AI in Aviation Engineering

Flight Delay Prediction

DIEUX Jimmy MIELCAREK Flavie MOYAUX Etienne

Introduction

For this project we were asked to develop an artificial intelligence applied to aviation. Using the methods seen during the lectures we implemented an artificial intelligence helping us to predict delays in commercial flights. Specifically interior flights in the United States. The development of our project took place in three main steps which we will endeavour to detail in this report. We first developed a primitive code to understand the functioning and principles of training artificial intelligence, we then tried to improve this code to get results out of it. We then applied the knowledge gained with these to codes to a third one using more detailed data and approach allowing us to get more accurate results. We will finish this report by presenting our results and discussing about the challenges encountered during all this project.

Table of contents

I) Primitive code

The first code is the code where we familiarized with the functioning of an artificial intelligence. We implemented a basic version of our goal to understand how it works. It allowed us to correct its flaws in the main code.

1) Training

This part of the code focuses on training the Artificial Intelligence there are several steps on this process we will only focus on the main ones.

- Loading and preprocessing the data

The first thing to do when programming and training an AI is to provide data and to prepare it to be then processed by our program. In our case the preprocessing of our data included removing prefixes present on the datasheet to make the data readable by our program, converting the data into integers allowing it to be used in the further steps, creating dummy variables meaning that our data was converted in binary data to be then used in the machine learning model, and finally the handling errors and missing values.

- Building of the neuronal network

Once the data was processed and normalised, the dataset as well as the features and targets were split, we started to build the neuronal network. Our first hidden layer contains sixty-four neurons and Rectified Linear Units (ReLU) activation functions. The ReLU function is useful to resolve problems of activation between the neurons. We also use the dropout function which drops randomly twenty percent of our neurons to avoid overfitting. Our second hidden layer contains thirty-two neurons and also contains ReLU function. We then have our output layer with a single neuron.

The lasts step after that is the compiling of the code. In this first program we used the linear regression method using neuronal network which is the main method we used in the project.

2) Prediction

The first code is developed for training and saving the model. It takes raw data, cleans and preprocesses it, trains a machine learning model, and saves the necessary artifacts for future use. While the second one is developed for making predictions. It loads the saved model and artifacts, processes user-provided data, and predicts flight delays. The two codes are complementary: the first prepares the model, and the second utilizes it.

II) The main code

The main code has been improved thanks to the knowledge we gained with the primitive one. It can be divided into three main parts. The processing of data, the training of our model and then the prediction.

1) Data processing

We used a dataset much more complete and more precise than in our first attempt, the preprocessing of this dataset required adjustments for it to work with our model. This data set regroups all interior flights in the United States in the year 2013. In this dataset we had formatting issues with the time which we had to adjust so that the program could read it, we also added supplemental data with the scheduled departures and arrival compared to actual times of departures and arrival allowing us to train our model more accurately. We cleaned the data removing empty or superfluous columns and rows. Finally, we sorted the data to obtain delays statistics by airline.

2) Training program

We implemented three different training codes, trying each time to reduce our mean error.

For the first method we used linear regression. Even though we processed the data in the previous part of the code there is still some preprocessing to do in this part. We created a function that filters the information for each airline company, this parameter allows us to get the relevant information in all the data set. It filters the flights of this company, converts the times of departure and arrival into datetime objects, it calculates the weekday which influences the delays, this function also contains an option to ignore delays greater than sixty-minutes. The preprocessing includes the cleaning of missing data, the encoding of origin airport into integers and the conversion of these integers into binary vectors known as One-Hot encoding. Then the data is sorted between the input and output before the data is split into two parts: the training data set and the testing set. The data is then normalized to ensure that all data is on the same scale, which is important for the regression model we are using. A hyperparameter is searched so it can reduce overfitting, the hyperparameter is searched in different configurations and the model is trained for each model, it is at this stage that the mean squared error is calculated. The best parameters are saved and the model is retrained with those and a prediction is generated on the test set.

Finally, to assess the performance of the training the mean squared error is calculated on the predictions.

The mean squared error that we obtain for this test is 176.24.

We generated two diagrams to better understand what is happening. On the second one we can see all the delays referenced in the data set; they are represented by the blue dots. The green line represents the median of our delays and is the value on which our model will base itself to predict future delays.

FIGURE 1

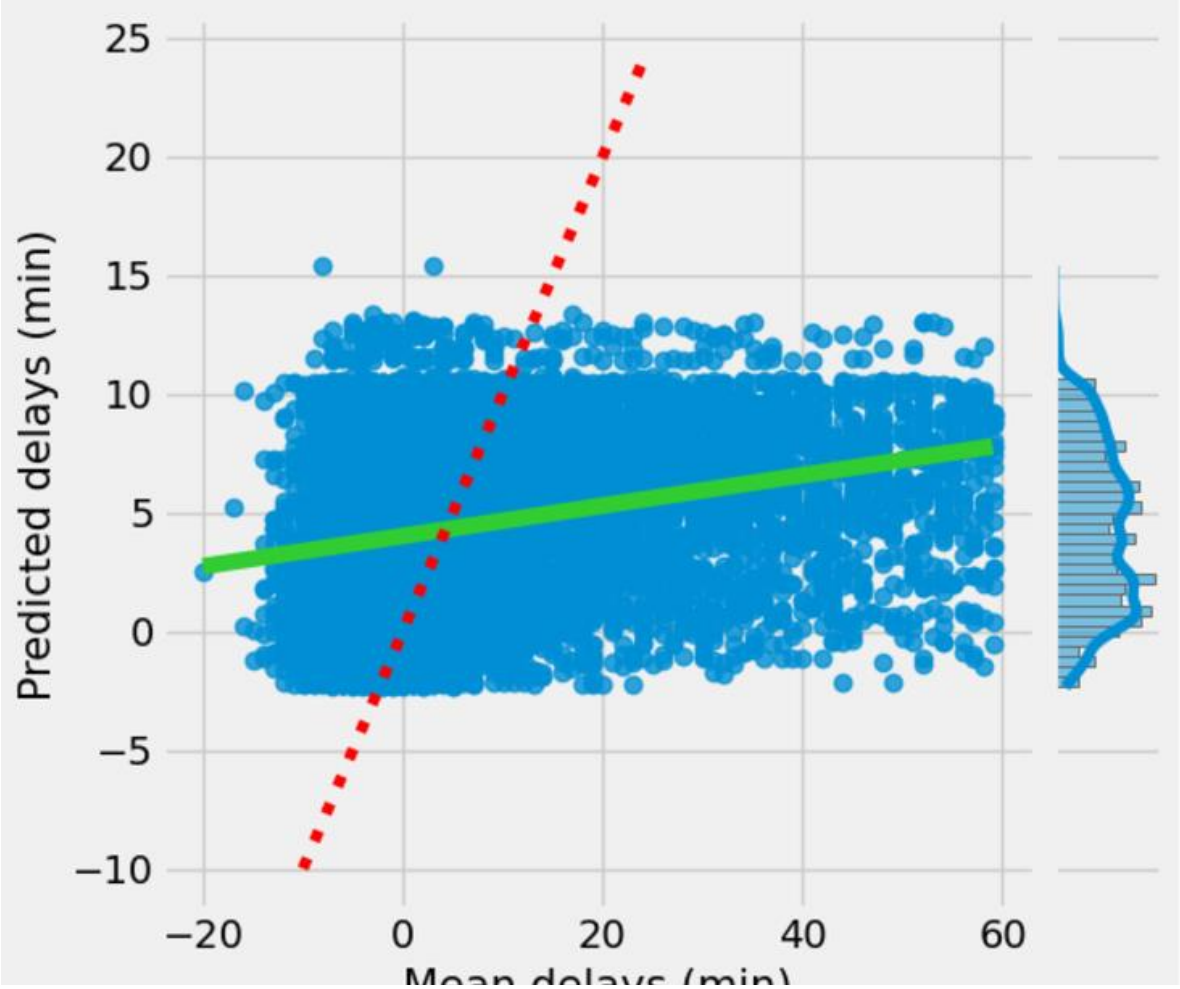

FIGURE 2 : MEDIAN VALUE COMPARED TO DATASET

The second method we used is the neuronal network to train our model. The initial steps are very similar to the previous method, we preprocess the data, we split the data set, we divide between inputs and targets, in short, we set up all the parameters to start the training. The difference resides in the construction of the model itself. In this case we constructed a neuronal network composed of four layers in total. The first hidden layer contains 128 neurons with a ReLU activation. As we did with the neuronal network we used in the primitive code, we deactivated 20% of neurons during the training on the first layer to avoid overfitting. We also applied batch normalization, it is useful to stabilize activations and to speed up the training. After that, we move on to the next hidden layer that is constituted of 64 neurons with ReLU activation, once again we apply dropout and batch normalization. We then have the third hidden layer composed of 32 neurons with ReLU activation. This time no dropout or normalization because the next layer is our last with only one neuron to provide us with a unique result. The next step is to compile our model and to set an early stop parameter. Our model is trained for a maximum of a hundred epochs but if the performance of the model does not improve for ten epochs, then the training is stopped. The final step is the evaluation of our model. We have two metrics here, the mean squared error, and the root mean squared error expressed in minutes, we have a values of the MSE of 1475.73 and a value for the RMSE of 38.42 minutes. We can see that despite an improvement in the error it is still quite high. The diagram presented lower shows us the accuracy of our model which is already quite good, the blue curve is the mean square value.

Despite fairly accurate results, the error value is still quite high. We tried to improve

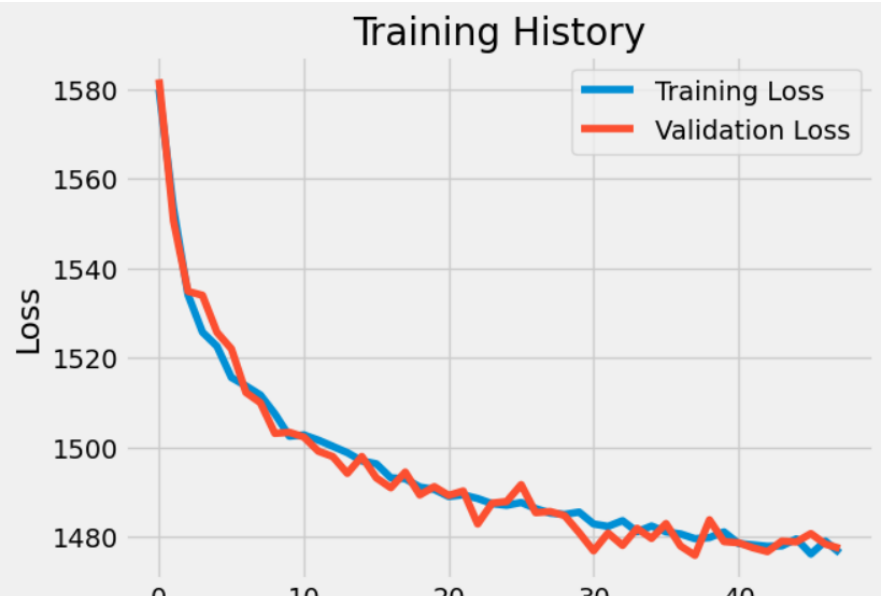

FIGURE 3 : MEAN ERROR COMPARED TO REAL DATA

the neuronal network training without success. We went back to the linear regression

method. We can see this third training method as an upgrade from the first attempt. The main difference resides in added parameters refining the finding of the hyperparameter.

```
# Initialize the XGBoost model
xgb_model = xgb.XGBRegressor(objective='reg:squarederror', random_state=42)

# Parameters to test with RandomizedSearch
param_dist = {
    'n_estimators': [1000, 1500, 2000, 2500],
    'learning_rate': [0.01, 0.05, 0.1, 0.15],
    'max_depth': [5, 7, 9, 11],
    'subsample': [0.7, 0.8, 0.9, 1.0],
    'colsample_bytree': [0.7, 0.8, 0.9],
    'gamma': [0, 0.1, 0.2, 0.3],
    'min_child_weight': [1, 3, 5]
}
```

FIGURE 4 : CODE EXTRACT PRESENTING THE ADDED PARAMETERS

Those parameters are helping us to adjust and tune the hyperparameters. Once they are tested and the best of these parameters are saved, the model is tested giving us the smallest metric of our three training methods with a value of MSE of 27.44 and a value of the RMSE of 5.24 minutes. We also have a diagram comparing the real values with our prediction and we can see a good accuracy even on the highest delays that are the most difficult to predict since they are only occasional and far from the median value. In blue are the real values, in orange is our predicted values and in red is the overlap between the two.

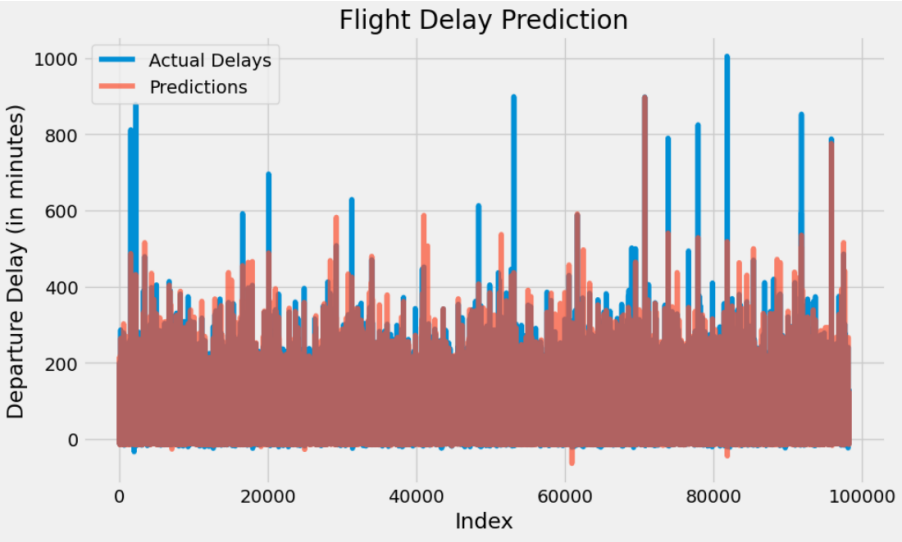

FIGURE 5 : DIFFERENCE BETWEEN PREDICTION AND REAL VALUES

The next diagram shows us the repartition of the error rate showing us its accuracy seeing how many predictions have an error value near zero.

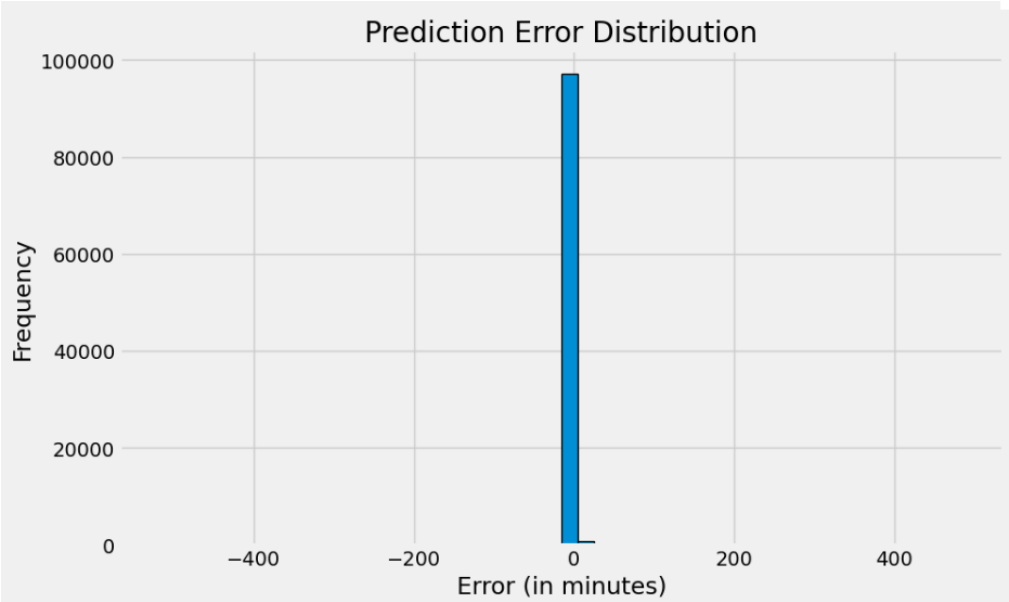

FIGURE 6 : ERROR RATE DISTRIBUTION

Ultimately, we realized that our code, even though it was the most accurate we had, needed too specific data to make our project accessible to anybody. It requires information that is not easily available when you want to predict a delay on a flight. We adjusted to code one last time and obtained the following results:

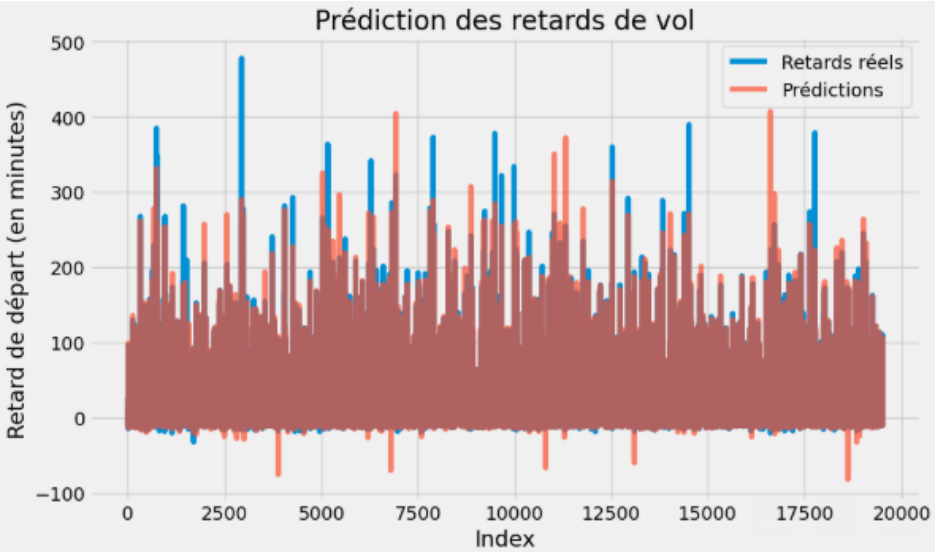

FIGURE 8 : PREDICTED DELAYS COMPARED TO ACTUAL DATA

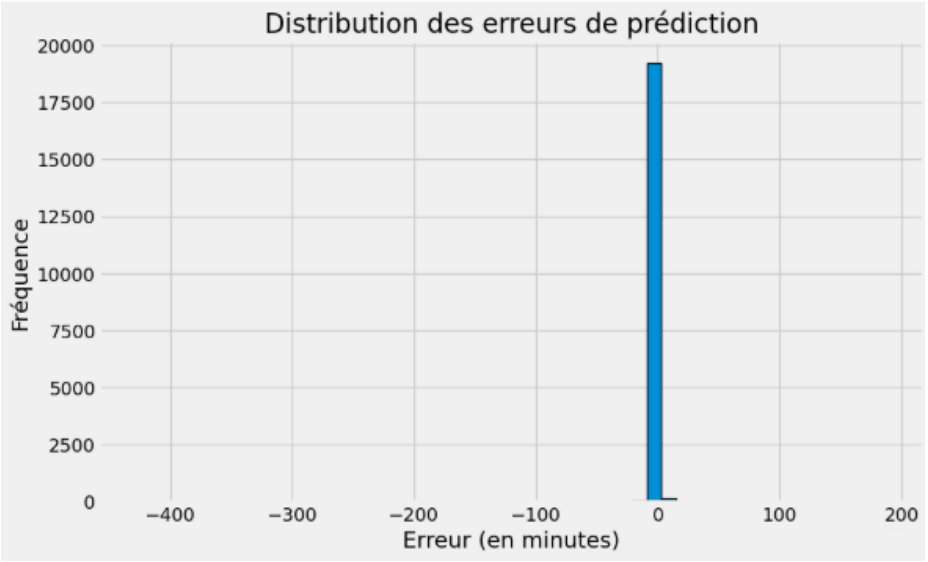

FIGURE 7 : ERROR NORMAL REPARTITION

As we can see in those graphics, we lost accuracy on pur results, if we compare the diagrams from the model right above and our actual model, we can observe a greater error between the actual values and our predicted ones.

3) Prediction phase

After all the training we did, our final goal is to be able to predict a flight delay for interior flights of the United States. To do so we aimed to conceive a user interface where the user can enter flight parameters and is given a delay estimation as precise as possible. The first step in this last program is to load all the resources from the trained model. The program simply takes directly the information in all the files necessary composing the model. The second step is to prepare all data to make it compatible with the model. Then the user interface itself is conceived using widgets. It is then displayed to the user who can input data of their choice in the interface and with the push of a button starts the calculation that gives them a delay value. The user interface is displayed as follows:

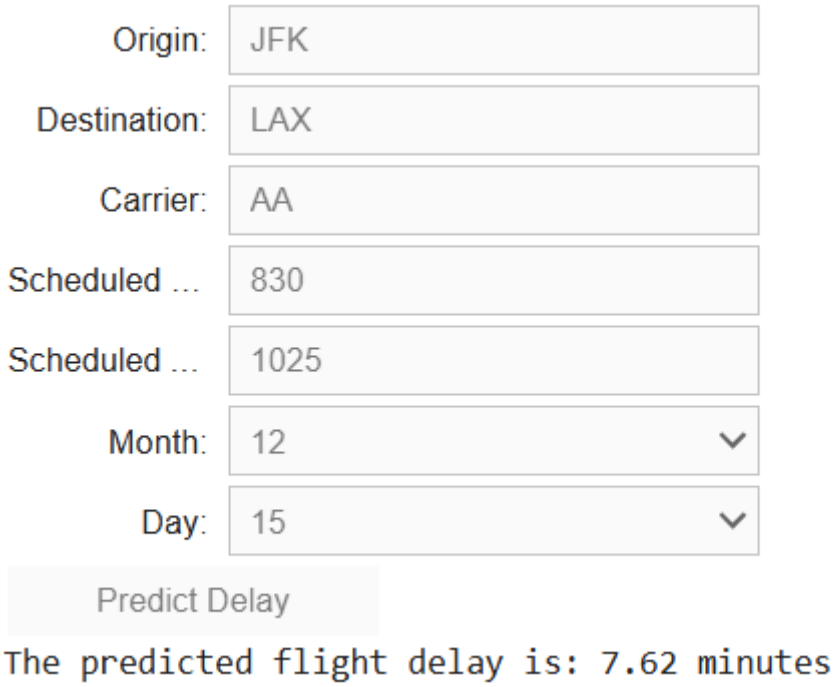

FIGURE 9 : OVERVIEW OF THE INTERFACE

Conclusion:

Our goal for this project was to develop an Artificial Intelligence using an existing data set to train a model that could help us resolve a problem linked to aviation. Our group decided to take flight delays as a topic and tried to develop an AI that could predict flight delays for any flight. We chose an existing data set presenting all the interior flights in United States for the year 2013. After familiarizing ourselves with the functioning of Artificial Intelligence and training models with a rough version of a predictive program, we moved on our main program with a complete data set. After processing it, we tried multiple approach to train a model in most efficient and accurate way possible. Once we obtained a satisfactory result, we constructed the interface allowing any user to predict a flight delay in the range of our data set, meaning any flight inside of the United States.

We encountered some challenges along the way. The first one being the correct processing of the data set and conversions that had to be made so that our document could communicate with our code. The second one was training the model. We not only needed to properly train a model but make sure that its training was accurate that is the reason for the three different training in order to have the lowest mean error possible. The last challenge was to balance accuracy with accessibility. Our aim with this project was to make a system that could be useful in any case despite our limited application, that is why it had to work with data available to anybody that had booked a flight. This meant that we had to lower our accuracy and increase our error since we removed parameters that helped us making our predictions more accurate.

Overall, our goal has been successfully encountered but there is always room for improvement. The main improvement that could be done is to have supplemental data from other years which would allow us to make predictions more trustworthy. Obviously, more data would mean more time required which could also be a problem in the long run.