Francesco Mallard-Marini June 12, 2017

Coding Challenge

1. Explanation of the Machine Learning model

I chose to implement a ML algorithm instead of using a built-in package in Python, even if I could have obtained more accurate results. In particular, I built a classifier to predict the likelihood (%) of a flight to have a delay bigger than 15 minutes. Indeed, if the delay is smaller, I consider the flight to be on time.

Here are the variables that I chose to select:

- Year
- Quarter
- Month
- Day of Month
- Day of Week
- Origin City Name
- Origin State Name
- Destination City Name
- Destination State Name
- Diverted (1=Yes, 0=No)
- Delay bigger than 15min (1=Yes, 0=No)

As I am trying to predict the likelihood of a flight to be late, I chose to use a generative model where the outcome represents the probability that someone books the flight: P(Y=1| X=x). The logistic function of a logistic regression allows to calculate it. Moreover, the different features are sometimes correlated (ex: for a particular departure airport, we cannot have the same arrival airport) and logistic regression works in these situations (whereas a Naïve Bayes algorithm maybe would have not).

Here is my approach for this model:

1) The hypothesis function is: $h_\theta(x) = g(\theta^T x)$

   where:
   $g(z) = \frac{1}{1+e^{-z}}$ (logistic function)
   $\theta^T x = \theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n$

   $x = (x_1, \dots, x_n)$ are the features of the dataset
   $(x_0 = 1)$ for every row
   $\theta = (\theta_0, \dots, \theta_n)$ are the coefficients for each feature

The hypothesis function gives the decision boundary for this dataset: for every data point, when the value of the hypothesis function is positive, the algorithm predicts 1 (=delay), otherwise it predicts 0 (=no delay).
The logistic function is always comprised between 0 and 1.

2) The cost function is: $J(\theta) = -\frac{1}{m}\ [\ \sum_{i=1}^{m} y_i\ log\big(h_\theta(x_i)\big) + (1 - y_i)\log\ (1 - h_\theta(x_i))]$

   where:
   m = size of the training set
   y_i = real value of the label for the point i

The cost function is positive: its value increases as the number of misclassified points increases (the cost function is null when all the points are correctly classified).
We want to find the parameters θ that minimize the function

3) The algorithm used to minimize the function is the gradient descent. In a hypothetical graph representing the cost function as a function of the different parameters, we start with random parameters θ and we take the direction of the steepest descent. We reiterate the process until we reach a global minimum (which exist for sure if we use this cost function, because the graph is always convex).
4) Once the optimal θ has been found, I used the model to estimate the labels and I look at the proportion of well-classified points compared to the total number. Here, I obtained 77.78%, which is satisfactory.

Once the model has been tuned, I estimated the likelihood of delay for 10,000 new flights.

2. Manipulations on the data

In order to apply my model, I needed to get rid of the strings. I have several alternatives and generally creating dummy variables is a good one. But here, as I have too many different strings, that would have been equivalent to create hundreds of new columns, and thus, make the dataset too complex. So, I simply decided to transform my strings into integers, even if this can create some undesired ordered set.

Then I separated the dataset into training and test one: the training set contains all rows where the values corresponding to the delay are either 0 or 1; the test set contains all rows where there is no value (NaN).