**PROJECT REPORT**

**Flight Delay Prediction**

**By E. Krishnakanth**

**1) Problem Statement**

The aim of the project was to build a two stage ML model that can predict whether a flight is delayed or not during arrival. If the flight is delayed at arrival, the amount of delay time is calculated. We consider flights travelling in 15 airports (*refer table 1*), for the years 2016 and 2017.

**2) Introduction**

The model built is useful in real life application of prediction of flights which are delayed. This will allow the passenger to know beforehand whether his flight is delayed thereby minimizing the inconveniences that may be caused due to delay. (missed appointments, interviews, presentations etc.) The model is predicted only with the help of weather data at the specified airport while in real life many other factors come into play in determining whether delay occurs like technical problems, air traffic restrictions, security issues etc.

**3) Dataset description**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ATL | CLT | DEN | DFW | EWR |
| IAH | JFK | LAS | LAX | MCO |
| MIA | ORD | PHX | SEA | SFO |

**Table 1**

|  |  |  |  |
| --- | --- | --- | --- |
| WindSpeedKmph | WindDirDegree | WeatherCode | precipMM |
| Visibilty | Pressure | Cloudcover | DewPointF |
| WindGustKmph | tempF | WindChillF | Humidity |
| date | time | airport |  |

**Table 2**

|  |  |  |  |
| --- | --- | --- | --- |
| FlightDate | Quarter | Year | Month |
| DayofMonth | DepTime | DepDel15 | CRSDepTime |
| DepDelayMinutes | OriginAirportID | DestAirportID | ArrTime |
| CRSArrTime | ArrDel15 | ArrDelayMinutes |  |

**Table 3**

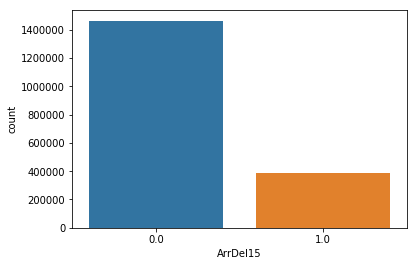
The flight data is merged into a single data frame for the selected 15 airports (*refer table 1*). The important features belonging to weather data (*refer table 2*) and flight data (*refer* *table 3*) is selected.

Weather data corresponding to the Departure airport and Arrival airport is merged (inner merge) with the parameters namely

* Date of the flight departure/arrival
* Time(hourly) of the flight departure/arrival
* Airport (name) during flight departure/arrival

In the data provided the almost 80% of the flights are not delayed and rest are delayed. So, the problem of **class imbalance** arises- Most machine learning algorithms works best when the number of instances of each class are roughly equal. So, machine learning algorithms tend to produce unsatisfactory classifiers when faced with imbalanced datasets.

The bar graph below indicates the class imbalance present in the given data with the minority class being ‘delay’ and majority class ‘not delay’.

****

**Figure 1.1**

**4) Classification**

Classification is done with the help of features (*refer table 1*) chosen which make up the feature list in order to predict the value of the output class which indicates whether flight is delayed or not. A flight is considered as delayed if delay time is greater than 15 minutes.

This model works as soon as the flight takes off as it takes into account the departure delay time as a factor for predicting whether delay occurs.

The dataset was split as 80% training data and 20% test data.

The accurate representation of prediction of whether the flight is delayed or not is given by the confusion matrix:

Actual

Delay

Not Delay



Predicted

Not Delay

Delay

1. **TN / True Negative:**actualcase was negative and predicted case negative (i.e. flight is actually delayed and predicted as delayed)
2. **TP / True Positive:**actualcase was positive and predicted case positive (i.e. flight is actually not delayed and predicted as not delayed)
3. **FN / False Negative:** actualcase was positive but predicted case negative (i.e. flight is actually not delayed and predicted as delayed)
4. **FP / False Positive:**actual case was negative but predicted case positive (i.e. flight is actually delayed and predicted as not delayed)

**4.1) Classification metrics**

We evaluated the following metrics for classification:

Precision = TP/ (TP + FP)

Recall = TP/ (TP+FN)

F1 Score = 2\*(Recall \* Precision) / (Recall + Precision)

**4.2) Trade-off**

The passenger in an ongoing flight will have more inconvenience caused when flight is actually delayed but is predicted as not delayed (this is indicated by FP in the confusion matrix). When a flight predicted delayed but it is actually not delayed the passenger is not disturbed as much as the previously mentioned case. Trade off of precision and recall results in consideration of recall for class 1 (emphasis on the minority class ‘delay’ as this class is used in regression phase). Recall and FP have an inversely proportional relationship and the model is chosen based on the highest value of recall for class ‘delay’.

**4.3) Classification approaches**

The classifiers used were: Decision trees classifier, Extra trees classifier, XGBoost classifier, Gaussian classifier

No sampling

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **XGBoost** |  | **Precision** | **Recall** | **f1-score** |
| **class-not delay** | **0.80** | **0.99** | **0.89** |
| **class-delay** | **0.68** | **0.05** | **0.09** |
|  | | | | |
| **Extra trees** | **class-not delay** | **0.83** | **0.92** | **0.87** |
| **class-delay** | **0.48** | **0.30** | **0.37** |
|  | | | | |
| **Decision Tree** | **class-not delay** | **0.84** | **0.82** | **0.83** |
| **class-delay** | **0.36** | **0.39** | **0.38** |
|  | | | | |
| **Gaussian NB** | **class-not delay** | **0.82** | **0.88** | **0.85** |
| **class-delay** | **0.36** | **0.26** | **0.30** |

**Table 4**

Class ‘not delay’ produced much better results compared to class ‘delay’ due the class-imbalance problem. (*refer Figure 1.1 in Dataset Description*)

**Possible solutions to class imbalance**

1. Under sampling: removing some of the majority class so it has less effect on the machine learning algorithm

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **XGBoost** |  | **Precision** | **Recall** | **f1-score** |
| **class-not delay** | **0.87** | **0.65** | **0.74** |
| **class-delay** | **0.32** | **0.63** | **0.42** |
|  | | | | |
| **Extra trees** | **class-not delay** | **0.87** | **0.71** | **0.78** |
| **class-delay** | **0.35** | **0.58** | **0.43** |
|  | | | | |
| **Decision Tree** | **class-not delay** | **0.85** | **0.59** | **0.70** |
| **class-delay** | **0.28** | **0.61** | **0.39** |
|  | | | | |
| **Gaussian NB** | **class-not delay** | **0.83** | **0.79** | **0.81** |
| **class-delay** | **0.32** | **0.39** | **0.35** |

**Table 5**

Under-sampling doesn’t produce satisfactory results relative to other methods. (generally, because it discards potentially useful data)

2)Oversampling- adding more of the minority class so it has more effect on the machine learning algorithm.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **XGBoost** |  | **Precision** | **Recall** | **f1-score** |
| **class-not delay** | **0.87** | **0.65** | **0.74** |
| **class-delay** | **0.32** | **0.63** | **0.42** |
|  | | | | |
| **Extra trees** | **class-not delay** | **0.83** | **0.92** | **0.87** |
| **class-delay** | **0.48** | **0.29** | **0.36** |
|  | | | | |
| **Decision Tree** | **class-not delay** | **0.83** | **0.83** | **0.83** |
| **class-delay** | **0.36** | **0.37** | **0.37** |
|  | | | | |
| **Gaussian NB** | **class-not delay** | **0.83** | **0.78** | **0.81** |
| **class-delay** | **0.32** | **0.39** | **0.35** |

**Table 6**

SMOTE synthesises new minority instances between existing minority instances. First it finds the n-nearest neighbours in the minority class for each of the samples in the class. Then it draws a line between the neighbours and generates random points on the lines.

SMOTE is preferred over random oversampling as random oversampling makes exact copies of existing samples which does not produce optimum results.

1. SMOTE oversampling

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **XGBoost** |  | **Precision** | **Recall** | **f1-score** |
| **class-0** | **0.94** | **0.92** | **0.93** |
| **class-1** | **0.71** | **0.79** | **0.75** |
|  | | | | |
| **Extra trees** | **class-0** | **0.91** | **0.95** | **0.93** |
| **class-1** | **0.79** | **0.63** | **0.70** |
|  | | | | |
| **Decision Tree** | **class-0** | **0.92** | **0.91** | **0.92** |
| **class-1** | **0.68** | **0.71** | **0.69** |
|  | | | | |
| **Gaussian NB** | **class-0** | **0.94** | **0.90** | **0.92** |
| **class-1** | **0.66** | **0.77** | **0.71** |

**Table 7**

An appropriate classifier is found based on the classification reports (report shows the main classification metrics precision, recall and f1-score for each individual class) of different classifiers used. (*refer* *Results*)

XGBoost performs better than other classifiers to produce a good value of recall. This is due to the fact that XGBoost uses Gradient boosting algorithm where new models are created that predict the residuals or errors of prior models and then added together to make the final prediction.

**5) Regression**

Regressor is used so as to predict the amount of delay (ArrivalDelayMinutes) only for the flights that were delayed as predicted by the classifier.

Classification is done for all the flights to predict ‘delay’ or ‘not delay’ and these values are used in the regressor.

**5.1) Regression metrics**

The metrics used are MAE (mean absolute error) and RMSE (Root **mean squared error**).

MAE describes the typical magnitude of the residuals(errors). It is the average of all absolute errors. A smaller MAE value implies greater accuracy of prediction with less errors.

MAE= (1/n) Image result for sigma symbol |y-y’|

**Root mean squared error** (RMSE) – it is the square root of the Mean Square Error

RMSE= [ (1/n) Image result for sigma symbol (y’ - y)2 ](1/2)

n-total data points

y-actual output value

y’-predicted output value

Outliers (extreme values) in data will contribute to much higher total error in the RMSE than they would the MAE so RMSE is more sensitive compared to MAE. The appropriate regressor was chosen based on the above-mentioned metrics.

**5.2) Regression approaches**

The regressors used were: Linear, Extra Trees and XGBoost Regressor. **Case 1**-Without including ‘DepDelayMinutes’ as a feature

|  |  |  |
| --- | --- | --- |
| **XGBoost Regressor** | **RMSE** | **70.004** |
| **MAE** | **42.754** |
|  | | |
| **Extra Trees Regressor** | **RMSE** | **76.091** |
| **MAE** | **46.279** |
|  | | |
| **Linear Regressor** | **RMSE** | **70.773** |
| **MAE** | **43.441** |

**Table 8**

**Case 2**-Including ‘DepDelayMinutes’ as a feature

|  |  |  |
| --- | --- | --- |
| **XGBoost Regressor** | **RMSE** | **16.507** |
| **MAE** | **11.615** |
|  | | |
| **Extra Trees Regressor** | **RMSE** | **17.243** |
| **MAE** | **12.3385** |
|  | | |
| **Linear Regressor** | **RMSE** | **16.816** |
| **MAE** | **11.807** |

**Table 9**

**Result:**

Classification is done to determine whether the flight was delayed or not. SMOTE oversampling is chosen for classifiers as it produced better results than random under-sampling/over-sampling. The top 2 classifiers that produced best results were the Extra trees classifier and XGBoost classifier. The XGBoost classifier was chosen as the best one as it produced better results of recall (0.79 – *refer Table 7*) for the minority class ‘delay’.

The amount of delay is calculated for values which were predicted as delayed by the classifier. This amount is calculated with the help of a regressor. Case 2 is chosen over Case 1 due to better results produced (*refer Table 9*). XGBoost regressor is chosen as the best one as it has lesser RMSE (16.5) and MAE (11.6 *refer Table 9)* compared to other regressors. XGBoost is inherently better than other regressions like linear which work only if linear relationships are present (doesn’t account for curved relationships) and algorithm used by XGBoost is gradient boosting which results in a high execution speed and can predict the errors of the prior models as well.

The two-stage machine learning model is constructed with help of the XGBoost classifier and XGBoost regressor.