Prediction of the delay of a given flight

Aprenentatge Automàtic 1

Lucía De Pineda

Adriana Díaz

June 2022

INDEX

1. DESCRIPTION OF THE WORK, GOALS AND DATA

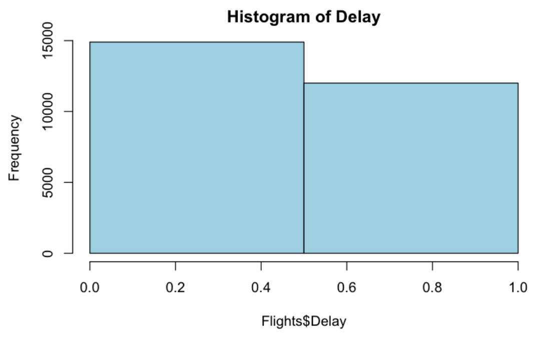
This project consists in predicting whether a given flight will be delayed or not, given the information of the scheduled departure. Therefore, the objective is to classify each flight as delayed or not delayed, which is a binary classification.

The dataset used to carry out this project is the Airlines Dataset. The source of this dataset is the Data Expo Competition (2009), it is inspired by the regression dataset from Elena Ikonomovska (http://kt.ijs.si/elena_ikonomovska/data.html). The original data consists of flight arrival and departure details for all commercial flights within the USA, from October 1987 to April 2008. For our dataset, the task was downsampled to 5 percent.

The Airlines dataset has 26969 instances. For each instance there are 8 variables: Airline, Flight, AirportFrom, AirportTo, DayOfWeek, Time and Length. The target variable is Delay, which is the one we want to predict. The description of the variables is the following:

- Airline: Categorical variable with 18 distinct values, which are the codes for each airline.
- Flight: Categorical variable with 5857 distinct values, the different possible flight numbers.
- AirportFrom: Categorical variable with 283 distinct values, the codes for all the airports of departure.
- AirportTo: Categorical variable with 285 distinct values, the codes for all the airports of departure.
- DayOfWeek: Categorical variable with 7 distinct values, one for each day of the week (1-Monday and 7-Sunday).
- Length: Numeric variable, distance between the two airports.
- Time: Numeric variable, time of the flight in minutes.

These are the features we are going to use to predict the target variable Delay, which is a categorical variable with two classes (binary feature). The classes are 0 (not delayed) or 1 (delayed). There are 14934 observations of class 0 (0.55%) and 12035 of class 1 (0.45%).



With all this data we will make our predictions, but for doing that first we need to process the data and analyze it. Then we will choose a model to carry out the classification of our problem.

2. DATA EXPLORATION PROCESS

2.1. Data pre-processing

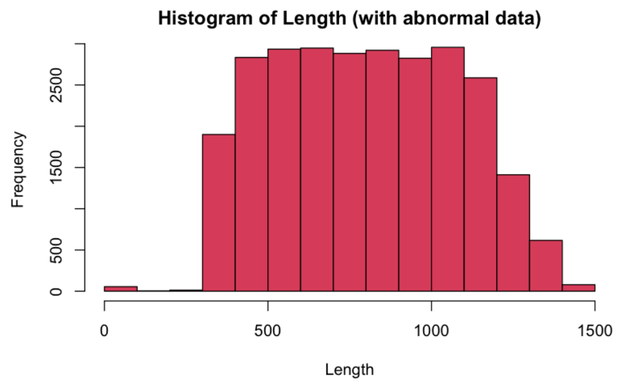
The first step of our data exploration process is the data pre-processing, which is a necessary step before any analytical work. Its objective is to prepare and shape the dataset for further analysis . The pre-process approach, in what concerns data cleaning and preparation, is different for each dataset depending on its characteristics, but it is crucial because it can have a deep impact on future performance and results.

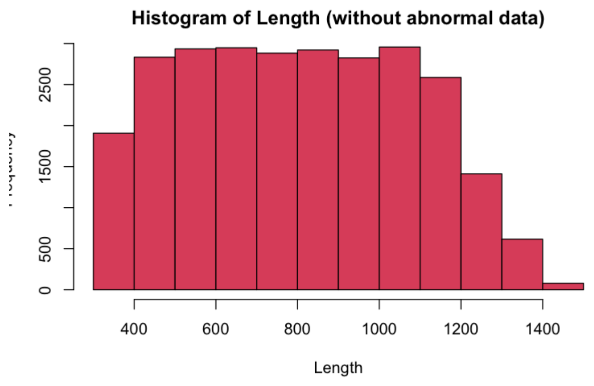
The usual steps of this pre-process are: treatment of missing values, treatment of anomalous values (outliers), treatment of incoherent or incorrect values, coding of non-continuous or non-ordered variables, feature selection (possible elimination of irrelevant or redundant variables), feature extraction (creation of new variables), normalization of the variables and transformation of them.

To begin with, we load our dataset to RStudio and we save it into a dataframe called Flights. Then, we name the variables and we print the summary to see the values for each variable. We analyze the minimum and maximum values, together with the mean, so that we can see if there are anomalous values. We also count the frequency of values for the target value and we observe that there are 14934 flights without delay and 12035 with delay, so it is approximately half and half.

Our data showed that there were no missing values, but we confirm it using the is.na() function, which reports if there is any missing value.

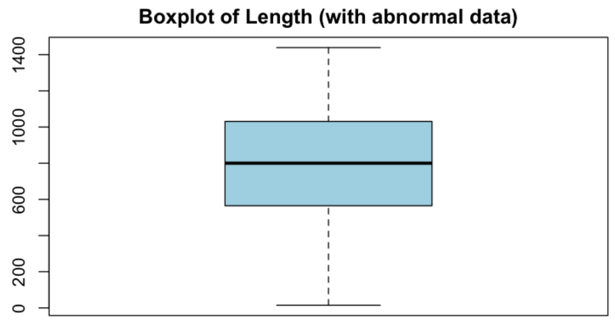
Secondly, when printing the summary, it could be seen that there were some anomalous values in the length section. Since the minimum value (15) was far from the mean (approximately 800), the possibility of it being an outlier had to be taken into account. Since these values were very distant from all the others and did not make sense, they were eliminated. The results are:

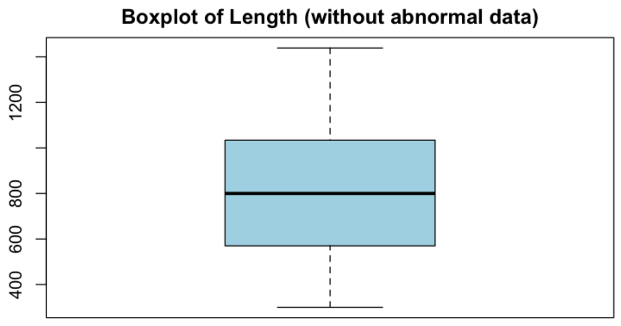




We can see that when eliminating the abnormal values the histogram is more uniformly distributed and without distant values.

If we plot each boxplot we obtain:





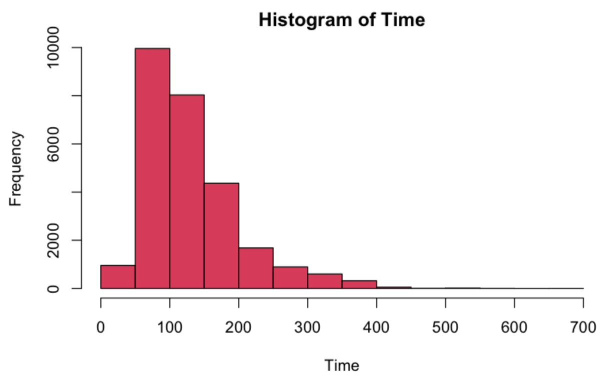
Once again, we observe that by eliminating the abnormal values, the minimum value is now closer to the mean. Also, the range of values is smaller now.

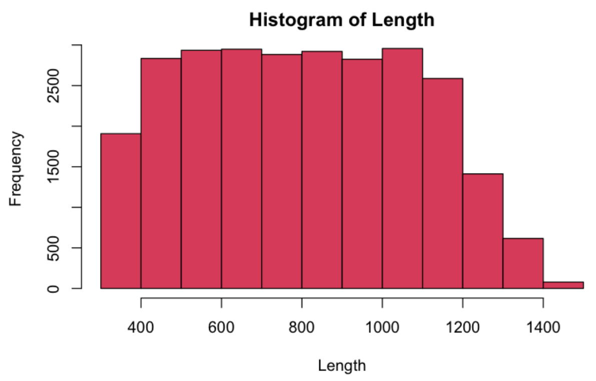
Nevertheless, it was considered that all the variables were important so the elimination of redundant or irrelevant variables was not carried out.

The next step was to code those non-continuous or non-ordered variables. The categorical variables are Airline, Flight, AirportFrom, AirportTo, DayOfWeek and Delay, so all of them were treated correctly being explicitly declared as categorical with the R function as.factor.

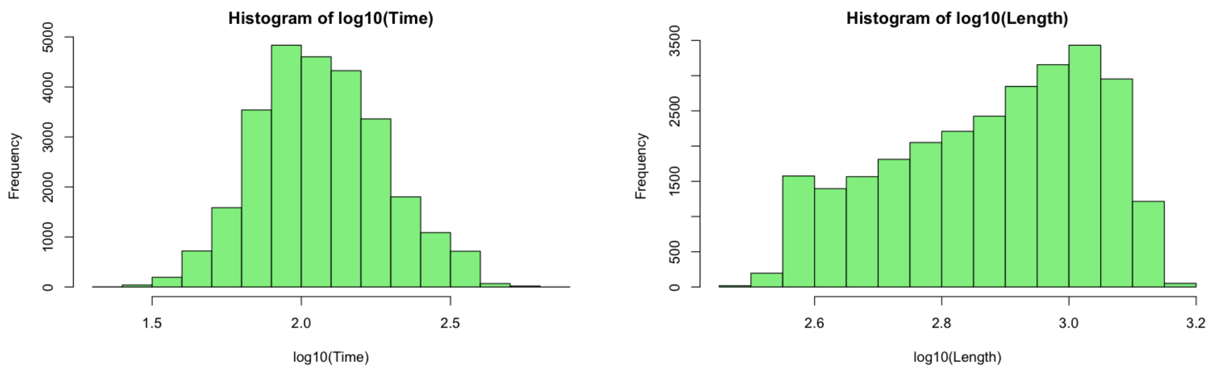
Given our case, we determined that it could be sensible to derive new variables, also known as feature extraction. We considered extracting the continuous variable speed. Firstly, we decided to convert time from minutes to hours and length from miles to kilometers so that it could be more understandable from us given the fact that we constantly deal with these units.

Since it is useful to have an idea of the empirical distribution of the dataset's variables, we did some histograms for the continuous variables (Time and Length).





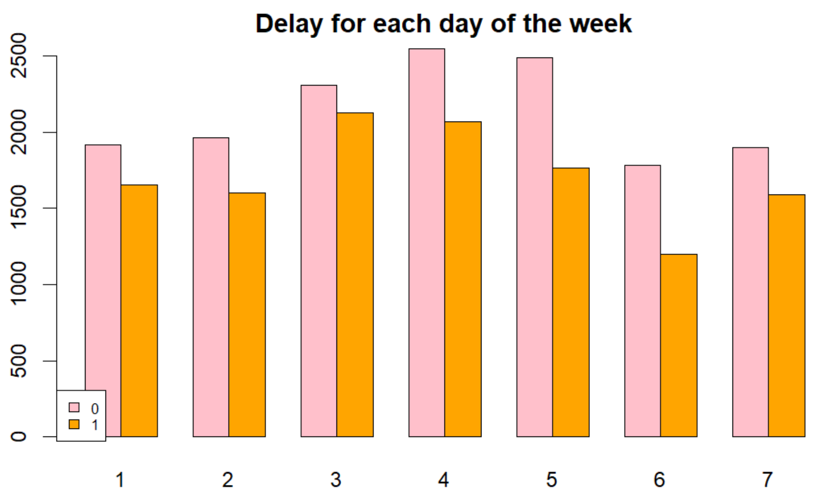
The plots obtained suggested taking logarithms on both variables so we plot them too with base 10.



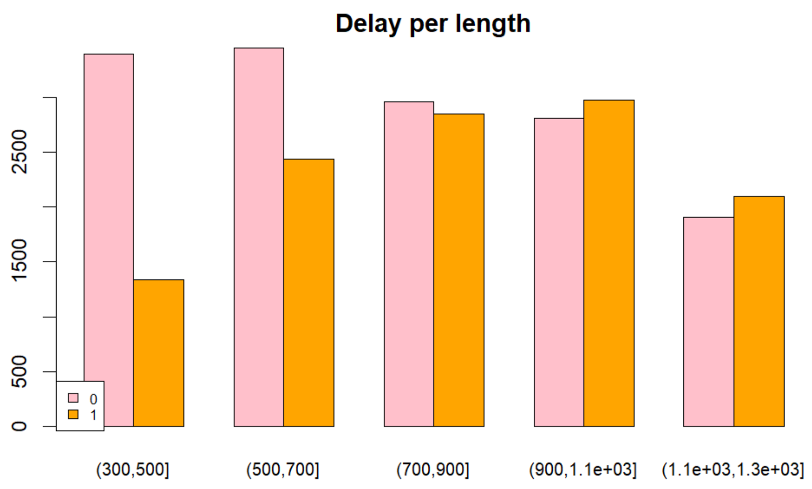
Now we can consider that the variable Time has a normal distribution. On the other hand, in the variable Length we see the shape of the half of a Gaussian distribution, so we also apply the logarithm.

2.2. Visualization of the data

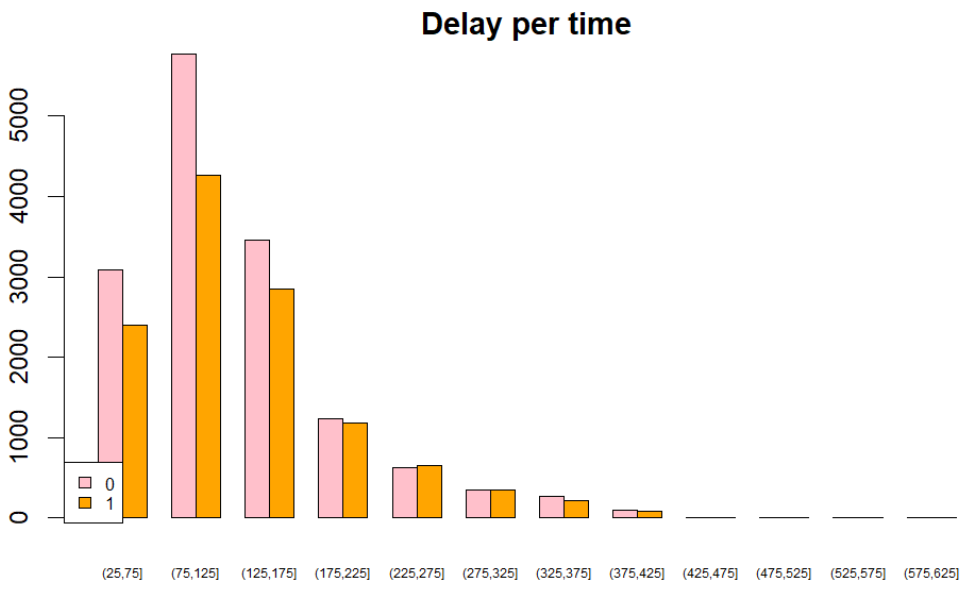
To have a better understanding of our dataset, we show some plots that represent different variables and their relationships. In particular, we considered it relevant to plot the relationship between the delay and the day of the week, time and length.



We can observe that the day with more delayed flights is Thursday, closely followed by Friday. On the other hand, the days with less delayed flights are Saturday and Sunday. Furthermore, the days that have more difference in the proportion of delayed versus not delayed flights are Friday and Saturday.



It can be seen that when the length of the flight is small, it is more probable that it won't be delayed than otherwise. Furthermore, the bigger the length, the higher the probability of the flights being delayed until 1101, length on which it decreases.



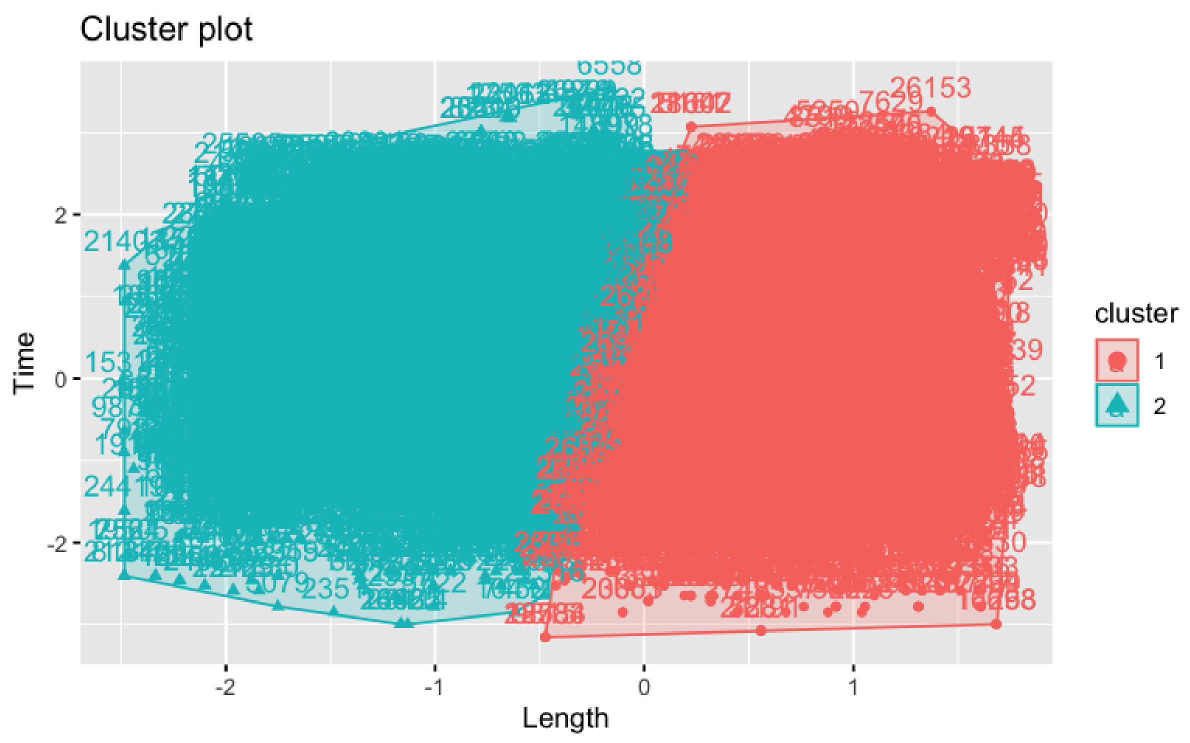
We can observe that from 176, the difference between being delayed and not is very slight. Furthermore, the rank of time that has more delayed flights is (75,125), which is also the one that has more difference between the two possibilities.

2.3. Clustering

The next process to carry out is the clustering of the data. Its goal is to find groups or clusters of similar data. When we have large amounts of unsupervised data, we want to find some patterns to group the data. In our case, the clustering should separate the data naturally into two groups, each of them corresponding to each class of the target variable.

We are going to use the method K-means, where we label each data vector as belonging to one of K clusters. We only have two classes, so K=2. When plotting the clusters, we should differentiate two clear groups of data. And, each cluster should correspond to one class. To perform clustering we only take the numeric variables (time and length). Moreover, when trying to run the code we came across an error involving the size of our data. It was not possible to do clustering with that many samples, so only for this part of the project we decided to take half of the data (the first 13000 observations). It is important to note that the proportions of the classes of the target variable are still the same (so it is a representative group of data).

Once we do the clustering assignment, we plot it. The result is:



We can observe two differentiated groups that should correspond to the two classes of our target. It appears that length is the variable that has more weight when separating the two classes. It is possible that the error is considerable, as we are only taking two of the eight variables into account.

3. RESAMPLING PROTOCOL AND MODELING METHODS CONSIDERED

In this section, we are going to use different modeling methods to find an appropriate model for our data, so we can predict our target variable. Later on, we will use resampling to estimate the true error of the model and, therefore, find the one best fitted for our data.

Since we only have one data sample and it is needed to accomplish three different tasks: to fit models to data (calculation of the model's coefficients or parameters), select one model if we have several candidates (model selection or hyper-parameter optimization) and estimate the error of the selected model (error estimation), it is required different (and independent) data samples. The basic resampling method consists firstly of fitting models (parameter optimization) with the training data. Afterwards, using the validation data to average prediction errors and choose the model with the lowest prediction error. Subsequently, the chosen model is refitted using the full learning data (both training and validation data). Lastly, the test set is used to estimate the true performance of the chosen model.

An important observation is that when splitting the data, we check that the proportion of the two classes of the target variable remains the same (stratified partitions). In our case, we have decided to take 70% of the dataset for our learning data and 30% for the test set. Once we have done this, we are going to fit four different models to our data, and then choose the best one. To do so, we will perform cross-validation on each model to estimate the prediction error. Cross-validation is a method that divides the learning data into training and validation, it does this using different partitions and then takes the average error.

For this section, we will use Python instead of R. We will use the dataset we processed in the first part of the project. Also, one last thing we must do before fitting the models is transform our categorical variables into numbers, as it is needed for the models we are going to work with. To do that, we use the Python function get_dummies, which will convert the variables indicated into dummy variables. So, for each categorical variable we will have a dummy variable for each category. These variables only take value 0 or 1, indicating the presence or absence of that class. Now our dataset is ready to fit different models.

The modeling methods we are going to use are: logistic regression, random forest, support-vector machine and naive-bayes classifier.

First we are going to describe the main idea of each method and why it seems like a good fit for our classification problem. Then, we will proceed to fitting the models. For each modeling method, we will choose the model with the best hyperparameters (if it proceeds) and then we will compare their performance.

3.1 Logistic regression

Since our problem is a binary classification, a good option of Generalized Linear Model is the Logistic Regression, which is a process of modeling the probability of one event, in our case the flights being delayed (out of the two alternatives: being or not) as a function of the explanatory variables, taking place by using the log-odds. We chose this modeling method as it is easy to implement, as well as very fast and efficient to train.

3.2 Random forest

A random forest is a classification algorithm consisting of many decision trees. A decision tree predicts the target by learning simple decision rules. During the training it will split the data so it can learn these rules. This model is very fast to train and interpretable but it can overfit easily. In a random forest, each individual tree makes a class prediction and the class with the most votes becomes our model's prediction. It is simple, but at the same time it produces a great result for classification problems. It gets a more accurate and stable prediction and it avoids the overfitting of decision trees, so that's why we chose it.

3.3 Support-vector machine

SVM is an algorithm that can be used both to predict regression and classification problems, but it is mostly used for classification ones such as ours. The idea of the algorithm is to plot each data item as a point in n-dimensional space (where n is a number of features you have) and then, we perform classification by finding the hyper-plane that differentiates the two classes very well. However, SVM works with small datasets, so we will have to divide ours, as it can be extremely slow with a dataset the size of ours. On the other hand, it works really well with complex datasets so it is a good choice for us.

3.4 Naive-Bayes classifier

A Naive Bayes classifier is a probabilistic machine learning model that's used for classification task. The Naive-Bayes classifiers are based on applying Bayes' theorem, assuming strong independence between the features. That is, the presence of one particular feature does not affect the other. They are fast and easy to implement but their biggest disadvantage is that the requirement of predictors to be independent. In our case, we will assume that the attributes are, in fact, independent, and we will see how the model performs.

4. COMPARISON OF RESULTS FOR EACH MODEL

To compare each model, we used the metrics accuracy, precision, recall, F-measure and support. The description of them is the following:

The accuracy is calculated by dividing the number of correct predictions by the total number of predictions:

$$Accuracy = Correct\ predictions / Total\ predictions$$

On the other hand, the precision is the number of True Positives divided by the sum of the True Positives plus the False Positives:

$$Precision = True\ Positives\ /\ (True\ Positives + False\ Positives)$$

Therefore, the lower the precision the higher the number of false positives. In other words, it can be said that it indicates, when a model makes a prediction, how often it is correct.

The recall, on the other hand is measured by the True Positives divided by the sum of the True Positives plus the False Negatives:

$$Recall = True\ Positives\ /\ (True\ Positives + False\ Negatives)$$

Thus, low recall means that the model contains many False Negatives, so it won't correctly identify a large proportion of class members.

The F1-score is the Average between Precision and Recall. It should be noted that weights can be applied if one metric is more important than the other for a specific use case.

Finally, the Support is the Number of actual observations in that class.

The decision metric will be accuracy, since it has been proven that we are dealing with balanced data, so it is adequate. But we will also check the other metrics and see how the model performs.

We performed 5-fold cross-validation for each model fitted and we computed the mean of each metric, for it to be more representative. We decided on a 5-fold CV based on our experience and, more importantly, so the computational cost is not too high and the execution time too slow (as our dataset is large). The results are:

3.1 Logistic regression

To begin with, one of the Logistic Regression assumptions is that only meaningful variables should be included. Therefore, we checked if all of them were significant and removed the variable Flight since it had p-values smaller than 0.05. We didn't remove any other variables, since some of the dummy variables had significant values, so we considered them meaningful for the model. Then, we use cross validation and we compute the metrics:

	Accuracy	Precision	Recall (mean)	F1-score (mean)	Time(s)
Logistic Regression	0.633	0.635	0.615	0.609	4.083

These are the metrics for the logistic regression model, the validation accuracy is 63.3%. An advantage of this model is the speed of training, even though the dataset is quite large. Also, it is important to note that there are no critical hyperparameters to tune in this model.

3.2 Random forest

Now we fit our data to a Random Forest model. In addition to the metrics already introduced, for this model we will also use the Out-of-Bag (OOB) error. This error is a metric that we can compute on the Random Forest model while training. It is very useful because it allows us to perform model selection (tune hyperparameters) without having to execute costly cross-validations. However, we decided to also implement 5-fold cross-validation, as we consider it is more reliable and, in our case, it was not too slow. The results obtained with the default parameters are:

	Accuracy	Precision	Recall (mean)	F1-score (mean)	Time(s)
Random Forest	0.627	0.62	0.618	0.615	6.901

```
OOB accuracy= 0.6247252747252747
```

We see how OOB accuracy is almost the same as the true validation accuracy, as we already predicted.

In addition, we will try tuning the hyperparameters to improve our Random Forest model. We try different values of number of trees (number of estimators), maximum depth of the trees, the minimum number of samples required to split an internal node, the minimum number of samples required to be at a leaf node and the balance of the data. We find the best parameters (the ones increasing accuracy) and we fit a Random Forest model using them. The results are:

	Accuracy	Precision	Recall (mean)	F1-score (mean)	Time(s)
Random Forest	0.627	0.62	0.618	0.615	6.901
Random Forest-best	0.641	0.65	0.619	0.611	12.274

```
OOB accuracy= 0.6463736263736264
```

When tuning the hyperparameters we obtain a better model, so that is the Random Forest model we are going to use for comparison with the other modeling methods.

The parameters are: `{'class_weight': None, 'max_depth': None, 'min_samples_leaf': 4, 'min_samples_split': 4, 'n_estimators': 200}`

3.3 Support-vector machine

We are going to run our SVM with different parameters and then choose which is the best one taking into account the metrics computed. We start with the default parameters and then we will try to tune them. The results of the cross validation of the first model are:

	Accuracy	Precision	Recall (mean)	F1-score (mean)	Time(s)
SVM-default	0.55	0.275	0.5	0.355	56.198

Then, we try to balance the data using the function class_weight. The idea is we can weight the C hyperparameter based on the number of samples of each class, penalyzing this way the majoritary classes. The results for this case, added to the results table, are:

	Accuracy	Precision	Recall (mean)	F1-score (mean)	Time(s)
SVM-default	0.55	0.275	0.5	0.355	56.198
SVM-balanced	0.498	0.527	0.52	0.481	61.398

With these results, we decided to not try tuning more hyperparameters, as it was extremely slow to train and compute all the metrics for the models. We consider SVM is not the optimal modeling method for our problem, as it is too big and the execution time is too high. Moreover, the accuracy was lower than all the other models adjusted.

In conclusion, the best SVM is the one with the default parameters (C=1 and kernel=RBF) without balancing the data. However, it is not the model we are going to end up choosing, even though we will add it to the results table for comparison.

3.4 Naive-Bayes classifier

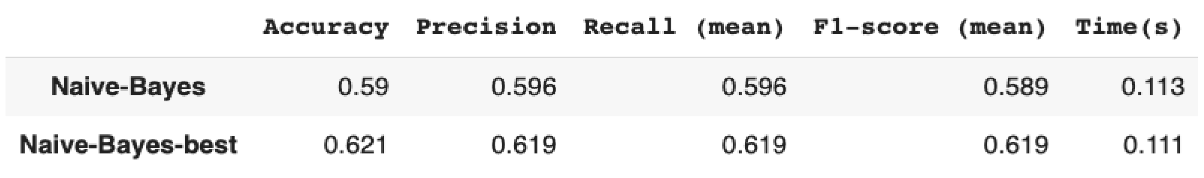
For the Naive-Bayes problem we have to take into account the zero probability problem (when there is no occurrence of a category in the training set but it appears in the test set and then the model assigns 0 probability to it, leading to incorrect calculation). To deal with this problem, we can use Laplace Smoothing. The Naive-Bayes model offers a hyperparameter to tune variance smoothing.

Furthermore, the model assumes that the data follows a normal distribution, which is usually not very probable. To solve this problem, we can perform a Power Transformation in our data, so that it is more or less normally distributed.

First we try with the default parameters of the Gaussian Naive-Bayes classifier. The results are:

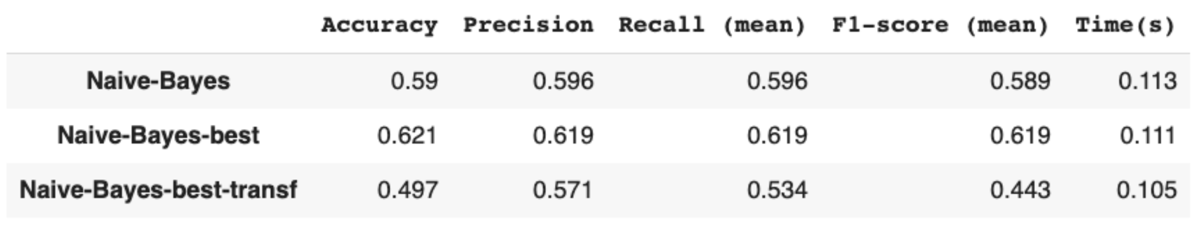
	Accuracy	Precision	Recall (mean)	F1-score (mean)	Time(s)
Naive-Bayes	0.59	0.596	0.596	0.589	0.113

Then, we try tuning the variance smoothing hyperparameter to improve the accuracy of the model. We tried 1.e+00, 1.e-01, 1.e-02, 1.e-03, 1.e-04, 1.e-05, 1.e-06, 1.e-07, 1.e-08 and 1.e-09. We found that the best variance smoothing was `6.579e-09`, and redoing the fitting we obtained:



	Accuracy	Precision	Recall (mean)	F1-score (mean)	Time(s)
Naive-Bayes	0.59	0.596	0.596	0.589	0.113
Naive-Bayes-best	0.621	0.619	0.619	0.619	0.111

We see an improvement of the model. Finally, we tried performing the Power Transformation to our training data, as well as using the hyperparameter tuning (the value found for the variance smoothing was `0.15199`). The results are the following:



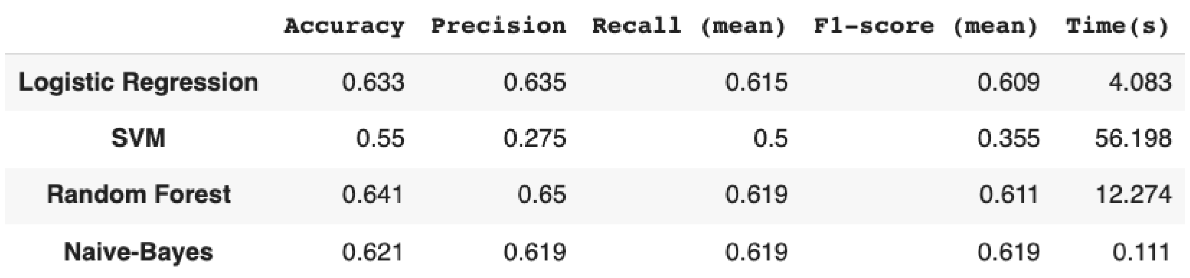
	Accuracy	Precision	Recall (mean)	F1-score (mean)	Time(s)
Naive-Bayes	0.59	0.596	0.596	0.589	0.113
Naive-Bayes-best	0.621	0.619	0.619	0.619	0.111
Naive-Bayes-best-transf	0.497	0.571	0.534	0.443	0.105

We see the Power transform makes our model worse. So, for the Naive-Bayes classifier the best model is the one with the tuned hyperparameter `'var_smoothing': 6.579e-09`, which can be rounded to zero and leads to an accuracy of 62.1%.

3.5 Comparison of results

Finally, we compare the best model of each modeling method and we choose our final model, the one we will make predictions with.

The results table is the following:



	Accuracy	Precision	Recall (mean)	F1-score (mean)	Time(s)
Logistic Regression	0.633	0.635	0.615	0.609	4.083
SVM	0.55	0.275	0.5	0.355	56.198
Random Forest	0.641	0.65	0.619	0.611	12.274
Naive-Bayes	0.621	0.619	0.619	0.619	0.111

Analyzing the table, we decide the best model for our dataset is the **Random Forest** model. This decision is based on the performance of the models compared to the other ones. We can see that it has the highest accuracy value, of 63.5%. It is quite similar to the Logistic Regression model, but Random Forest has a higher F1-score too, as well as a better recall and almost equal precision. The time is a little higher, but we considered anyways that it is the most suitable model for our data.

5. FINAL MODEL AND ESTIMATION OF ITS ERROR

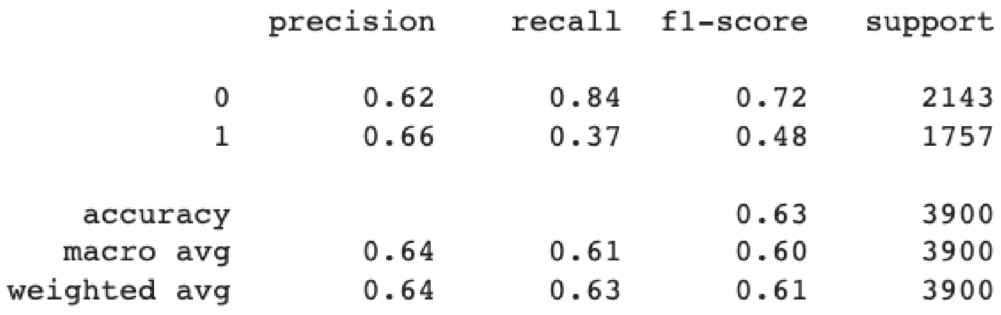
Having chosen our best model, the Random Forest one, now we proceed to making the predictions, therefore evaluating the true performance of the model. The predictions are made using the test set, so it is data that was not used to train the model, that is why here we see how the model truly performs. Before doing this, we will fit again the model with the full training data.

Once we have our prediction of the test set, we compare it with the actual value of the target. To represent how accurate the prediction is, we build a confusion matrix to see the value of the true positives (Delayed-Delayed), true negatives (Not-Delayed-Not-Delayed), false positives (Not-Delayed-Delayed) and false negatives (Delayed-Not-Delayed). Our confusion matrix is:

Predicted Actual	Not Delayed	Delayed
Not Delayed	1808	335
Delayed	1102	655

Therefore, we have 1808+655 correct predictions and 1102+335 incorrect ones. We can also see that it is easier to predict a flight when it is not delayed than otherwise and that our model is quite likely to wrongly predict the flight being delayed.

Lastly, we compute the precision, recall, F-measure and support and obtain:



```
              precision    recall  f1-score   support

           0       0.62      0.84      0.72      2143
           1       0.66      0.37      0.48      1757

    accuracy                           0.63      3900
   macro avg       0.64      0.61      0.60      3900
weighted avg       0.64      0.63      0.61      3900
```

For each metric, we will take into account the weighted average, as we can see it performs better.

We can observe that the model has a 64% accuracy, which is acceptable and leads to a generalization error of 36%. By looking at the precision, it can be seen that, when our model makes a prediction, it is correct 64% of the time (62% for the 0 class and 66% for the 1 class). From the recall we conclude that our model will be able to identify correctly 63% of the observations in a class. Additionally, it must be taken into account that, for the 1 class (delayed flights), the model has difficulty identifying them (it only does so 37% of the time), leading to a very low partial recall, compared to the 0-class (87%). Finally, with the f1-score, we see that the average between precision and recall is acceptable too (61%). Furthermore, it is observed that the 0 f1-score is a lot higher than the 1 one, since the model predicts and identifies more correctly the not delayed flights.

6. CONCLUSIONS

First of all, the main conclusion we extracted from the results of our project is that the prediction of the delay of a flight is not an easy problem to tackle. As we have seen, the best model we were able to find had an accuracy of 64%, this means that 36% of the time it incorrectly classifies the delay of a flight. It is not an extremely high error, but it is considerable, especially when dealing with problems of real life. For our problem, it is not a huge problem to be mistaken with the classification, but it could be for other problems.

Also, we have seen there is a significant difference between how the model classifies a flight being delayed or not delayed. The number of observations in each class is almost the same, but we observed how when a flight is not delayed it classifies it correctly 84% of the time, while when it is delayed only 37%. If we use our common sense, it makes sense that it is easier to predict a flight without a delay, as delays most of the time come from external factors that we do not take into account or can not predict beforehand. This difference between one class and the other is something we should always check and try to see why it happens.

Taking the results of our models into account, we concluded that the best model for us was a Random Forest one, which is an appropriate model for a complex problem like ours. What we think is best about this kind of model is that, not only it had a better accuracy than the others, but also it was easy and fast to train, which are two characteristics that are essential for real life problems. Trying the different modeling methods, we have seen how each of them performs in a dataset like ours. An important conclusion we learned is that Support Vector classifiers do not perform well with large datasets, especially because the execution time is extremely high. Moreover, the Logistic Regression model is also a good choice, being easy and fast to implement. The same thing we can say about the Naive-Bayes classifier, this having the advantage of the tuning of hyperparameters.

On the other hand, we realized the importance of preprocessing our data in order to prepare it for the fitting model. Thanks to this project, we have better understood the necessity of balanced data, dealing with outliers, treatment of incoherent values and coding non-continuous or non-ordered variables, etc. Furthermore, we have visualized the significance of transforming the data in order to have the variables with a gaussian distribution shape. In addition, we have checked the importance of visualization of the data in order to better understand the problem and perform further conclusions, such as how each variable affects the target.

Moreover, we have understood the importance of trying different models in order to have different options and this way choose the one that fits the most. Also, we have checked how trying different hyperparameters on each model leads to it adjusting to our specific dataset and usually better predictions.

To sum up, we learned how to tackle a real life problem and noticed that there are a lot of things to consider and many problems to overcome.

7. POSSIBLE EXTENSIONS AND KNOWN LIMITATIONS

While working on this project, we realized how we encountered some limitations, as well as aspects we could have investigated deeper.

First of all, with our specific problem we have seen how having a large dataset can affect our results and limit the different models we try. It is an aspect we have to take into consideration during the entire process. We had to reduce our dataset to fit the models, so if we had access to more resources or more time, an improvement of the project would be to use more data to train the model.

Another problem we faced is the fact that as our dataset was extracted from a bigger one with more features, our model was not able to predict that accurately our target. As we mentioned before, there are many more aspects to take into account when predicting the delay of a flight (for example, the weather). If we only use the features we have, we are missing a lot of the variables that actually make a flight be delayed. That is the reason why it was so hard for our model to classify the delayed flights, as opposed to the not delayed ones. So, a possible extension would be to add more variables to our dataset.

Other measures we could take to obtain better results on the comparison metrics of our models are, for example, identifying commonly occurring types with low accuracy, finding patterns in the false negatives and false positives in the confusion matrix or adjusting the variables based on their relevance and the information they provide. Furthermore, we could improve the model by doing cross validation with a higher number of observations in the training set.

Also, we could always try more hyperparameters on our models, this way they would be more accurate and fitted to our data. Of course, we know that it requires more time and computational cost.

In conclusion, this project helped us understand the difficulty of a real life problem, as well as all the limitations we have. However, we consider we solved our problem quite efficiently, even though being aware of the things we could improve.

8. REFERENCES

https://towardsdatascience.com/building-a-logistic-regression-in-python-step-by-step-becd4d56c9c8
https://towardsdatascience.com/laplace-smoothing-in-na%C3%AFve-bayes-algorithm-9c237a8bdece#:~:text=Conclusion,the%20positive%20and%20negative%20reviews.
https://www.analyticsvidhya.com/blog/2021/01/a-guide-to-the-naive-bayes-algorithm/
https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/
https://towardsdatascience.com/understanding-random-forest-58381e0602d2
Laboratory documentation from AA1 class