IML Hackathon

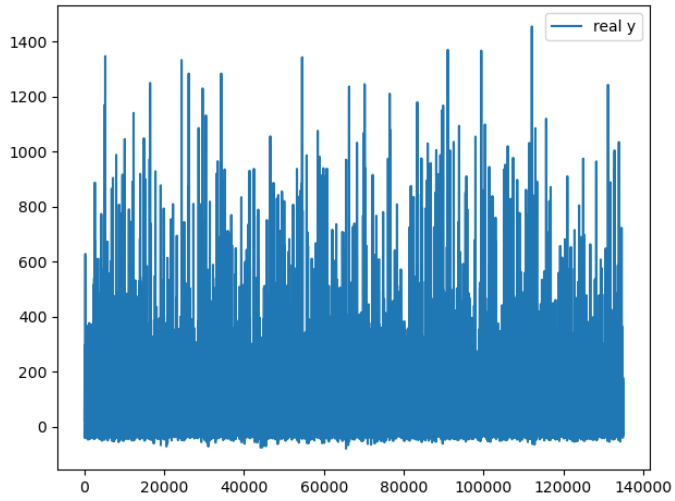
The problem: we were tasked with predicting the delay of a given flight and the reason for the delay. Looking at the samples, their features, and what we were required to predict, we came to the conclusions that:
a) In order to predict the delay (in minutes) we'd need to apply Linear Regression.
b) In order to predict the reason for the dalay we'd need to use a Multiclassifier (non binary).

In our approach to 'a' we tried to use the model sklearn.LinearRegression which we saw in class, but we quickly realized it wouldn't be a valid option due to its bad results. We then went online in search of a better existing model and found the library XGBoost, containing the model XGBRegressor. This model yielded much better results than sklearn.LinearRegression.
The first algorithm we used for 'b' was adaboost on DecisionTree with max_depth = 1, which we'd also seen in class, however as in 'a' the results were unsatisfactory. We then used RandomForest (bagging of decision trees and decorrelation), but still the results weren't as good as we'd hoped.
The third algorithm we tried was sklearn.LogisticRegression in the multiclass case, where we also toyed around with the amount of iterations. All the aforementioned algorithms were examined based on the T which represents the amount of learners used in the boosting, where the error rate was plotted for us to examine. In addition to changing the value of T we also toyed around with the max_depth parameter. Eventually we noticed that XGBoost contains the model XGBClassifer. We also found another boosting algorithm called OneVsRestClassifier, which we then used to boost XGBClassifier. This implementation yielded far better results than the first two algorithms. Same as before we evaluated our progress based on the plots which showed us the error rates as a function of max_depth which is defined as described above.

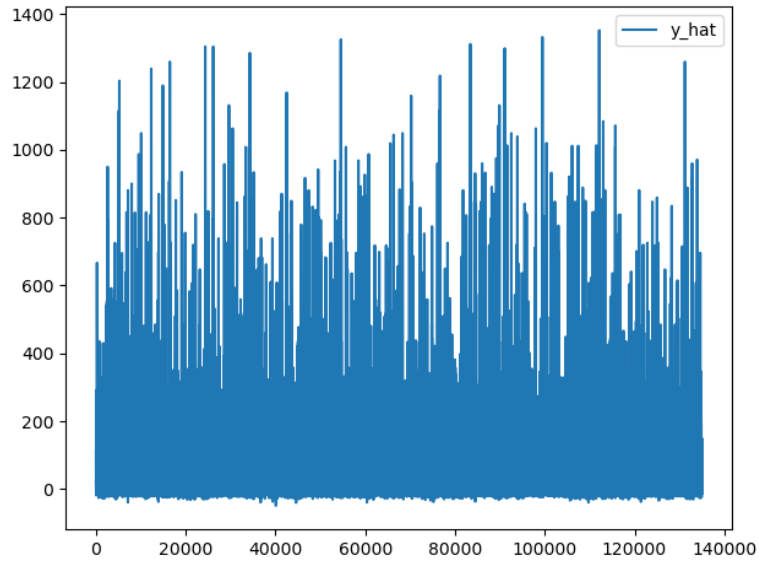
The given parameters for all the learners are the result of trial and error, and all the preliminary tests were run on the validation samples.

Preprocess: given a training sample we made the following adjustments:
a) We removed all the data we deemed to be irrelevant: OriginCityName, OriginState, DestCityName, DestState, FlightNumberReportingAriline, TailNumber, FlightDate, Distance, CRSElapsedTime.
b) CRSDepartTime was changed to a 'by minute' representation, as was CRSArrTime.
c) We added a column 'DistanceElapsed' which is the product of Distance and ElapsedTime. This was done due to their high correlation.
d) We added the columns RealDeparture and RealArrival, which were calculated by adding together CRSDepartTime and ArrDelay, and CRSArrTime and ArrDelay respectively.

When preprocessing the data we received in predict(), we want to calculate the RealDeparture and RealArrival, which we do by saving the median of ArrDelay in the training data. By doing so we get an estimate of the prediction in the typical case.



Actual delay of test samples



Predicted delay of test samples

Once we reached a stage where the numbers backing up the graphs above fit our needs, i.e. the error rate was low enough and the graphs looked very similar, we stopped learning and saved the learner.