716535 -1011463
```

FIGURE 19 – Result backward regression

5 Stochastic Gradient Descent (SGD)

At a theoretical level, gradient descent is an algorithm that minimizes functions. Given a function defined by a set of parameters, gradient descent starts with an initial set of parameter values and iteratively moves toward a set of parameter values that minimize the function. This iterative minimization is achieved using calculus, taking steps in the negative direction of the function gradient.The term stochastic derives from the fact that the gradient based on a single training sample is a stochastic approximation of the whole cost gradient. Due to its stochastic nature, the path towards the global cost minimum is not direct, as in GD, but may zigzag if we visualize the cost surface in a 2D space. However, it has been shown that SGD almost surely converges to the global cost minimum if the cost function is convex (or pseudo-convex). In other words, in a full batch GD, the gradient is computed for the full dataset, whereas SGD takes a single sample and performs gradient calculations. It can also take mini-batches and perform the calculations. One advantage of SGD is faster computation of gradients. To perform regression analysis with the SGD algorithm in R, we can use the THE sgd package. Strangely, we obtain a higher score than before.

FIGURE 20 – Score :Stochastic Gradient Descent regression

6 Penalized regression methods

As an alternative to the selection methods discussed in the previous sections, we adopt methods that use all predictors but bind or adjust the coefficients by bringing them to very small or zero values (shrinkage). Actually,when a large number of variables are available, the least square estimates of a linear model often have a low bias but a high variance with respect to models with fewer variables. Under these conditions, there is

an overfitting problem. The best way to address the problem of overfitting is to modify the estimation method by neglecting the requirement of an unbiased parameter estimator and instead considering the possibility of using a biased estimator, which may have smaller variance. There are several biased estimators, most based on regularization : Ridge, Lasso, and ElasticNet are the most popular methods. To perform regression analysis with the penalized regression methods in R, we can use the THE **glmnet** package.

6.1 Lasso

Lasso regression find a parsimonious model which performs L1 regularization. The L1 regularization adds a penality equivalent to the absolute of the maginitude of regression coefficients and tries to minimize them Lasso translates each coefficient by a constant factor λ, truncating at zero. This is called "soft thresholding".We use the cv.glmnet.This function does k-fold cross-validation for glmnet, produces a plot, and returns a value for the best lambda value which is in our case $\lambda_{lasso} = 0.001019089$. When we performed Lasso regression **we got a public score of : 0.55010.**

FIGURE 21 – Score : Lasso regression

6.2 Ridge

Ridge regression is a parsimonious model which performs L2 regularization. The L2 regularization adds a penality equivalent to the square of the maginitude of regression coefficients and tries to minimize them.Ridge regression does a proportional shrinkage and handles collinear variables but it does not perform a selection.We use the cv.glmnet() function available in the glmnet package to find the best λ. This function does k-fold cross-validation for glmnet, produces a plot, and returns a value for the best lambda value which is equal in our case to $\lambda_{ridge} = 0.0569753$. Using the ridge regression,**we obtain a better public score of :0.54551 .**

FIGURE 22 – Score : Ridge regression

6.3 Elastic Net

ElasticNet is a hybrid of both Lasso and Ridge regression. It is trained with both L1-norm and L2-norm prior as regularizer. Like LASSO regularization it results in sparse solutions, however it also has the advantage of performing well with highly correlated variables like ridge regularization. Elastic net is used by solving the following optimization problem : $min_x \left\| y - Ax \right\| + \lambda_1 \left\| x \right\|_1 + \lambda_2 \left\| x \right\|_2$. Ultimately, we can say that ElasticNet balance the trade-off bias-variance with the choice of λ. It assumes that part of the coefficients are zero, or at least not significant. When computed, the Elastic Net regularization give us **a public score of : 0.55**.

As for the other regularized methods the function cv.glmnet compute a cross validation to determine the best

FIGURE 23 – Score : Elastic Net regression

lambda which is equal to $\lambda_{ElasticNet} = 0.001857111$.

To sum up, the best score obtained throw penalized methods is around : 0.54 (Ridge).That led us to try a new method of regression :KNN regression.

7 KNN regression

k-Nearest Neighbors (k-NN) is an algorithm that is useful for making classifications/predictions when there are potential non-linear boundaries separating classes or values of interest. Conceptually, k-NN examines the classes/values of the points around it (i.e., its neighbors) to determine the value of the point of interest. The majority or average value will be assigned to the point of interest. A simple implementation of KNN regression is to calculate the average of the numerical target of the K nearest neighbors. Another approach uses an inverse distance weighted average of the K nearest neighbors. KNN regression uses the same distance functions as KNN classification.We tried to find the most optimal k value by choosing different ones randomly and repeating the test multiple times till we find a minimal error. When we performed KNN regression **we got an even better public score of : 0.49035 for K=25.** The best model here which give the best score (around 0.49) is the model

FIGURE 24 – Score : KNN regression

obtained by knn regression.

8 Adding external data

8.1 Weather data in New York City - 2016

Always in order to get a better understanding of our dataset, we have decided to expand it and add external information that has not been taken in consideration although it can be regarded as factors to our target changes. This new data is the Weather in New York City for the year 2016. Indeed the atmospheric conditions can comprise our target in terms of temperature and wind and clouds and precipitation.

8.1.1 Data modification and visualisation

Before getting started, it is necessary to correct and modify the new dataset in order for it to be homogeneous with our initial one.The changes we made are :

— Disposing of all the **missing 'T'** values by replacing them with 0.01 which we deemed to be a correct rate to represent the trace of precipitation.

— Adding an **"All precipitation"**column that sums the values of both the "Rain" and "Snow fall".

— Adding two more boolean columns that represent whether it has **rained/snowed** or not and then converting them into integers.

Let us take a glimpse on initial modified dataset after being blended with this external one.

```
Observations: 1,443,750
Variables: 18
$ vendor_id          <int> 2, 1, 2, 2, 2, 2, 1, 2, 1, 2, 2, 2, 2, 2,…
$ passenger_count    <int> 1, 1, 1, 1, 1, 6, 4, 1, 1, 1, 1, 4, 2, 1,…
$ pickup_longitude   <dbl> -73.98215, -73.98042, -73.97903, -74.0100…
$ pickup_latitude    <dbl> 40.76794, 40.73856, 40.76394, 40.71997, 4…
$ dropoff_longitude  <dbl> -73.96463, -73.99948, -74.00533, -74.0122…
$ dropoff_latitude   <dbl> 40.76560, 40.73115, 40.71009, 40.70672, 4…
$ trip_duration      <dbl> 6.120297, 6.496775, 7.661056, 6.061457, 6…
$ distance           <dbl> 7.315333, 7.500894, 8.761031, 7.302923, 7…
$ month              <dbl> 3, 6, 1, 4, 3, 1, 6, 5, 5, 3, 5, 5, 2, 6,…
$ wday               <dbl> 2, 1, 3, 4, 7, 7, 6, 7, 6, 5, 3, 1, 6, 4,…
$ hour               <int> 17, 0, 11, 19, 13, 22, 22, 7, 23, 21, 22,…
$ rain               <dbl> 0.29, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,…
$ s_fall             <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,…
$ has_snow           <int> 1, 1, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1,…
$ has_rain           <int> 2, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1,…
$ s_depth            <dbl> 0.00, 0.00, 0.01, 0.00, 0.00, 6.00, 0.00,…
$ max_temp           <int> 51, 83, 28, 48, 55, 39, 78, 66, 87, 79, 6…
$ min_temp           <int> 40, 62, 16, 30, 38, 28, 63, 54, 73, 63, 5…
```

FIGURE 25 – Expanded dataset

8.1.2 The OLS and the penalized regression methods

Now, we shall do over all the methods mentioned above and compare the results we get from both our datasets seeking in this way the best score, thus the best model.For better results, we used the following logarithmic model :

— Changing trip duration with ln(trip duration)

— Changing distance with ln(distance +1)

8.1.2.1 OLS and penalized regression methods scores

submitENwlog.csv 34 minutes to go EN - log - weather	0.55013	0.54945
submitridgewlog.csv 33 minutes to go Ridge - log - weather	0.54514	0.54496
submitlassowlog.csv 31 minutes to go Lasso- log - weather	0.55025	0.54956
submitolsweatherlog.csv 30 minutes to go OLS log weather	0.55036	0.54965

FIGURE 26 – Ridge,Lasso,OLS and Elastic net methods scores

8.1.2.2 Initial dataset VS Expanded dataset

We notice that no matter the method, expanding the dataset and adding more information improved our scores even if only by little. We can also see that using the logarithmic model enabled us to decrease the error even more. However, keeping all the results found so far in mind, none of these methods has given us the best score. We should then move on and try another model; the KNN regression.

	Initdata	Initdata_log	Expdata	Expdata_log	Diff	Diff_log
OLS	0.58901	0.55021	0.58835	0.54965	**0.00066**	**0.00056**
Ridge	0.60448	0.54551	0.60379	0.54496	**0.00069**	**0.00055**
Lasso	0.58925	0.55010	0.58852	0.54956	**0.00073**	**0.00054**
Elastic net	0.58949	0.55000	0.58870	0.54945	**0.00079**	**0.00055**

FIGURE 27 – Comparison between the datasets and the methods (public score)

8.1.3 KNN regression

As for the k-nearest neighbors method, we tried to find the most optimal k value by choosing different ones randomly and repeating the test multiple times till we find a minimal error. This was our only option knowing that we could not rely on the accuracy to figure out the best k for lack of the test's target values.We got the following results :

FIGURE 28 – KNN regression for k=50

8.1.3.1 Initial dataset VS Expanded one

We notice that contrary to the penalized regression methods, for the KNN method, adding information got our error to increase remarkably (+($\approx 10^{-2}$)) in comparison with the last models (-($\approx 10^{-4}$)) as we can see on the figure 14 and 16. Nevertheless, this method is the one that got us the best score so far with an error of **0.49137** for **k=17**, using our initial dataset.

	k=17	k=25	k=50	k=100
Init data	0.49137	0.49147	0.49396	0.49673
Exp data	0.50274	0.50297	0.51115	0.52803
Diff	**-0.01137**	**-0.01150**	**-0.01719**	**-0.03130**

FIGURE 29 – Initial dataset VS Expanded dataset (public score)

9 Comparision of the Regression models

To compare these different methods we will use the k-fold method, by dividing our initial training database into two subsets, one train set (80% of the training dataset) on which to develop our models and a testing set (20%) on which we will compare which models give the best results and then apply that model on the original test dataset from kaggle. Cross-validation is the process of assessing how the results of a statistical analysis will generalize to an independent data set. If the model has been estimated over some, but not all, of the available data, then the model using the estimated parameters can be used to predict the held-back data. If, for example, the out-of-sample mean squared error, also known as the mean squared prediction error, is substantially higher than the in-sample mean square error, this is a sign of deficiency in the model. For each model,we split randomly the initial dataset in k folds. The k-1 folds are used to train our model (train dataset) and the remaining observations are used as the 'Test' data set). We use the training data set to estimate the parameters of the model and compute the RMSE of the model. Given the previous model,we use the test data set to compute the RMSE to evaluate the performances of the model. We compare the results obtained for each model with help of boxplots. Here the best model is the model with the smallest error.

10 Performance and ranking on Kaggle

After trying out many machine learning algorithms we have come to the conclusion that the best predictions was obtain with the k nearest neighbors algorithm applied on the initial data set without the weather data, **using the logarithmic transformation** with k equal to 25. We obtained a score of **0,49035 and were ranked approximately 771th** .

FIGURE 30 – score obtained with KNN regression

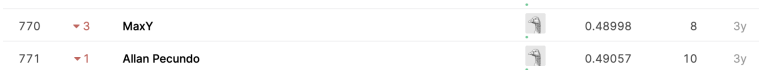

FIGURE 31 – approximate ranking

11 Annex : R Code

```
#uploading the different libraries
library('lubridate') # date and time
library('geosphere') # geospatial locations
library("glmnet")
library('alluvial') # visualisation
library('dplyr') # data manipulation
library('readr') # input/output
library('data.table') # data manipulation
library('tibble') # data wrangling
library('tidyr') # data wrangling
library('leaflet') # maps
library('leaflet.extras') # maps
library('maps') # maps
library('caret') # modelling
library('stringr') # string manipulation
library('forcats') # factor manipulation
library('ggplot2') # visualisation
library('scales') # visualisation
library('grid') # visualisation
library('RColorBrewer') # visualisation
library('corrplot') # visualisation
#uploading the data we use fread instead of read.table because its way faster
train <- as_tibble(fread("/Users/princessemame/APPRENTISSAGE AUTOMATIQUE /train.csv"))
#looking into the data with the summary function
as.table(summary(train))
#getting rid of the id column
train <- train[,-1]
#getting rid of extreme values for trip duration
train <- subset(train,train$trip_duration<21600)
train <-subset(train,train$trip_duration>=60)
#transforming the date and time into workable data and the vendor_id and passenger_count
#into factors ( for visualization purposes )
train <- train %>% #this is a pipe meaning that you give the object train to the mutate function
```

```
  mutate(pickup_datetime = ymd_hms(pickup_datetime),
         dropoff_datetime = ymd_hms(dropoff_datetime),
         vendor_id = factor(vendor_id),
         passenger_count = factor(passenger_count))
#selecting randomly 8000 observation in our data set and placing them on the map
#of new york city to see the repartition of rides
set.seed(1234)
foo <- sample_n(train, 8e3)
leaflet (data = foo) %>% addProviderTiles("Esri.NatGeoWorldMap") %>%
  addCircleMarkers(~ pickup_longitude, ~pickup_latitude, radius = 1,
                   color = "deeppink", fillOpacity = 0.5)
#looking the distribution of the trip  duration vector by plotting its  histogram using ggplot
train  %>%
  ggplot(aes((trip_duration))) +
  geom_histogram(fill = "pink", bins = 150)
#looking the distribution of the trip  duration vector by plotting the histogram of its logarithm using ggplot
train  %>%
  ggplot(aes(log(trip_duration))) +
  geom_histogram(fill = "cornflowerblue", bins = 150)
#visualisation of  the number of trips for each of passenger_counts
p1 <- train %>%
  group_by(passenger_count) %>%
  count() %>%
  ggplot(aes(passenger_count, n, fill = passenger_count)) +# this give a color for each passenger count (which we
      already transformed into a factor)
  geom_col(fill=c("cornflowerblue","pink", "yellow", "orange","blue","deeppink","red","purple","brown")) +
  scale_y_sqrt() +# transforming the y_axis into the root square of counts of trips
  theme(legend.position = "none")+ labs(x = "number of passengers", y = "Total number of pickups")
#visualisation of  the number of trips with for each of vendors_id
p2 <- train %>%
  ggplot(aes(vendor_id, fill  = vendor_id)) +
  geom_bar(fill=c("cornflowerblue","pink")) +
  theme(legend.position = "none")
#visualisation of  the number of trips with for each flag status
p3 <- train %>%
```

```
  ggplot(aes(store_and_fwd_flag)) +
  geom_bar(fill=c("red","green")) +
  theme(legend.position = "none") +
  scale_y_log10()
#visualisation of the number of trips for each vendors_id according to the week day
p4 <- train %>%
  mutate(wday = wday(pickup_datetime
  )) %>%
  group_by(wday, vendor_id) %>%# this mean for each day and each vendor id we sum the number of trips that
      hapened in that day
  count() %>%
  ggplot(aes(wday, n, colour = vendor_id)) +
  geom_point(size = 3) +
  labs(x = "Day of the week", y = "Total number of pickups") +
  theme(legend.position = "none")
#visualisation of the number of trips for each vendors_id according to the hour of day
p5 <- train %>%
  mutate(hpick = hour(pickup_datetime)) %>%
  group_by(hpick, vendor_id) %>%# this mean for each hour and each vendor id we sum the number of trips that
      hapened in that hour
  count() %>%
  ggplot(aes(hpick, n, color = vendor_id)) +
  geom_point(size = 3) +
  labs(x = "Hour of the day", y = "Total number of pickups") +
  theme(legend.position = "none")
p1;p2;p3;p4;p5
#visualisation of the number of trips for each vendors_id according to the hour of day for each month
p1 <- train %>%
  mutate(hpick = hour(pickup_datetime),
         Month = factor(month(pickup_datetime, label = TRUE))) %>%
  group_by(hpick, Month) %>%
  count() %>%
  ggplot(aes(hpick, n, color = Month)) +
  geom_line(size = 1.5) +
  labs(x = "Hour of the day", y = "count")
```

```
#visualisation of the number of trips for each vendors_id according to the hour of day for each week day
p2 <- train %>%
  mutate(hpick = hour(pickup_datetime),
         wday = factor(wday(pickup_datetime, label = TRUE, week_start = 1))) %>%
  group_by(hpick, wday) %>%
  count() %>%
  ggplot(aes(hpick, n, color = wday)) +
  geom_line(size = 1.5) +
  labs(x = "Hour of the day", y = "count")
p1;p2
#visualisation of the mean of trip duration accoridng to the week day
p1 <- train %>%
  mutate(wday = wday(pickup_datetime, label = TRUE, week_start = 1)) %>%
  group_by(wday, vendor_id) %>%
  summarise(mean_duration = mean(trip_duration)) %>%
  ggplot(aes(wday, mean_duration, color = vendor_id)) +
  geom_point(size = 4) +
  labs(x = "Day of the week", y = "Mean trip duration [s]")
#visualisation of the mean of trip duration accoridng to the hour of day
p2 <- train %>%
  mutate(hpick = hour(pickup_datetime)) %>%
  group_by(hpick, vendor_id) %>%
  summarise(mean_duration = mean(trip_duration)) %>%
  ggplot(aes(hpick, mean_duration, color = vendor_id)) +
  geom_point(size = 4) +
  labs(x = "Hour of the day", y = "Mean trip duration [s]") +
  theme(legend.position = "none")
p1;p2
#visualisation of the boxplot of duration accoridng for each vendor id
train %>%
  ggplot(aes(passenger_count, trip_duration, color = passenger_count)) +
  geom_boxplot() +
  scale_y_log10() +
  theme(legend.position = "none") +
  facet_wrap(~ vendor_id) +
```

```
  labs(y = "Trip duration [s]", x = "Number of passengers")
#visualisation of the boxplot of duration accoridng for each passenger count
train %>%
  filter (vendor_id == 1) %>%
  ggplot(aes(passenger_count, trip_duration, color = passenger_count)) +
  geom_boxplot() +
  scale_y_log10() +
  facet_wrap(~ store_and_fwd_flag) +
  theme(legend.position = "none") +
  labs(y = "Trip duration [s]", x = "Number of passengers")
#calculating the distance between the pick up and the drop off
library(geosphere)
n <- nrow(train)
#recupeation of the latitude and longitude columns
latA <-as.matrix(train[,6])
lonA <- as.matrix(train[,5])
latB <- as.matrix(train[,8])
lonB <- as.matrix(train[,7])
#putting the pick up geolocation into a matrix
p1 <- matrix(data = c(lonA,latA),nrow=n, ncol=2)
#putting the dropoff geolocation into a matrix
p2<- matrix(data = c(lonB,latB),nrow=n, ncol=2)
#computing the distance with the distgeo function which give the distance in meter
distance <- distGeo(p1,p2)
#taking the log of the distance +1 (to avid taking the log of 0 ) because we transformed our target variable into its log
train$distance <- log(distance+1)
#adding the coloumn distance to the train data
train <- cbind(train,distance)
# switch des colonnes distance et duree
a <- c(colnames(train))
b <- a
b[11] <- a[12]
b[12 ] <- a[11]
#getting rid of extreme distance which are clearly outside of ne york
train <- subset(train,train$distance<=log(100000))
```

```r
#plotting the relation between the log of the distance and the log of trip duration
set.seed(3)
train %>%
  sample_n(5e4) %>%#taking a random sample of size 5000
  ggplot(aes((distance), log(trip_duration)) )+
  geom_point()+
  labs(x = "Direct distance [m]", y = "Trip duration [s]")
#adding the columns corresponding to the day month ad hour of pick up into the data set and transforming the trip
    durantion into its log
train <- train %>%
  mutate(month = month(pickup_datetime, label = TRUE),
         wday = wday(pickup_datetime, label = TRUE, week_start = 1),
         wday = fct_relevel(wday, c("Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun")),
         hour=hour(pickup_datetime),
         trip_duration=log(trip_duration)
  )
#getting rid of obsevation with passenger_count greater than 7
subset(train, train$passenger_count<7)
#getting rid of the date time values since we already have the hour mont and day columns
#getting rid of the store flag column
train <- train[,-c(2, 3, 9)]
#reputting passenger count into a numeric values we do minus 1 because the factor start at 1 instead of
train$passenger_count <- as.numeric(train$passenger_count)-1
#same for wday and month
train$wday <- as.numeric(train$wday)
train$month <- as.numeric(train$month)
#correlation plot of our data
corrplot(cor(data.matrix(train)))
#linear regression using lm function. We take trip duration as a linear function of all the other variables
modreg_R1 <- lm(train$trip_duration~.,data=train)
#using summary to see the differnt coefficient obtained previously
summary(modreg_R1)
#adding the intercept column
intercept <- rep(1,nrow(train))
#making the X matrxi composed of the intercept and the explanatory variable
```

```
X=as.matrix(cbind(intercept,train[,c(1:6,8:11)]))
#taking the beta vector which corresponds to the coefficient of the linear relation
beta=t(t(modreg_R1$coefficients))
#uplooading the test and the submit data set
test  <- as_tibble(fread("/Users/princessemame/APPRENTISSAGE AUTOMATIQUE /test.csv"))
submit<- as_tibble(fread("/Users/princessemame/APPRENTISSAGE AUTOMATIQUE /sample_submission.csv"))
#Remaking the same trnsformation we did on the train data set on the test data set
test  <- test[,-1]
test  <- test %>%
  mutate(pickup_datetime = ymd_hms(pickup_datetime))
library(geosphere)
n <- nrow(test)
latA <-as.matrix(test$pickup_latitude)
lonA <- as.matrix(test$pickup_longitude)
latB <- as.matrix(test$dropoff_latitude)
lonB <- as.matrix(test$dropoff_longitude)
p1 <- matrix(data = c(lonA,latA),nrow=n, ncol=2)
p2<- matrix(data = c(lonB,latB),nrow=n, ncol=2)
distance <- distGeo(p1,p2)
distance <- log(distance+1)
test  <- cbind(test,distance)
test  <- test%>%
  mutate(store_and_fwd_flag = as.integer(factor(store_and_fwd_flag)),
         vendor_id = as.integer(vendor_id),
         month = (month(pickup_datetime)),
         hour = hour(pickup_datetime),
         wday = wday(pickup_datetime, label = TRUE, abbr = TRUE) , wday = as.integer(fct_relevel(wday, c("Sun",
           "Sat", "Mon", "Tue", "Wed", "Thu", "Fri"))),
  )
test$wday <- as.numeric(test$wday)
test$month <- as.numeric(test$month)
test  <- test[,-c(2, 8)]
#creating the X for the test  dataset
X=as.matrix(cbind(rep(1,(nrow(test))), test))
beta=t(t(modreg_R1$coefficients))
```

```
#computing the prediction by doing X*beta
Y_hat=X%*%beta
#writting the submit files
submit$trip_duration <- exp(Y_hat) # we take exponnetial cause we were using the log of trip duration
write.csv(x = submit, file  = "sample_submission3.csv",row.names = FALSE)
#Feature Selection
#stepwise regression using the step function the starting model is that the target is constant and the last model is the
    one containing all the variables
regboth <- step(lm(train$trip_duration~1,data=train),list(upper=modreg_R1),direction = "both")
#backward regression using the step function the starting model is is the one containing all the variables
regbackward <- step(modreg_R1,direction = "backward")
#Stochastic gradient descent(SGD)
library(sgd)
model_train3<-train[,-c(2,3,13)]
#we run stochastic gradient descent on the underlying loss function using our model.
sgd.theta <- sgd(model_train3$trip_duration ~ ., data=model_train3, model="lm")
#creating the X for  the test  dataset
X=as.matrix(cbind(rep(1,length(nrow(model_test))),model_test))
beta=t(t(sgd.theta$coefficients))
#computing the prediction by doing X*beta
Y_sgd=exp(X%*%beta)
#writting the submit files
sample_submission_data$trip_duration <- Y_sgd
write.csv(x = sample_submission_data, file = "submitsgd.csv",row.names = FALSE)
#Penalized regression methods
#Lasso regression
library(glmnet)
#Lasso regression find a parsimonious model which performs L1 regularization. The L1 regularization adds a penality
    equivalent to the absolute of the maginitude of regression coefficients and tries to minimize them
#Lasso translates each coefficient by a constant factor $\lambda$, truncating at zero. This is called "soft thresholding".
modreg1=cv.glmnet(as.matrix(model_train3[,-7]),model_train3$trip_duration,
    alpha=1,nlambda=100,lambda.min.ratio=0.0001)
#This function does k-fold cross-validation for glmnet, returns a value for the best lambda value.
best_lam <- modreg1$lambda.min
lasso_best <- glmnet(as.matrix(model_train3[,-7]),model_train3$trip_duration, alpha = 1, lambda = best_lam)
```

```
#computing the prediction using "predict"
Y_test<- exp(predict(lasso_best, s = best_lam, newx = as.matrix(model_test)))
#writting the submit files
sample_submission_data$trip_duration <- Y_test
write.csv(x = sample_submission_data, file = "submitlasso.csv",row.names = FALSE)
#Ridge regression
#Ridge regression is a parsimonious model which performs L2 regularization. The L2 regularization adds a penality
    equivalent to the square of the maginitude of regression coefficients and tries to minimize them.
#Ridge regression does a proportional shrinkage and handles collinear variables.
modreg2=cv.glmnet(as.matrix(model_train3[,-7]),
                  model_train3$trip_duration,alpha=0,
                  nlambda=100,lambda.min.ratio=0.0001)
#This function does k-fold cross-validation for glmnet, returns a value for the best lambda value.
best_lambda=modreg2$lambda.min
best_ridge= glmnet(as.matrix(model_train3[,-7]),model_train3$trip_duration,alpha=0,lambda=best_lambda)
#computing the prediction using "predict"
Y_testridge=exp(predict(best_ridge, s =best_lambda, newx = as.matrix(model_test)))
#writting the submit files
sample_submission_data$trip_duration <-Y_testridge
write.csv(x = sample_submission_data, file = "submitridge.csv",row.names = FALSE)
#Elastic Net
#ElasticNet is a hybrid of both Lasso and Ridge regression. It is trained with both L1-norm and L2-norm prior as
    regularizer.
modreg3=cv.glmnet(as.matrix(model_train3[,-7]),
      model_train3$trip_duration,alpha=0.5,
      nlambda=100,lambda.min.ratio=0.0001)
best_lam <- modreg3$lambda.min
EN_best <- glmnet(as.matrix(model_train3[,-7]),
                  model_train3$trip_duration,alpha=0.5,
                  lambda = best_lam)
#computing the prediction using "predict"
Y_test_EN<- exp(predict(EN_best, s = best_lam,newx = as.matrix(model_test)))
#writting the submit files
sample_submission_data$trip_duration <- Y_test_EN
write.csv(x = sample_submission_data, file = "submitEN.csv",row.names = FALSE)
```

```
#k−Nearest Neighbors(KNN)
#K nearest neighbors is a simple algorithm that stores all available cases and predict the numerical target based on a
    similarity measure
prediction <− FNN::knn.reg(model_train3[,−7],model_test,
                           model_train3$trip_duration, k=25,
                           algorithm="kd_tree")
#computing the prediction using "predict"
Y_KNN <− exp(prediction$pred)
#writting the submit files
sample_submission_data$trip_duration <− Y_KNN
write.csv(x = sample_submission_data, file = "submitKNN.csv",row.names = FALSE)
#Externe data
#Downloading the dataset from kaggle : Weather data in New York City − 2016
#Uploading the dataset
weather<−data.table::fread(file="C:/Users/nisri/Downloads/weather_data_nyc_centralpark_2016(1).csv",
                           sep=",",header=TRUE)
#Adding new columns that dispose of all the missing 'T' values by replacing them with 0.01.
#Adding an All precipitation column that sums the values of both the "Rain" and "Snow fall".
#Adding two more boolean columns that represent whether it has rained/snowed.
weather <− weather %>%
  mutate(date = dmy(date),
         rain = as.numeric(ifelse(precipitation == "T", "0.01", precipitation)),
         s_fall = as.numeric(ifelse(`snow fall` == "T", "0.01", `snow fall`)),
         s_depth = as.numeric(ifelse(`snow depth` == "T", "0.01", `snow depth`)),
         all_precip = s_fall + rain,
         has_snow = (s_fall > 0) | (s_depth > 0),
         has_rain = rain > 0,
         max_temp = `maximum temperature`,
         min_temp = `minimum temperature`)
#Keeping only the columns we added/modified.
foo <− weather %>%
  select(date, rain, s_fall , all_precip, has_snow, has_rain, s_depth, max_temp, min_temp)
#Adding the new columns/information to the initial dataset to create an expanded one.
model_train2 <− left_join(model_train2, foo, by = "date")
#Disposing of the pickuptime/dropofftime/date columns
```

```
model_train4<-model_train2[,-c(2,3,13)]
#Converting the boolean values into integers.
model_train4$has_snow<-as.integer(as.factor(model_train4$has_snow))
model_train4$has_rain<-as.integer(as.factor(model_train4$has_rain))
#Disposiong of the all precipitation column
model_train4<-model_train4[,-14]
#Doing the same manipulations to the test dataset
model_test2 <- left_join(test_data1, foo, by = "date")
model_test2<-model_test2[,-c(1,3,9,14,17)]
model_test2$has_snow<-as.integer(as.factor(model_test2$has_snow))
model_test2$has_rain<-as.integer(as.factor(model_test2$has_rain))
##Using the same code for the OLS-RIDGE-LASSO-ELASTIC NET and KNN.
#COMPARAISON DES MODELES DE REGRESSIONS
#calculate the RMSE



data2 <- sample_n(train,1000)


RMSE1=c()
RMSE_lm=c()
residu_lm=matrix(nrow =200, ncol=20)
for (i in 1:20){
  smp_size = floor(0.8 * nrow(data2))
  set.seed(10+200*i)
  train_ind = sample(seq_len(nrow(data2)),size = smp_size)
  TabTrain =data2[train_ind,]
  TabTest=data2[-train_ind,]
  modreg1= lm(TabTrain$trip_duration~.,data=TabTrain)
  Y_test=predict(modreg1,newdata=TabTest[,-7],interval="confidence")
  Y_predicted=data.frame(Y_test)$fit
  Y_reel=TabTest$trip_duration
  residu_lm[,i]=Y_reel-Y_predicted
  RMSE_lm[i]=sqrt(((sum(residu_lm[,i])**2))/length(TabTest))
}
```

```
RMSE_ridge=c()
residu_ridge=matrix(nrow =200, ncol=20)
for (i in 1:20){
  smp_size = floor(0.8 * nrow(data2))
  set.seed(200*i+10)
  train_ind = sample(seq_len(nrow(data2)),size = smp_size)

  TabTrain =data2[train_ind,]
  TabTest=data2[-train_ind,]

  modreg1=cv.glmnet(data.matrix(TabTrain[,c(1:6,8:11)]),TabTrain$trip_duration,alpha=0,nlambda=100,lambda.min.ratio=0.0

  best_lambda=modreg1$lambda.min

  best_ridge=glmnet(data.matrix(TabTrain[,-7]),TabTrain$trip_duration ,lambda=best_lambda)

  pred=predict(best_ridge, s = best_lambda, newx= data.matrix(TabTest[,c(1:6,8:11)]))
  Y_predicted=data.matrix(pred)
  Y_reel=TabTest$trip_duration
  residu_ridge[,i]=Y_reel-t(Y_predicted)
  RMSE_ridge[i]=sqrt(((sum(residu_ridge[,i])**2))/length(TabTest))
}

RMSE_lasso=c()
residu_lasso=matrix(nrow =200, ncol=20)
for (i in 1:20){
  lambdaseq <- seq(0,50,0.01)
  smp_size = floor(0.8 * nrow(data2))
  set.seed(200*i+10)
  train_ind = sample(seq_len(nrow(data2)),size = smp_size)
  TabTrain =data2[train_ind,]
  TabTest=data2[-train_ind,]
  modreg1=cv.glmnet(data.matrix(TabTrain[,-7]), TabTrain$trip_duration, alpha=1, lambda=lambdaseq)
  best_lam <- modreg1$lambda.min
  lasso_best <- glmnet(data.matrix(TabTrain[,-7]), TabTrain$trip_duration, alpha = 1, lambda = best_lam)
```

```
  Y_test<- predict(lasso_best, s = best_lam, newx = data.matrix(TabTest[,c(1:6,8:11)]))
  Y_predicted=data.frame(Y_test)
  Y_reel=TabTest$trip_duration
  residu_lasso[,i]=(Y_reel)-t(Y_predicted)
  RMSE_lasso[i]=sqrt(((sum(residu_lasso[,i])**2))/length(TabTest))
}
RMSE_EN=c()
residu_EN=matrix(nrow =200, ncol=20)
for (i in 1:20){
  lambdaseq <- seq(0,50,0.01)
  smp_size = floor(0.8 * nrow(data2))
  set.seed(200*i+10)
  train_ind = sample(seq_len(nrow(data2)),size = smp_size)
  TabTrain =data2[train_ind,]
  TabTest=data2[-train_ind,]
  modreg1=cv.glmnet(data.matrix(TabTrain[,-7]), TabTrain$trip_duration
      ,alpha=0.5,nlambda=100,lambda.min.ratio=0.0001)
  best_lam <- modreg1$lambda.min
  EN_best <- glmnet(data.matrix(TabTrain[,c(1:6,8:11)]), TabTrain$trip_duration,alpha=0.5, lambda = best_lam)
  Y_test<- predict(EN_best, s = best_lam, newx = data.matrix(TabTest[,c(1:6,8:11)]))
  Y_predicted=data.frame(Y_test)
  Y_reel=TabTest$trip_duration
  residu_EN[,i]=(Y_reel)-t(Y_predicted)
  RMSE_EN[i]=sqrt(((sum(residu_EN[,i])**2))/length(TabTest))
}

library(FNN)
RMSE1=c()
RMSE_KNN=c()
residu_KNN=matrix(nrow =200, ncol=20)
for (i in 1:1){
  smp_size = floor(0.8 * nrow(data2))
  set.seed(10+200*i)
  train_ind = sample(seq_len(nrow(data2)),size = smp_size)
  TabTrain =data2[train_ind,]
```

```
  TabTest=data2[-train_ind,]
  prediction <- FNN::knn.reg(data.matrix(TabTrain[,c(1:6,8:11)]),data.matrix(TabTest[,-7]), TabTrain$trip_duration, k
      = 25, algorithm="kd_tree")
  Y_predicted <- as.data.frame(prediction$pred)
  Y_reel=TabTest$trip_duration

  residu_KNN[,i]=Y_reel-Y_predicted
  RMSE_KNN[i]=sqrt(((sum(residu_KNN[,i])**2))/length(TabTest))
}

group=c("lm","Lasso","Ridge","ElasticNet","KNN")
boxplot(RMSE_lm,RMSE_lasso,RMSE_ridge,RMSE_EN,RMSE_KNN,names=group,notch=F, outlier.color = "red",
    outlier.shape = 8, outlier.size = 4, col=(c("darkgreen","purple","pink","blue","red")))
grid()
```