International Journal of Computing and Artificial Intelligence

E-ISSN: 2707-658X
P-ISSN: 2707-6571
www.computersciencejournals.com/ijcai
IJCAI 2025; 6(1): 141-148
Received: 11-02-2025
Accepted: 19-03-2025

Manan Buddhadev
Rochester Institute of Technology, Rochester, NY, USA

Corresponding Author:
Manan Buddhadev
Rochester Institute of Technology, Rochester, NY, USA

Data preparation for predictive analytics: Cleaning and balancing airline datasets for flight diversion prediction

Manan Buddhadev

DOI: https://www.doi.org/10.33545/27076571.2025.v6.i1b.144

Abstract
The paper looks into the various aspects of data mining, primarily the cleaning and preparation phase. A majority of the work required to be done while doing any analysis on a given dataset is its cleaning and preparation. Various approaches of cleaning the data include handling null values, missing values, outliers, anomalies and balancing the dataset. However, these are a few approaches amongst numerous others which need consulting from the domain expert.
This paper looks at the airline dataset which consists of one main dataset and three supplemental datasets that link the names of carriers, the airplane details, and the airport details. But, the dataset is not clean at all. To achieve the goal of predicting whether a flight will be diverted or not, this data needs to be cleaned and balanced. The paper will dive deeper into the process of the preparation of data and the final results of determining whether a flight will be diverted or not.

Keywords: Data cleaning, machine learning, predictive analysis, data visualization

Introduction
The number of flights getting diverted in 2016 was 13,652 [2]. There are a lot of reasons for these diversions which include factors like adverse landing conditions, medical emergencies, and unruly passengers [1]. It is not easy to predict if a particular flight will be diverted during the taxi in, taxi out of the flying phase. Although the number of flights getting diverted is not a lot, it is certainly significant.
This paper aims at predicting the diversion of a flight due to some of the many reasons. The reasons considered for the analysis are carrier delay, weather delay, NSA delay, security delay, late aircraft delay. This paper is focused mainly on the data cleaning and preparation phase of data analysis. Four datasets will be used for analysis. The main dataset is airline data [3] and the three supplemental datasets are: airport data, carrier data and plane data [5]. The airline data consists of 29 attributes with approximately a million records [3]. The airport data helps in the mapping of the IATA code mentioned in the airline data. It provides information about the names and the locations of the corresponding airports. The carrier data maps the carrier code to the name of the airlines. The plane data maps the tail number with the details of the airplane.
The data to be experimented with will be prepared by combining all the four datasets to form a single dataset. The final dataset will be cleaned to remove the problems arising because of the heterogeneity of the resulting data. The cleaning process will aim at detecting the outliers in the dataset and fixing or removing them according to the situation using exploratory analysis. The techniques utilized for the experiments are Decision Trees(J48) and Naive Bayes classifier for predicting whether a flight would be diverted or not.
The organization of this paper will be as follows, the design section will describe the design of the entire project, the techniques adopted to solve the problems and why these approaches were chosen. The implementation section will walk through the step-by-step process of data analysis performed in the project including data cleaning, preparation, classification, and visualization. Results will review the results obtained from the experimentation. Lessons learned will discuss all the learning we gained during the development of the project. Section Conclusions will contain the conclusions derived from the experiments performed along with the best result and a possible explanation of that.

This section will also discuss the lessons learned during the project.
Section Current status will talk about the current work done in the application, and the data cleaning techniques used. Future work will discuss the scope of improvement in the project and the ideas that will be implemented in the later versions of the application. The additional possible experiments that can be done to make the results even better will also be discussed in the same section.

Design

The project is divided into three phases. The first phase deals with assembling the data sets, combining them and preparing them for further operations. This phase also includes implementation of two techniques to handle missing or empty values in the dataset. In the second phase, the dataset will be reduced to 100,000 records, and the two techniques to handle the missing values in the dataset will be applied again. The data will then be balanced using a balancing technique called 'Synthetic Minority Over-sampling Technique (SMOTE)' [6]. The diversion of the flight will be predicted using the Decision Tree algorithm. The last phase of the project includes normalizing the data set using WEKA's default normalizer. Two classifiers, Decision Tree, and Naive Bayes will be applied to the normalized data. The results of the three prediction techniques will be compared and discussed in the conclusions.

Implementation

Data pre-processing and mining

There are four different datasets used for the projects as mentioned in the introduction. In the first phase, all the four datasets were collected from the source. The datasets were then combined based on the information needed to make sense from the datasets. Since the dataset contained only tail numbers, an attribute called 'TailNum', it was hard to learn what it corresponded to. After looking at the other dataset, it was clear that the 'TailNum' attribute was a foreign key to the 'planedata.csv'. This CSV file consisted of the tail numbers mapped to the type of the aircraft, the manufacturer, the license issue date, the model number, the status of the licenses whether it is valid or invalid, the type of aircraft, the engine type and the plane year. Since, some of the attributes in plane data could lead to a flight diversion, given that the engine type or the issue date of the license could be a factor, it became necessary to merge the datasets.
It was crucial to find the airlines which would be the best based on the delays in the dataset and the prediction model used. The two datasets '2008.csv' and the 'carriers.csv' were joined to get the name of the airlines, thus giving almost each record a name to which it can be linked. Moreover, the airport location might play a role given harsh weather conditions lead to a flight diversion. Hence, the datasets '2008.csv' and 'airports.csv' were joined, giving an idea of each flight's route.
It was soon realized that the size of the dataset '2008.csv' was vast and it might not be possible to process it with the given resources. Therefore, the dataset was reduced from 7 million to 100,000 records. Data reduction was made randomly by choosing 100,000 records from the original dataset '2008.csv' using a script in Python. In the first attempt, the algorithm produced a dataset with the data containing only class '0' in the 'Diverted' column. The dataset was rendered useless for further processing because the data included only one class. In the second attempt, the original data was divided into two smaller datasets, one consisting of only 'Not Diverted' class and the one consisting of only 'Diverted' class. The aim was to extract 50,000 records from each one of these datasets and combine them to form the final dataset. But the number of '1' class records were only 2649 whereas the number of '0' class records were 50,000 which called for data cleaning including removal of outliers and re-sampling. But this had to be done in the dataset obtained after merging all the four datasets.

The query used to merge the datasets is:

```
SELECT *
FROM (
SELECT *
FROM Years
JOIN planeData
WHERE Years.TailNum = planeData.tailnum
) AS temp
JOIN carriers
WHERE temp.UniqueCarrier = carriers.Code;
```

Once the datasets were merged, there were a lot of missing values across all the attributes. The missing values were handled using the following techniques:

Removing the records

This technique is not very often used because it can be dangerous. It means removing all the missing values from the dataset. But this might lead to loss of important information. Therefore, it is better to use such an approach only in the worst case scenario. This technique is used in this project for the sake of comparison.
The columns Type, Issue_date, Year, DepTime and CarrierDelay had many missing values. The column 'Type' represents the type of the airplane and it cannot be empty because an aircraft needs to have a type. The column 'Issue_date' represents the date when the aircraft was issued. The value of this column cannot be empty for any aircraft because the aircraft can only fly once it is issued. The values of the column 'Year' can range between 1987-2008 as per the data dictionary. Therefore, it cannot be 0 or None. The values of the column 'DepTime' cannot be NA because a departing flight will always have a departure time. The values of the column 'CarrierDelay' cannot be NA because according to the data dictionary, this field is supposed to have time in minutes. These values were handled by the two methods mentioned in the previous phase, and the resulting numbers are shown in table 1 Other than these columns, some of the columns were removed entirely from the dataset because more than 80% of the values were missing in these columns. These columns include 'Year', 'DayOfWeek', 'Cancellation', 'CancellationCode', 'CarrierDelay', 'Weather', 'NASDelay', 'SECDelay' and 'Status'.

Table 1: Removing missing data

Column Name	Number of Records
Type	1567
Issue_date	1
Year	1565
DepTime	629
CarrierDelay	31982
Total Records Removed	35744

The total number of records in the dataset before removing the missing values were 49,777. It can be seen from the table that the number of records removed were 35,744 thereby leaving the total records to be 14,032 in which records with class '1' are 2,364 and the ones with class '0' are 11,667.

The data obtained after the removal of missing values needed balancing as the number of records with class '1' are relatively less than the ones with class '0'. The balancing of the data was done using SMOTE. In order to equalize the number of 0 and 1 class records, the records with class '1' were increased 100%, two times and then 23% one time leading to a total of 11631 records with class '1' which is almost equal to the ones with class '0'. After applying SMOTE, the dataset was randomized to avoid overfitting by order of the new records.

After balancing the dataset, Decision Tree classification algorithm was applied to the raw dataset and the dataset obtained after all the data pre-processing.

Substituting the missing values

This method includes replacing the missing values with the mean, median or mode of the attribute. This technique can be applied to the numeric type data.

In this stage, the missing values in the dataset were substituted with some values rather than just removing them because it is important not to miss records of critical importance. In order to do so, the "ReplaceMissingValues" filter of WEKA was used. Even though most of the missing values were filled, the problem of unbalanced data persisted. The data was balanced using SMOTE increasing it four times by 100% and one time by 18%. The final dataset had 47124 records with class '0' and 47280 records with class '1'. Once the dataset was balanced, a new problem arose when trying to run algorithms on WEKA. The problem was that many of our attributes were in 'string' format and the input of WEKA requires a non-string format. In order to fix this issue, the "RemoveType" filter was used, and all the string types in the data were removed. After that the decision tree (J48) classification algorithm was applied to it, the results of which are shown in the section results.

In the last phase, the dataset obtained after balancing was normalized using WEKA's default normalization technique. This method scales the dataset in the range $[0, 1]$. Once the dataset was normalized, two different classifiers, Naive Bayes and Decision Tree, were applied to it.

All the processing mentioned above, from merging the dataset to filling the missing values, was done in Python, version 2.7.x. The database used was SQLite which is available as an inbuilt library of Python. All the CSV files were loaded into the database in the form of tables with one table per file. Different scripts were written to store all the data in a single database file having multiple tables.

Furthermore, any relevant results obtained from the database like joining the table, selecting a particular airline etcetera, were stored in a CSV file. There is a need to store the intermediate result for faster processing as the total data size is very high

Data Visualization

Data visualization was done using an online tool called 'plot.ly'[4]. The figure 9 shows the percentage distribution of the raw airline data that was obtained after merging the main data and the supplemental data. The figure 8 shows the percentage distribution of airline data after the dataset was cleaned using the techniques described in the rest of the paper. The figure 11 shows the class distribution of column 'Diverted' in the raw data. The figure 10 shows the class distribution of column 'Diverted' after the data was balanced using SMOTE.

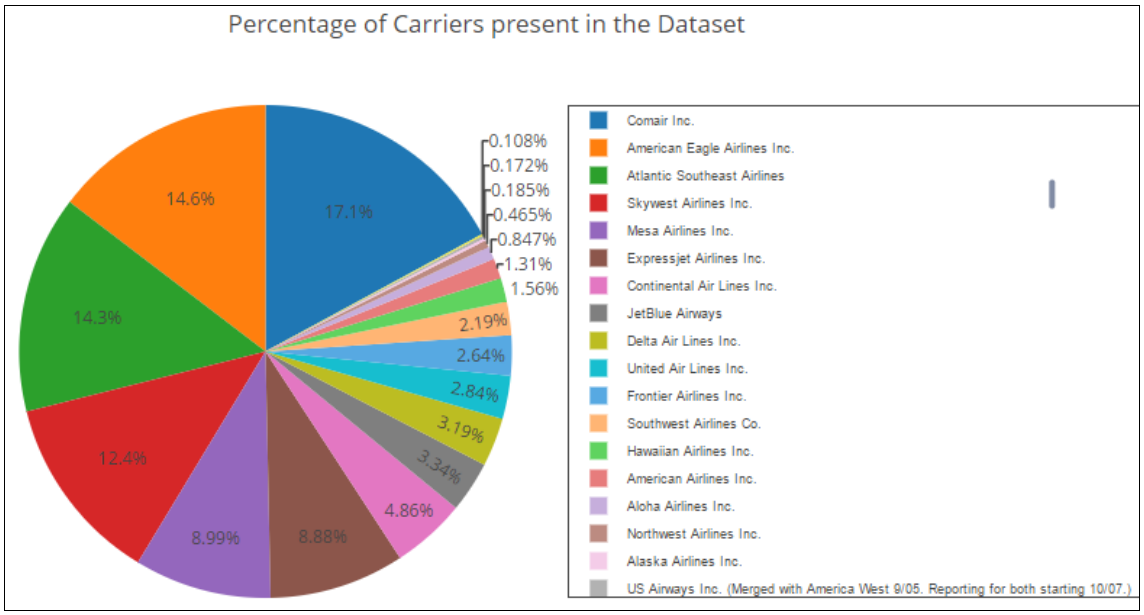

Fig 1: Data visualization (in %) of raw carrier data

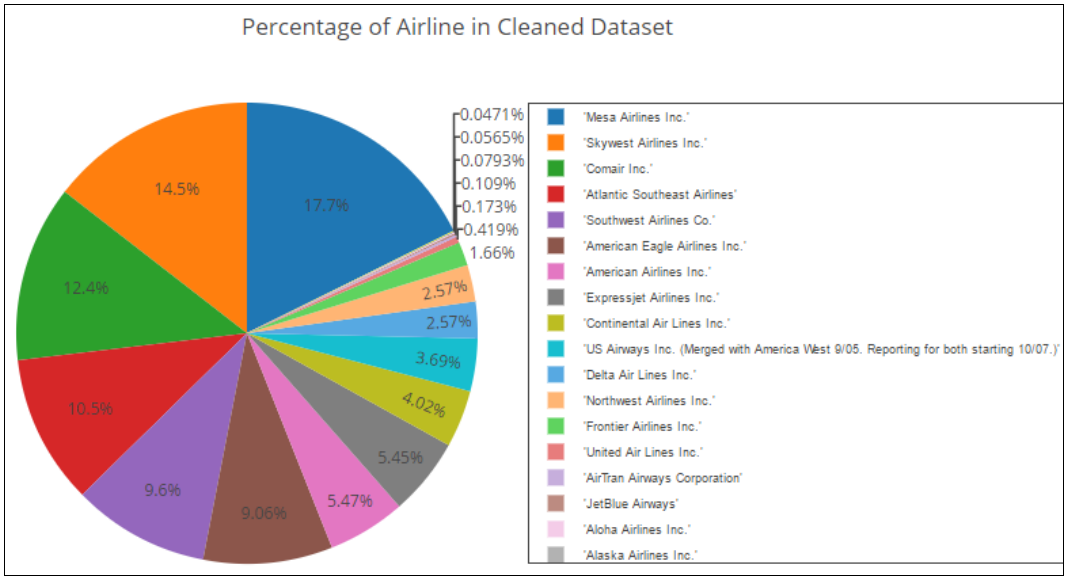

Fig 2: Data visualization (in %) of clean carrier data

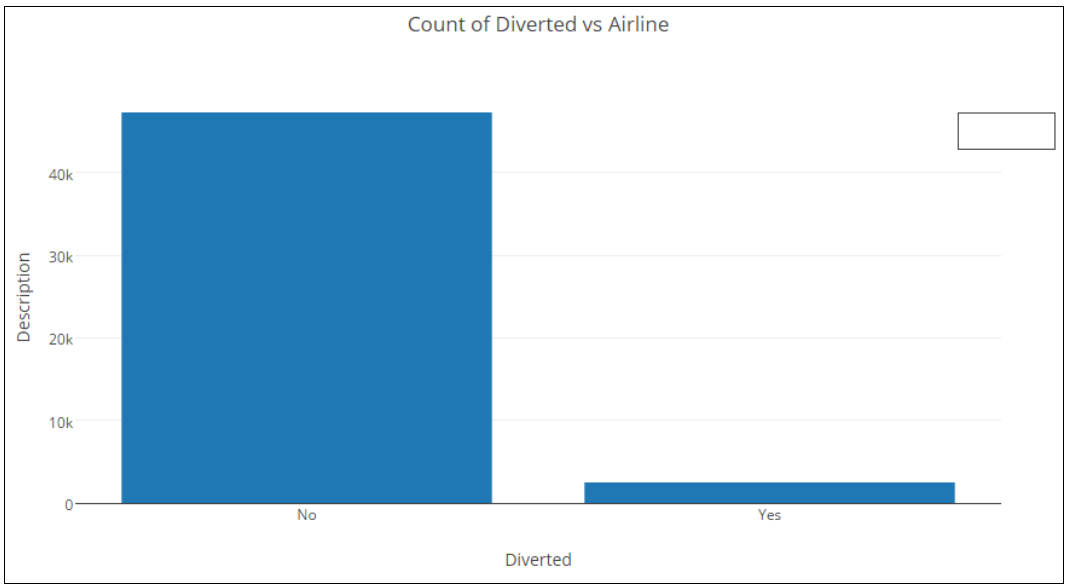

Fig 3: Data visualization (in %) of raw diverted data

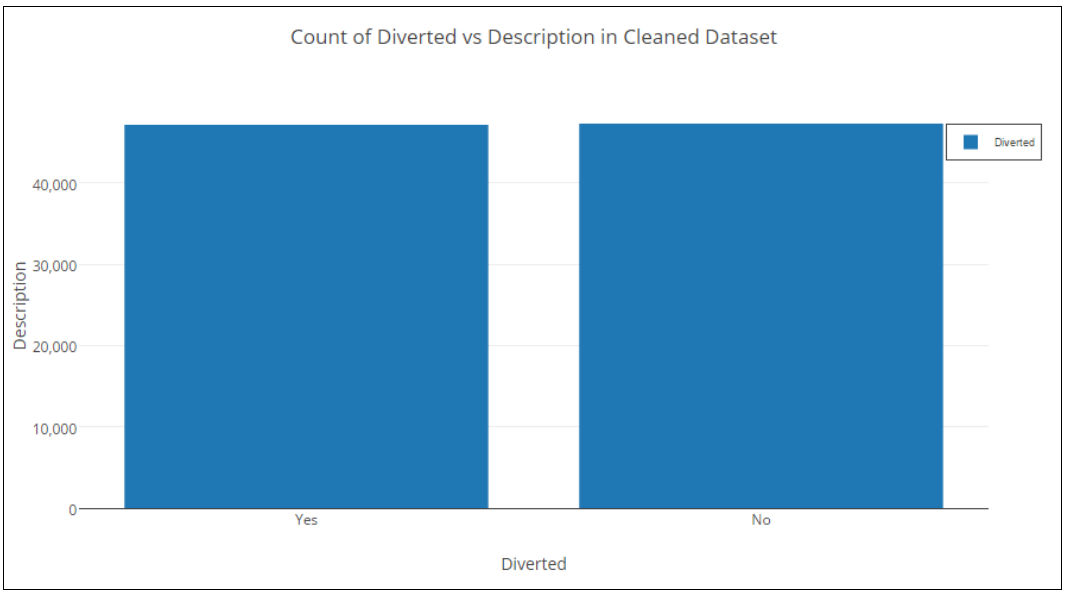

Fig 4: Data visualization (in %) of cleaned diverted data

Results

It can be seen from figures 3 and 4 that the classification of raw data using Decision Tree classifier is 83.15% whereas the classification, using Decision Tree classifier, after the data was processed is 100%. The confusion matrix in figure 4 shows that all the records with class 'YES' or '1' were classified as 'NO' or '0'. The reason behind this poor classification is that the number of records with class '0' in the raw data was a lot more than the ones with class '1' which causes the dataset to become biased towards class '0'. However, once the dataset was balanced, the accuracy increased to 100% indicating that the model is probably overfitted.

The figure 1 shows that the accuracy of the model using J48 on the substituted data is 95.95%. There is still a lack of domain knowledge to check if the records generated are correct or not.

The figures 5 show that the accuracy of the Naive Bayes classifier on the normalized dataset is 92.26% and the accuracy of the Decision Tree classifier on the normalized dataset is 95.97%.

```
Size of the tree :      79


Time taken to build model: 0.33 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       23297               100      %
Incorrectly Classified Instances         0                 0      %
Kappa statistic                          1
Mean absolute error                      0.0004
Root mean squared error                  0.0139
Relative absolute error                  0.0771 %
Root relative squared error              2.7752 %
Total Number of Instances            23297

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Yes
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     No
Weighted Avg.    1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000

=== Confusion Matrix ===

     a     b   <-- classified as
 11630     0 |     a = Yes
     0 11667 |     b = No
```

Fig 5: Decision Tree classification on Raw data

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       11667               83.1516 %
Incorrectly Classified Instances      2364               16.8484 %
Kappa statistic                          0
Mean absolute error                      0.2802
Root mean squared error                  0.3743
Relative absolute error                100      %
Root relative squared error            100      %
Total Number of Instances            14031

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    1.000    0.832      1.000    0.908      0.000    0.499     0.831     No
                 0.000    0.000    0.000      0.000    0.000      0.000    0.499     0.168     Yes
Weighted Avg.    0.832    0.832    0.691      0.832    0.755      0.000    0.499     0.720

=== Confusion Matrix ===

     a     b   <-- classified as
 11667     0 |     a = No
  2364     0 |     b = Yes
```

Fig 6: Decision Tree classification on Reduced data

```
Time taken to build model: 3.83 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       90586               95.9557 %
Incorrectly Classified Instances      3818                4.0443 %
Kappa statistic                          0.9191
Mean absolute error                      0.0619
Root mean squared error                  0.1985
Relative absolute error                 12.3725 %
Root relative squared error             39.7012 %
Total Number of Instances            94404

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.949    0.030    0.969      0.949    0.959      0.919    0.968     0.965     Yes
                 0.970    0.051    0.950      0.970    0.960      0.919    0.968     0.950     No
Weighted Avg.    0.960    0.040    0.960      0.960    0.960      0.919    0.968     0.958

=== Confusion Matrix ===

     a     b   <-- classified as
 44733  2391 |     a = Yes
  1427 45853 |     b = No
```

Fig 7: Decision Tree classification on Substituted data

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       87099               92.262  %
Incorrectly Classified Instances      7305                7.738  %
Kappa statistic                          0.8453
Mean absolute error                      0.0956
Root mean squared error                  0.2478
Relative absolute error                 19.1113 %
Root relative squared error             49.5648 %
Total Number of Instances            94404

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.953    0.107    0.898      0.953    0.925      0.847    0.971     0.970     Yes
                 0.893    0.047    0.950      0.893    0.920      0.847    0.971     0.964     No
Weighted Avg.    0.923    0.077    0.924      0.923    0.923      0.847    0.971     0.967

=== Confusion Matrix ===

     a     b   <-- classified as
 44896  2228 |     a = Yes
  5077 42203 |     b = No
```

Fig 8: Naive Bayes classification on normalized data

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       90595               95.9652 %
Incorrectly Classified Instances      3809                4.0348 %
Kappa statistic                          0.9193
Mean absolute error                      0.0618
Root mean squared error                  0.1983
Relative absolute error                 12.3514 %
Root relative squared error             39.6609 %
Total Number of Instances            94404

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.949    0.030    0.969      0.949    0.959      0.919    0.968     0.965     Yes
               0.970    0.051    0.950      0.970    0.960      0.919    0.968     0.950     No
Weighted Avg.  0.960    0.040    0.960      0.960    0.960      0.919    0.968     0.958

=== Confusion Matrix ===

     a     b   <-- classified as
 44734  2390 |     a = Yes
  1419 45861 |     b = No
```

Fig 9: Decision Tree classification on normalized data

Lessons Learned

1. Class Balancing

It is vital to balance all the classes before applying any classification algorithms to avoid any bias in the results. We observed that no matter what classifier we used the results were always wrong for the minority class as the model did not get enough data to train.

2. Handling Missing Values

Missing values must be taken care of using some technique such as imputing the value or substituting the value and if all fails we must resort to removing the instances. This step is important to obtain right as well as better results along with a better understanding of the data.

3. Overfitting

It is quite common that a model is overfit for the data. In the case of this project, the model was overfit to data when the missing values were removed from the data, and the classifier was applied to it. Overfitting might give full accuracy on the training and testing dataset but would fail miserably on real data. Hence, it is crucial that we keep track of results that are too good to be true.

4. Data Cleaning and Preparation

The lessons mentioned above point to a single and critical lesson that data cleaning and preparation is a significant part of the data analysis process. The raw data consists of noise, outliers, and missing data. Knowledge can be derived from this data only once it is free from the dirt and is left with the relevant information.

Conclusion

The results obtained by applying a decision tree classifier on the reduced data gave results too good to be true. The model was overfit to the data and therefore, was discarded.

The difference in the accuracies of Decision Tree and Naive Bayes classifiers was not much, the mean absolute, root mean squared, relative absolute and root relative squared errors of Naive Bayes classifier were higher than those of Decision Tree classifier. This situation might not be true for some other dataset which is why a model cannot be tagged as better than the other in general. Moreover, the models act better on a dataset that is cleaned properly using many techniques. It was seen in the section results how the results of the classifier improved with each cleaning step. But it is important to keep a domain expert in the loop to ensure that the data used for classification contains the necessary records and does not contain noise.

Current Status

Currently, the classification of the dataset is done using Decision Tree and Naive Bayes classifiers. The dataset has been pre-processed using techniques like substitution and removal of missing data and SMOTE for data balancing. The dataset has been normalized using WEKA`s default normalizer. The accuracies of the classification model on all the datasets have been discussed. The dataset has been visualized using 'plot.ly'.

Future Work

In the future, different pre-processing techniques could be merged and then SMOTE could be applied for balancing the class. Different class balancing methods could also be used and tried to see the difference in the output results. Also, a real-time application could be created wherein the inputs would be the flight number/name, source airport, destination airport and the time to fly. The application would also use the weather conditions as a parameter to determine the diversion of the flight. The output of the application would be a probabilistic result whether the flight would be diverted or not. It is believed that taking the weather data into account would improve the results.

References

1. Carrejo C. Are planes diverted often? The Air France flights weren't the only ones rerouted. Bustle. 2015 Nov 18 [cited 2025 Apr 19]. Available from: https://www.bustle.com/articles/124628-are-planes-

diverted-often-the-air-france-flights-werent-the-only-ones-rerouted.
2. U.S. Department of Transportation, Bureau of Transportation Statistics. TranStats: Home Drill Chart. Washington, D.C.: U.S. Department of Transportation; [cited 2025 Apr 19]. Available from: https://www.transtats.bts.gov/HomeDrillChart.asp.
3. American Statistical Association. Data Expo 2009 - Airline on-time performance. Washington, D.C.: American Statistical Association; 2009 [cited 2025 Apr 19]. Available from: https://community.amstat.org/jointscsg-section/dataexpo/dataexpo2009.
4. Plotly. Data apps for production. Montreal (QC): Plotly Technologies Inc.; [cited 2025 Apr 19]. Available from: https://plot.ly/.
5. U.S. Department of Transportation, Bureau of Transportation Statistics. Airline On-Time Performance Data, 1987-2008. Harvard Dataverse; 2009 [cited 2025 Apr 19]. Available from: https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/HG7NV7.
6. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP. SMOTE: synthetic minority over-sampling technique. J Artif Intell Res. 2002 Jun 1;16:321-357.