

Machine Learning algorithm for alternative airport destinations

BSc in Software Engineering

Authors:

Morten Krogh Jensen
Thomas Steinfeldt Laursen

mortj18@student.sdu.dk
tlaur18@student.sdu.dk

Supervisor:

Sanja Lazarova-Molnar

slmo@mmmi.sdu.dk

External supervisor:

Henrik Hansen

henrik@foreflight.com

Submission Date: 1st of June 2021

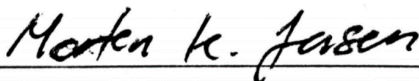


The Maersk Mc-Kinney Moeller Institute

University of Southern Denmark

Designate

<i>Project title:</i>	Machine Learning algorithm for alternative airport destinations
<i>Project members:</i>	Morten Krogh Jensen Thomas Steinfeldt Laursen
<i>Supervisor:</i>	Sanja Lazarova-Molnar
<i>Program:</i>	BSc in Software Engineering
<i>Course:</i>	Bachelor Project (T510012101)
<i>Educational Institute:</i>	University of Southern Denmark (SDU)
<i>Institute:</i>	Maersk Mc-Kinney Moller Institute
<i>Project period:</i>	2021-02-01 - 2021-06-01



Morten Krogh Jensen

mortj18@student.sdu.dk



Thomas Steinfeldt Laursen

tlaur18@student.sdu.dk

1 Abstract

When planning a flight, the pilot must select an alternate airport which functions as a place to land in case the primary destination becomes unavailable. In most cases, selecting an alternate comes down to the pilot's experience regarding which alternates are fitting for the type of flight. In reality however, several factors have to be considered such as sufficient runway length, the presence of navigational equipment, and much more making it close to impossible to formulate exact criteria for every type of flight.

This project aims to use records about previously chosen alternates for various flights to develop a system capable of recommending valid alternates through machine learning. The effectiveness of three different classifiers have been tested throughout this project: Decision Tree, Random Forest, and Multilayer Perceptron.

ForeFlight has provided a dataset containing approximately one million flight plans recorded world-wide over the duration of mid-2019 to start-2021. Through an interview with a pilot, the most important features from the flight plans were identified. These include the destination airport since the alternate often should be nearby and the aircraft type since it affects the required runway length. The effectiveness of using the raw features as inputs were tested as well as various engineered versions of them.

Experimentation showed the MLP classifier achieved the best results of the three tested classifiers. The best result was produced when training the model on the precise coordinates for the destination airport and the aircraft's wake turbulence category. Even though the achieved accuracy of 83% was not the highest seen through the experimentation, this classifier and combination of features produced results with overall more valid airports centered around the destination.

Compared to ForeFlight's existing Alternate Advisor, the new model performs better when tested on the same dataset. An important difference between the two is that the new model takes the flown aircraft into account when making suggestions. However, a limitation to the system is its inability to predict alternates not seen during training limiting its use in places with less available data.

2 Acknowledgements

We would like to thank ForeFlight for giving us the opportunity to work with this problem and providing the necessary flight plan data. Especially Henrik Hansen for setting up interviews with colleagues from Texas and always being available to assist.

Thanks to Anders Weber Borgermann, the pilot from Air Alsie, for providing information and clarity regarding the pilots' decision process when choosing alternate airports.

We would also like to thank our project supervisor from SDU, Sanja Lazarova-Molnar, for the guidance and assistance she provided.

Contents

1	Abstract	3
2	Acknowledgements	4
3	Introduction.....	7
3.1	Context about the ForeFlight company.....	7
3.2	Problem with choosing alternate airports.....	7
4	Background and related work	8
4.1	Recommender systems	8
4.2	Alternate Advisor.....	9
5	Approach for recommending alternative airports	11
5.1	Data available for use	11
5.1.1	Data Analysis.....	11
5.2	Data Pre-processing.....	14
5.2.1	Feature Selection.....	15
5.2.2	Feature Engineering	16
5.2.3	Further pre-processing	19
5.3	Classification algorithms.....	24
5.3.1	Decision Tree	24
5.3.2	Random Forest	29
5.3.3	Multilayer perceptron	34
5.4	Method of evaluation.....	39
5.4.1	Accuracy measurement.....	39
5.4.2	Visual evaluation	40
5.5	Technical implementation.....	41
6	Evaluation	42
6.1	Experimental evaluation.....	42
6.1.1	Decision tree.....	42
6.1.2	Random Forest	45
6.1.3	Multi-Layer Perceptron	49
6.2	Qualitative evaluation	52
6.2.1	Visual model comparison	52
6.2.2	Pilot user testing.....	54
7	Discussion	55
8	Conclusion	58

9	Perspectivation	59
10	References	60
11	Appendix	63
11.1	Appendix 1 - ICAO Model Flight Plan Form (Federal Aviation Administration, 2021)	63

3 Introduction

3.1 Context about the ForeFlight company

This bachelor's project has been done in cooperation with the software company ForeFlight. They specialize in state-of-the-art flight planning technologies for both private and commercial pilots across the entire world¹. These technologies include advanced route calculation capable of finding the optimal path between point A and B. This ensures the shortest travel time resulting in less time wasted, lower costs, and a reduced environmental impact.

3.2 Problem with choosing alternate airports

Every flight is planned beforehand by the pilot creating a so-called flight plan. The pilot has to enter information about the flight including the destination airport, aircraft type, fuel amount, and more. Every flight plan also must include an alternative airport called an alternate. An alternate is an airport used by flights for diversion if the primary destination is unavailable for landing. This can be due to weather or other factors such as traffic at the destination, accidents and so on. Choosing an alternative in an unfamiliar location can be difficult. The pilot needs to ensure sufficient runway length, sufficient navigational equipment, rescue equipment (fire trucks etc.), transport to get passengers out, a hangar to put the airplane of that size in, maintenance service, fuel availability and so on. The pilot's choice of alternate often comes down to their "tribal knowledge", knowing that for this destination, this airport is the typical alternative for this kind of aircraft. Therefore, it is difficult to formulate exact criteria that works in every situation.

ForeFlight has previously created a machine learning model to suggest alternates based on historical data. However, this model has a couple of issues:

- Never formally validated and does not fully account for the difference in aircraft type impact on alternate selection.
- Only works in the US. Could add training data for Europe.
- The mobile app is developed in objective-c. However, their web products have a Java backend and another of their products has a C# backend. The model is only used in the ForeFlight Mobile app as it was never generalized to work in other products requiring a JAVA and C# integration.
- The model is not continuously updated.

Considering the issues above, it has been possible to formulate the following questions that should be answered throughout this project.

¹ (AviationCloud, 2021)

Is it possible to make a solution that reliably determines a fitting alternate airport similar to what a professional pilot would?

- What is a suitable test for determining the solution's usability in the real world?
- How can the solution be compared to ForeFlight's current solution?
- How can the solution be made usable across all of ForeFlight's platforms?
- Is there a way to abstract the solution so it would work in any part of the world?
- Can the solution be continuously updated, and is there a way to allow new data weighting more than old data to allow the solution to adopt to new patterns?

4 Background and related work

4.1 Recommender systems

The completed system would serve as a supplementary tool for the pilots when creating the flight plans by providing a small list of alternate suggestions for the pilot to choose from. Such a system is often referred to as a Recommender system which you often encounter at web services like Netflix when you are suggested new movies to watch. However, instead of suggestions being based on your previously watched movies, the suggested alternates should be based on information about the flight and knowledge about previous choices from similar flights. This is to make sure the alternates are valid options by, for example, not being too far away.

Recommender systems come in several varieties². The most relevant for this project is the content-based method. The content-based recommender system suggests items similar to those other users (in this case, flights) selected by generalizing the users into meaningful characteristics called features. In the domain of this project, the term "users" does not refer to people but flights since the system should provide suggestions based on flight information. In the case of flights, features could be everything from the geographical location to the type of fuel used. A model is then trained to find any patterns between the features and the selected items.

Compared to other varieties of recommender systems, the content-based method has easier with handling new users that has not selected any items yet. The content-based method circumvents this problem (called the cold start problem) when reducing the user into meaningful features making the new user comparable to existing ones. However, it is still vulnerable if it is exposed to previously unseen features and new items to suggest, but this problem becomes less apparent the more experience the system gets. Another weakness

² (Rocca & Rocca, 2019)

of the content-based recommender system is that it tends to be less personalized because it reduces every user into features. Two users with similar characteristics do not necessarily have the same preferences.

The recommender system discussed in this section is often referred to as item-centered since the model focuses on predicting the probabilities of items based on user features. However, a recommender system can also reduce each item into features and predict the probabilities of each user wanting it. This method is referred to as user-centered. While the item-centered approach is often less personalized, the user-centered is only useful when every user has selected many items. If not, the model will have difficulty learning the preferences of the less active users.

The item-centered content-based approach is fitting in the domain of suggesting alternates. In most recommender systems, each user will interact with several items making it possible to build a profile around them. However, flights cannot be treated the same way as users since two flights are rarely the exact same. Therefore, it is relevant to reduce each flight to features and suggest alternates based on previous choices from similar flights thus preventing the cold start problem.

4.2 Alternate Advisor

As described in the introduction, ForeFlight already has a solution for recommending alternate airports called Alternate Advisor. It works by generating a list of possible alternates when pilots fill out the flight plan³. The system narrows down the list of suggestions by taking range and fuel consumption into account as well as the supported type of approaches, forecast weather conditions, and the pilot's previously chosen airports. An image of the Alternate Advisor in action is shown in Figure 1.

Alternate Advisor shows a map with a line and the distance to each of the recommended alternates. The pilot is shown several rows of information per alternate. The first row shows the weather forecast data for that airport. The second row shows information regarding if the alternate airport has a tower and the length of the longest runway. The last line shows the different type of instrumental approaches that the airport offers. The right side shows how much time and extra fuel is necessary if that given alternate is added to the flight plan.

³ (ForeFlight, 2019)

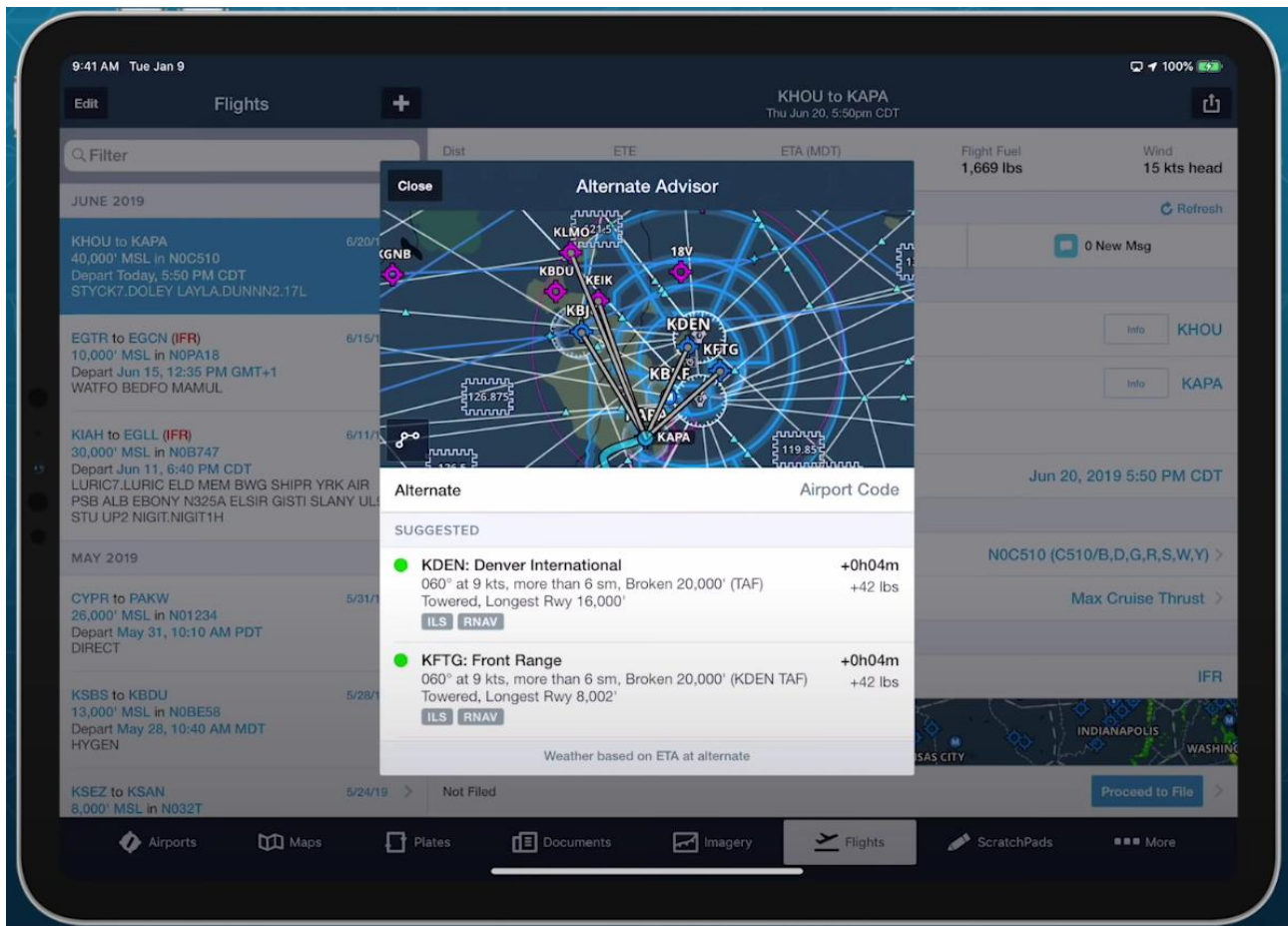


Figure 1 - ForeFlight's Alternate Advisor⁴

Based on the conversations with ForeFlight, the current Alternate Advisor system they offer to the pilots is not perfect. Their solution is divided in two, where the mobile platform uses a model based on machine learning, but not on their other platforms. Furthermore, the machine learning model does not take the impact that different aircraft types have on the decision process into account and currently only works for flights planned in the United States⁵.

⁴ (ForeFlight, 2019)

⁵ (Hansen, 2021)

5 Approach for recommending alternative airports

This section will go through the required work prior to training the model and the different considerations that had to be made. It will cover the data selection, preprocessing, choice of classification algorithms, and the method of evaluating the trained models.

5.1 Data available for use

To create a machine learning model that can reliably suggest valid alternate airports, it had to be trained on relevant data. Through ForeFlight, the following data sources have been contributed.

Flight plan database

A collection of flight plan data from the FAA SWIM (Federal Aviation Administration, System Wide Information Management) system. As stated earlier, flight plans are created by pilots before takeoff. The pilots themselves enter information about the flight such as destination, aircraft, fuel amount, flight rules, and alternate. These choices have been recorded in the flight plan database. In addition, information about the date and time and the weather is included for each flight plan.

This database contains millions of entries each including all relevant data for a flight plan. The flight plans are primarily concentrated around the United States with fewer entries from Europe as ForeFlight only captures a subset of European flight plans. The database includes even fewer flight plans from the rest of the world.

For this project, one million entries from this database have been provided. These flight plans represent the choices pilots made when planning the flight and has acted as the primary dataset the machine learning has been trained upon.

Airport database

A database of all airports in the world including many different attributes on airport data. From this database, information about an airport's location, runways, and features can be extracted.

Aircraft database

An aircraft database that allows mapping of aircraft types to data such as weight and runway requirement.

5.1.1 Data Analysis

To understand the data from ForeFlight, it has been important to analyze what the data contained, how the data could be related to the problem, and if the dataset was balanced or in any way biased.

This project has been given one million entries from the flight plan database. These flight plans contain information about the flight entered by the pilots typically before departure. An example of a completed flight plan following the standard defined by the International Civil Aviation Organization (ICAO) can be seen

under the Appendix 1 section. A flight plan contains numerous pieces of information about a flight, but only the most important ones will be explained here.

Departure aerodrome: The unique name of the flight's origin airport. Its name is a four-letter code following the standard defined by ICAO.

Destination aerodrome: The unique name of the flight's destination airport. Its name is a four-letter code following the standard defined by ICAO.

Alternate aerodrome: The unique name of the flight's alternate airport. Its name is a four-letter code following the standard defined by ICAO.

Type of aircraft: The unique name of the aircraft model used for the flight. Its name is a four-letter code following the standard defined by ICAO.

Wake turbulence category: One letter indicating the Wake Turbulence category of the aircraft. The wake turbulence category is determined by the weight of the aircraft at take-off. Light ("L") for aircrafts weighing 15,500 lbs or less. Medium ("M") for aircrafts between 15,500 and 300,000 lbs. Heavy ("H") for aircrafts above 300,000 lbs. Certain huge aircrafts with weights above 1 million lbs are specified as Super ("J").

Time: The time and date the flight plan was submitted to an air traffic services reporting office. This is typically done right before departure.

Flight rules: Whether the flight is performed with Visual or Instrumental Flight rules. During VFR flights, the pilots rely on what they can see outside the aircraft for navigation. With IFR flights, pilots rely on navigational instruments.

Destination airport weather: Not actually a piece of data in a standard flight plan but have been attached to all the flight plans provided in the database. The weather condition at the destination airport described by which flight rules are possible. From bad to good weather the values range "IFR", "Low IFR", "Marginal VFR", or "VFR". Or if unknown, the value is "Flight category unknown".

Of all these pieces of data, the most important is **Alternate aerodrome** as this project revolves around predicting this part of a flight. It has been interesting to take a deeper look at this piece of the data to gain a better understanding. First, it has been explored how many different alternates the dataset contained and how often each of them was represented. Figure 2 illustrates a histogram showing the differences in alternate representations from the one million flight plans in the dataset. This graph shows a great deal of imbalance amongst the alternates. Out of 8797 total alternates represented, almost half (4277) are represented ten or fewer times. Furthermore, a small number of alternates have been chosen very often compared to the

others. The reason for this imbalance might be a combination of these being the more popular choices and the dataset containing more flight plans near these airports.

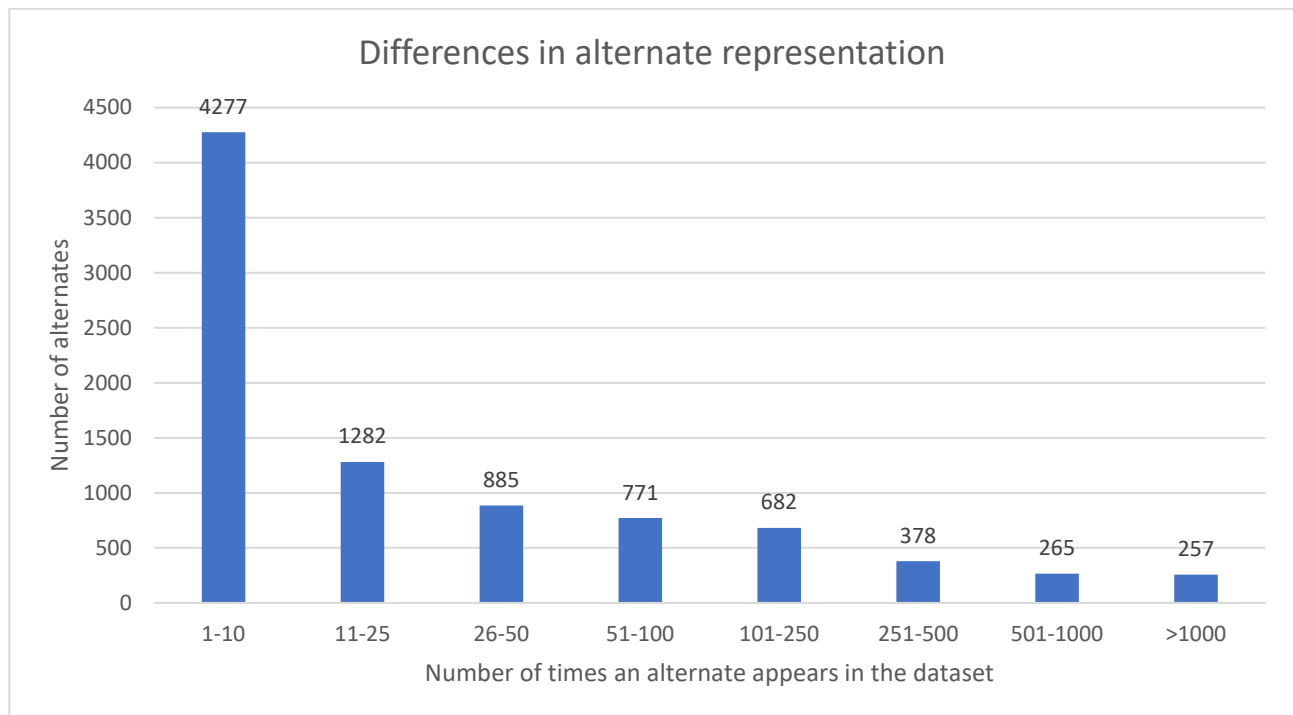


Figure 2 - The distribution of differently represented alternates in the provided flight plans

An unequal class distribution usually poses a problem when used to train machine learning models. Most classification algorithms become biased towards the majority classes and less able to predict the minorities⁶. The machine learning could favorize one alternate over another where both would be valid suggestions for the pilot to choose from. It could result in an underrepresented alternate being left out even though it was the optimal choice in a few edge cases as seen in the data.

According to the business supervisor, the provided flight plans were primarily from the US and fewer from Europe. Further analysis confirmed this. Figure 3 below shows a distribution of the flight plan alternates around the world. The distribution clearly shows how the majority of the flight plans have alternate airports in the United States and have large empty spots in other parts of the world, for example in Russia and China. Out of a total 994,889 flight plans, 720,015 have registered destinations in North America (USA and Canada). 116,014 have destinations in Europe, and the remaining 157,553 were distributed around the rest of the

⁶ (Brownlee, A Gentle Introduction to Imbalanced Classification, 2019)

world. This shows a bias towards Europe and especially the US potentially making it difficult for the model to predict on the underrepresented locations in the rest of the world.



Figure 3 - The geographical distribution of the provided flight plans

5.2 Data Pre-processing

Understanding what the flight plan data consists of is not enough to train a machine learning model from. It has been necessary to consider how a model would perceive and benefit from different features. In some cases, it may be helpful to alter the features into newer and more meaningful ones making it easier for the model to detect patterns. This section will go through the process of feature selection where the most relevant features of flight plans have been identified. Training models with a dataset containing irrelevant and redundant data can reduce its accuracy⁷. Afterwards, feature engineering is covered where the relevant data from feature selection has been preprocessed to become more meaningful.

⁷ (Shaikh, 2018)

5.2.1 Feature Selection

From the beginning, assumptions were made to which of the features would be most important based on interviews with the company supervisor. The destination airport was expected to have a big influence on the choice of alternate since they must be somewhat close to each other. It would not make any sense to select an alternate airport in New York when your destination is Los Angeles.

The aircraft type used for the flight can also influence what alternate the pilot chooses. Different aircrafts require different lengths of runways. Typically, heavier aircrafts will need longer runways when landing than lighter aircrafts. For example, a pilot flying in a gigantic Airbus 388 is unlikely to select a small airport with short runways as the alternate.

To gain a better insight in what pilots use to determine alternates for their flights, a pilot from Air Alsie Airlines was contacted and qualitatively interviewed⁸. With over seven years of experience and around 1000 total flights in both commercial and private jets, he was believed to be a credible source. He confirmed the correlation between alternate and destination airport stating that alternates rarely are more than 300km away from the destination. However, it is sometimes necessary when the destination airport is far away from any other airports (e.g., on a small island). In these cases, pilots would choose a so-called en-route alternate that they will pass on the way to the destination. These are only used as destination alternates in some extreme cases. To represent the destination airport when training the machine learning model, the flight plan's "Departure aerodrome" has been used. It depicts a geographical area around which alternates should be considered.

The pilot also confirmed that the type of the aircraft is important. They are not allowed to choose an alternate before confirming that it has a runway long enough to land on. To allow the model to distinguish between different aircrafts, the flight plan's "Type of aircraft" has been used. This is the unique code of the aircraft and would serve well to distinguish them from each other. However, "Wake turbulence category" divides them into categories based on their weights which could be useful also.

Other than the destination airport and aircraft weight, the pilots consider many properties relating to the alternates in the area. Typically, they want to choose the biggest airports based on how much traffic they support and the length of their runways. A good indicator that an airport supports plenty of traffic is usually if it has a tower. Furthermore, the airport should preferably support precision approaches where landing is

⁸ (Borgermann, 2021)

performed with instrumental guidance. The importance of an airport having precision approaches also depends a lot on the weather. In good weather (VFR available), instrumentally guided landing is not necessary. Even though these characteristics are important for the decision process, they are not easily incorporated into the model since they are tied to the alternate airport. The model cannot use data from an airport before it has predicted it.

Knowing the pilot's considerations helped narrowing down what data from the flight plan was relevant to use when training a machine learning model. Other than the features already mentioned, there exists plenty whose importances are initially unknown. When training the models, experimentation has been done to see if they could detect any patterns not easily seen by people. Examples of the features experimented with include the destination weather condition and flight rules. From the flight plan's destination aerodrome, it is also possible to extract other possibly useful information such as the length of the airport's longest runway and whether it supports precision approaches. Whether these makes a difference is explored later.

5.2.2 Feature Engineering

5.2.2.1 *Destination airport representations*

In the flight plan database, destination airports are represented by their unique ICAO code. Examples of this data are "KHOU" for Austin-Bergstrom International Airport and "EKCH" for Copenhagen Airport. This feature is of categorical nature as there is a finite amount of possible values limited by the number of airports around the world. They are considered nominal since there does not exist a natural order to them. However, the first letter does indicate which country/region they are from (K for airports inside USA). Most machine learning algorithms require numerical attribute values. It was important to be careful with converting the destination airport codes to numerical values since it would introduce an order to an otherwise unordered set of values.

It is also interesting to consider other ways of representing the destination of a flight. Having access to an Airport database, it has been possible to replace each airport name with its respective coordinates. This introduced two new attributes of ordinal numeric nature. This change could bring several benefits. First, it would make the attribute compatible with most machine learning algorithms. Second, the feature could be more meaningful for the ML model. Coordinates are easier to relate to each other than names and would make it easier for the model to detect patterns. Lastly, it should prepare the model for unseen input. After training, if the model is exposed to a previously unseen airport ICAO, it would not be able to compare it against the codes trained upon because of their nominal nature. This is not a problem with ordinal values such as coordinates.

Another way to represent the destinations is to group them together based on location. This method would divide the map into several sections each with a unique ID. An airport would then be given the ID of the

section it resides in. This transformation would keep the feature as a categorical value. Like before, it would be mainly considered nominal as there is no natural order to them even though you could give nearby sections similar ids. Compared to the exact coordinates, this change would discretize the feature. Coordinates are continuous features with infinitely many possible values and reducing them into a finite number of fields would simplify the problem.

A simple way to group airports would be to divide the world into squares of sizes 1 degree latitude times 1 degree longitude similar to the squares created by the latitude and longitude lines of a world map. Figure 4 illustrates how this method would divide the cities Houston, San Antonio, and Austin into separate squares. The two airports KHOU and KIWS in Houston reside within the same square and would not be distinguished from each other when used as destinations in a flight plan. This makes sense since the suggested alternates should be similar for destinations this close to each other. Though, this method has the risk of dividing an area with a high density of airports, e.g. a city, over the middle resulting in similarly located airports to be divided into different squares. This is almost the case of Houston whose most northern part lies in the square above. A way to possibly bypass this problem could be to make all squares overlap slightly. Airports on the edges of squares would then belong to both squares. However, this method has not been tested due to time limitations.

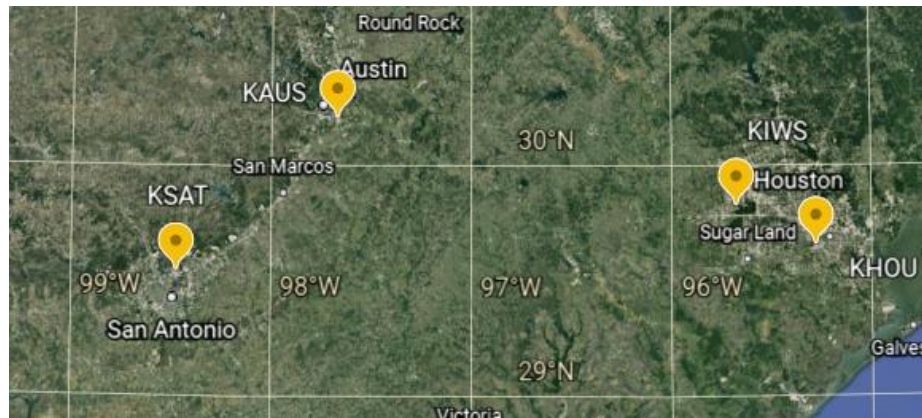


Figure 4 - Illustration of dividing airports into squares

5.2.2.2 Aircraft representations

As stated in section 5.2.1 about Feature Selection, the flown aircraft can be represented by either its model's unique identifier or the wake turbulence category. Like the destination airport, aircraft type is a nominal categorical value represented as a four-digit ICAO code. Examples are "B747" and "GLF6". Being nominal and a string constitutes the same problems as for the destination airport since it cannot be easily changed to numerical values without introducing unwanted order. It is also possible to argue that distinguishing between

every aircraft model would be unnecessary. As stated by the pilot in the interview, it is important to include the aircraft into the consideration since it requires a certain length of runway. However, many different aircraft models have the same runway requirements. From the aircraft database, it has been possible to look up the minimum runway length requirements for different aircrafts. This abstraction of the aircraft type could be more meaningful to the model and help it relate to any new aircrafts it might encounter after training. However, before using this data, it was necessary to know exactly what it meant and how it had been measured (e.g., with the aircraft at maximum or minimum load and dry or wet runway conditions). The business supervisor was inquired regarding this and told not to rely on that value; “It’s a pure guesstimate.”⁹. Because of the uncertainty of the minimum runway length, it was chosen not to trust it and has been excluded from the model.

As stated earlier, the weight of the aircraft has an impact on the landing distance. Therefore, considerations were made regarding training the model on the MLW (Maximum Landing Weight) for the aircrafts. This value indicates the maximum weight limit an aircraft can land with due to design and operational limitations. However, this value would not accurately represent the weight of the aircraft but an upper limit. An aircraft’s weight varies from flight to flight due to differences in passengers, cargo, and fuel. One could argue that using an upper limit would add a margin of safety since flights will never have their weights underestimated. Though, making these estimations would mean that the assumed and real value differ a lot. A possible solution would be to use the attribute wake turbulence category from the flight plans. This is an ordinal categorical feature consisting of one of four letters: “L”, “M”, “H” and “J”. This is meaningful for the aircraft’s runway requirements since wake turbulence depends on weight which affects runway requirements. This feature could be preferable over MLW for the aircraft models since it is decided with the aircraft’s cargo in mind and no assumptions has to be made. Furthermore, the feature is ordinal and can therefore be mapped to numerical values while still having the same natural ordering.

The effect that each of the alterations discussed in this section have had to the models’ performances is explored later in the report during the experimental evaluation.

⁹ (Hansen, 2021)

5.2.3 Further pre-processing

5.2.3.1 Dealing with nominal features

Some of the features discussed above consists of nominal string values (destination airport codes and aircraft model codes). Most machine learning models require the values to be numerical when used for training. In this case, the LabelEncoder from the scikit-learn preprocessing library has been used to map each value to a number between 0 and $n_{classes} - 1$.¹⁰

The features that are required to be encoded using the LabelEncoder are considered nominal. This introduces a problem as assigning them numerical values will have the model automatically perceive the values as ordinal even though no natural order exists between them. A technique used to bypass this problem is One Hot Encoding¹¹. Consider a dataset consisting of the following entries.

ID	Destination airport	Alternate Airport
0	KHOU	KAUS
1	KCPS	KSTL
2	KAUS	KEDC
3	KHOU	KGYB

Table 1 - Example data before One Hot Encoding

There are a total of three different **Destination airport** values (KAUS, KCPS, KHOU). Normal LabelEncoding would assign them values 0, 1 and 2 thus adding an unwanted order. One Hot Encoding will on the other hand introduce a new feature for each distinct value of the original feature. The dataset from Table 1 would then take the form shown by Table 2. Each new feature is a binary value indicating whether or not the destination airport is one particular value. Representing the destination airport this way keeps the nominal relationship between the values. However, in certain cases it causes more harm than good. One Hot Encoding features with high cardinality will result in a dataset with a huge number of columns and can result in the curse of dimensionality¹². The problem of multicollinearity is introduced where there is a dependency between the features which can cause issues with certain machine learning models. Furthermore, the more features in the table, the more time and resource consuming it is to train the model. The flight plan data available contains more than 14,000 different destination airports and 800 different aircraft models.

¹⁰ (Scikit-learn, 2021)

¹¹ (Brownlee, Why One-Hot Encode Data in Machine Learning?, 2020)

¹² (Kumar, 2021)

Therefore, One Hot Encoding these variables was not considered a viable option and LabelEncoder has been used for the experimentation with the nominal values.

ID	KAUS	KCPS	KHOU	Alternate Airport
0	0	0	1	KAUS
1	0	1	0	KSTL
2	1	0	0	KEDC
3	0	0	1	KGYB

Table 2 - Example data after One Hot Encoding

5.2.3.2 Concentrating on North America and Europe

As described in section 5.1.1, the provided data is primarily centered around North America and Europe. The data inside these regions has been isolated from the rest to prevent confusing the machine learning with sparse data from the rest of the world. This does not conflict with the requirements set by ForeFlight since the model should primarily focus on predicting alternates in the US and Europe. All flight plans whose destinations starts with “K”, “C”, “E”, and “L” have been kept while the rest have been filtered away. These letters account for all airports in the US, Canada, northern and southern Europe. The flight plans have been filtered based on destination and not alternate because destinations close to the border of for example Mexico should be able to choose alternates in both countries.

Removing all flight plans outside of North America and Europe reduces the number of total flight plans by approx. 16% from 995,000 down to 836,000. However, the number of distinct destination airports in the dataset is reduced by almost 61% from 14,769 down to 5765. The bigger reduction in destination airports compared to the percentwise smaller reduction in the total number of flight plans, shows that the excluded destinations were underrepresented in the dataset.

5.2.3.3 Removing missing data

The flight plan dataset provided by ForeFlight is not entirely complete. In fact, some features have missing values in over half of the almost one million entries. Machine learning models cannot be trained on missing values meaning that these values must be replaced or have their entries deleted entirely.

As discussed in section 5.2.1 about feature selection, the features “destination airport” and “aircraft model code” should have a meaningful influence on the choice of alternate. Only 1307 of the entries are missing the destination airport; however, the aircraft model is missing for around 540,000 of the entries. It would not be safe to replace the missing aircraft models with an assumed value like a medium-sized aircraft or the most frequently appearing aircraft. It would introduce too many assumptions since the missing values account for over half of the entire dataset. The solution deemed reasonable was to delete all entries missing

these values. It would still leave around 380,000 complete flight plans which should be plenty to successfully train a classifier.

It was important that the data left after removing the missing values would give a realistic representation of the data provided. The data should be affected roughly the same way as it would if 540,000 random entries were deleted. The reason for the missing data is essential to make sure that a new bias would not be introduced to the dataset when removing them¹³. Ideally, the data should be Missing Completely At Random (MCAR). Here, the values are missing non-systematically thus still representative of the underlying distribution. First, it was important to examine whether North America or Europe contains most missing values. After removing all missing values, the number of flight plans with destinations in North America reduces by 53,07% while it reduces by 62,32% in Europe. This is not a big difference and shows the missing values are not caused by the geographical location.

Secondly, it should be examined if removing the rows with missing values had any effect on the chosen alternates. If there is no correlation between the rows with missing values, the number of times each airport is selected as alternate should be affected by an equal factor for all airports. However, this was not entirely the case for the provided dataset. The histogram illustrated in Figure 5 shows the relative changes in alternates' representations when removing missing values. So, if an airport had been chosen as alternate 10 times before deleting the rows but 4 times afterwards, it would have a relative change of 60%. The graph shows that every airport is not affected equally indicating that the rows with missing values cover a certain selection of flight plans, instead of them being uncorrelated. Furthermore, the histogram shows that approximately one thousand airports have a relative change of 95-100% and are therefore not used as alternate after the removal. However, after further examination, the majority of these airports are the ones that have been chosen as alternates only once or twice meaning they are unlikely to be important. Still, it is not ideal to lose these alternates since they have been deemed valid by pilots in a few cases. Despite this loss, it was considered to cause less harm than replacing the missing values would have.

¹³ (Open Data Science, 2019)

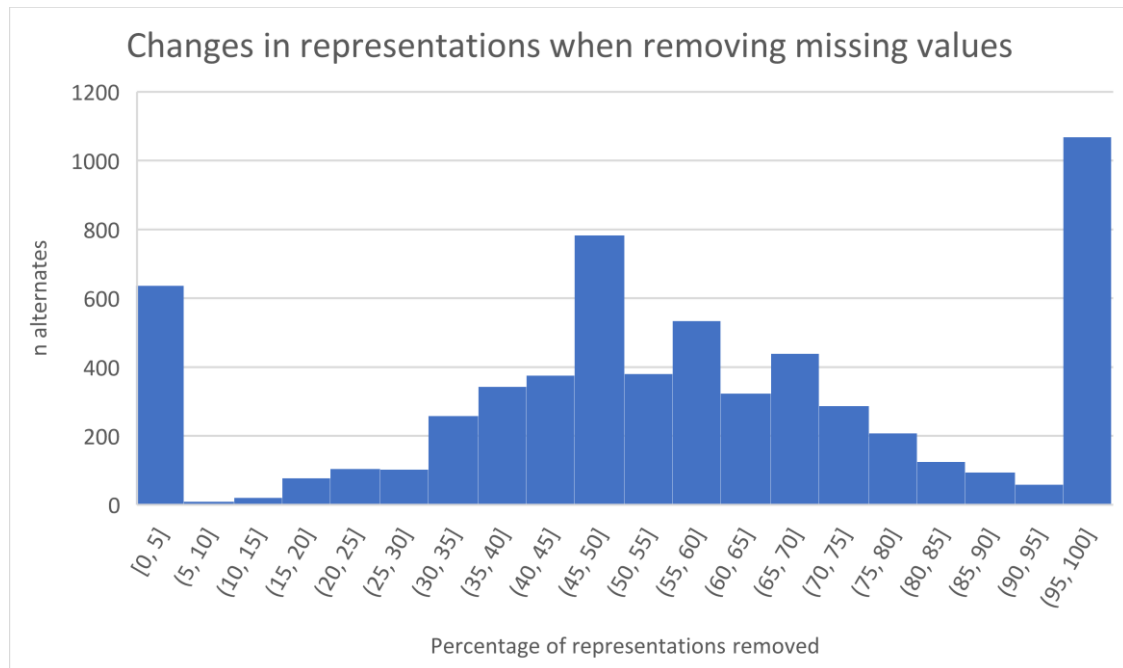


Figure 5 - Distribution of changes in alternate representations after removal of null values.

5.2.3.4 Dealing with outliers

When preparing for training it is important to examine the data for any outliers as they can lead to less accurate models¹⁴. Flight plans outside North America and Europe have already been removed since the rest of the world was not covered enough by the data. The remaining data should be examined for any flight plans with errors or other anomalies not representing real world data. After all, the data reflects the pilots' choices, and taking the human factor into consideration, some irregularities should be expected.

Figure 6 illustrates the sorted distance between the destination and alternate for all provided flight plans. It shows that the vast majority of flights have chosen an alternate between 10 and 400-ish kilometers from their destination airport. This agrees with information given by the pilot during the interview. Inspecting further, the dataset contains 2511 flight plans where the destination and alternate are registered as the same airport. These flight plans were invalid and therefore removed from the dataset.

Analyzing the flight plans with long distances between destination and alternate generally shows some illogical choices. Sometimes the alternate airport is in a different country from the departure and destination airport with seemingly no good reason. In some cases, the alternate might be an en-route alternate like the

¹⁴ (Tripathi, 2020)

ones mentioned by the interviewed pilot meaning the flight plan represents a valid choice. However, since these are so rare and the model should focus on destination alternates, it was decided that all flight plans with more than 1000km between destination and alternate airport should be discarded.

It would also be interesting to examine if any flight plans have registered an alternate too small for the flown aircraft. The point of including the aircraft in the decision process is to make the model aware that some aircrafts cannot land everywhere. Having outlier flight plans saying otherwise could cripple this aspect of the model. Unfortunately, there has been no easy way to check for these since the data on aircrafts' minimum runway length requirements could not be trusted.

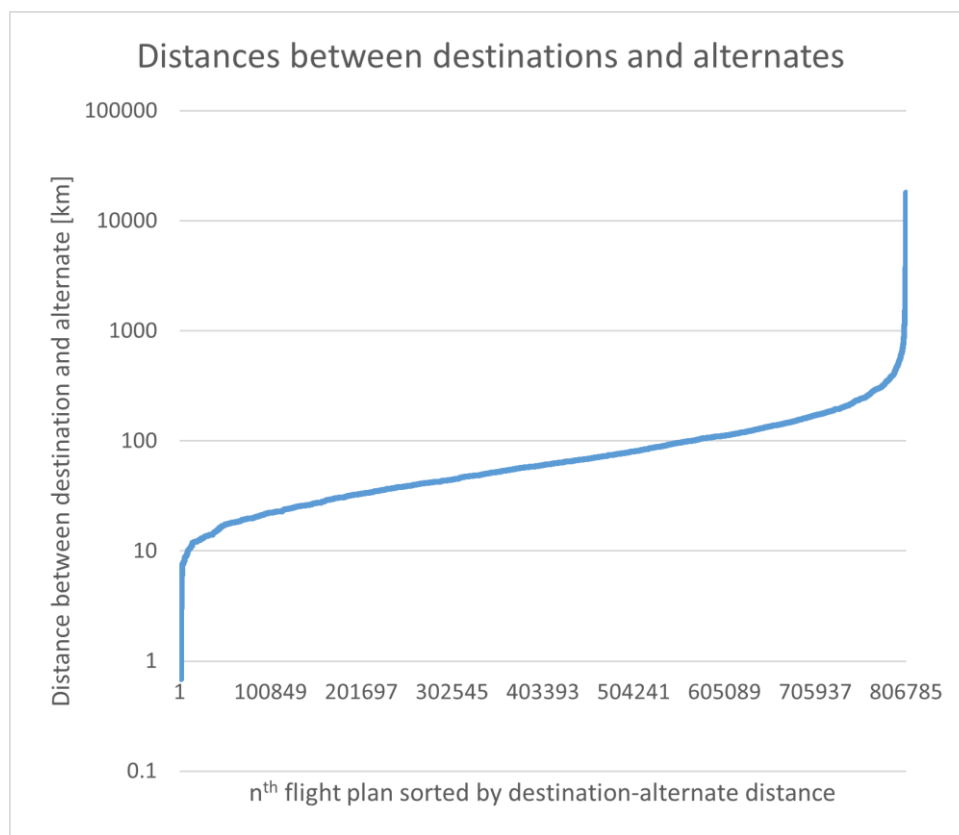


Figure 6 - Sorted distances between destination and alternate for all flight plans

5.3 Classification algorithms

As explained in earlier sections, the data provided contains labels indicating the chosen alternate for a given flight. The task of predicting this alternate can be seen as a classification problem. There exist several algorithms made for predicting labels, and three have been explored throughout this project. This section will describe the essence of each of them.

5.3.1 Decision Tree

The general idea behind the Decision Tree classifier is to ask a series of questions about a set of attributes to determine the class label.¹⁵ One can imagine it as a long series of if-else statements ending on different answers based on the route taken within the conditional statements. The Decision Tree consists of three different types of nodes: the root node, the decision node, and the leaf node. The leaf nodes reside in the bottom of the Decision Tree and are each assigned a class label. Both the root node and the decision nodes contain attribute tests that separate records with different characteristics. The separation is typically two-way thus forwarding the record to one of two child nodes. Here, records are possibly divided further if encountering another decision node. If encountering a leaf node, the record will be assigned the class label of the leaf node thus finalizing the prediction.

Creating a Decision Tree is often done through a greedy algorithm building the classifier through a series of choices one step at a time. One of the earliest algorithms is Hunt's algorithm. It functions by recursively dividing a set of data with mixed class labels into smaller more pure sets. It is defined by two steps:

1. If all records of a dataset D are of the same class, create a leaf node.
2. If D contains different class labels, create an attribute test dividing the data into smaller subsets. Recursively apply procedure to each subset.

When a dataset contains several class labels and needs to be split, the algorithm must decide which attribute to test on. This is done by calculating the impurity improvement of all possible splits. Several impurity measures exist including Gini index and Entropy. Each defines a method of calculating the impurity of child nodes during a split with the purpose of identifying the feature best dividing the class labels. The impurity for a node t is defined by Gini and Entropy as follows:

¹⁵ (Tan, Karpadne, Kumar, & Steinbach, 2006)

$$Gini(t) = 1 - \sum_{i=1}^C P_i^2$$

$$Entropy(t) = - \sum_{i=1}^C P_i \cdot \log_2(P_i)$$

Here C is the amount of different class labels within the node, and P_i is the fraction of records belonging to class i within the node. When performing a split, the impurity of each child node is determined using a measure like the ones above. Each child node's impurity is combined to form the impurity for the entire split. This is defined as the weighted average of the child node's impurities:

$$I_{split} = \sum_{i=1}^k \frac{n_i}{n} \cdot I(i)$$

Where k is the number of child nodes created with the split, $I(i)$ is the impurity of child node i , n is the number of data records in the parent node, and n_i is the number of data records in child node i . With the impurity of a split I_{split} , $Gain$ can be determined which is defined by the difference in impurity before and after the split:

$$Gain = I(parent) - I_{split}$$

All possible splits on a node can now be compared to each other. The split resulting in the largest $Gain$ is defined as the best possible split and chosen by the algorithm.

A simple example of the algorithm in action will now be provided using the Gini impurity measure. Table 3 contains a small sample of real data from the flight plans provided by ForeFlight. The features have been engineered into destination coordinates and encoded wake turbulence categories. Between the flight plans included in the table are two different destinations, one in Austin Texas USA (30.20, -97.67), and one in Copenhagen, Denmark (55.62, 12.65). Two of the flight plans are using a Boeing 737 which is categorized as having a medium wake turbulence giving it the value 1. The other flight plans use the smaller C172 which is categorized as small thus given the value 0.

ID	Destination Latitude	Destination Longitude	Wake cat.	Alternate Airport
0	30.20	-97.67	1	KGYB
1	55.62	12.65	1	ESMS
2	30.20	-97.67	0	KEDC
3	55.62	12.65	0	EKRK
4	30.20	-97.67	0	KGYB
5	30.20	-97.67	0	KGYB

Table 3 - Small example of flight plan data

Given this data, the decision tree needs to find a way to logically split the data to separate the class labels. It starts at the root node by determining a split is needed since the condition of step 1 of Hunt's algorithm is not met. To do so, it would have to calculate which of the features result in largest *Gain* when split upon. Before any split is performed, the impurity of the root node will be calculated. The node contains four different class labels between the six records. The impurity can be defined:

$$Gini_{parent} = 1 - \left(\left(\frac{3}{6} \right)^2 + \left(\frac{1}{6} \right)^2 + \left(\frac{1}{6} \right)^2 + \left(\frac{1}{6} \right)^2 \right) \approx 0.667$$

In this example, the algorithm starts by testing a split using the wake turbulence category. This would separate records based on whether their wake turbulence category is 0 or 1 thus generating two child nodes with four and two records respectively. The Gini index for the first node is determined. Three different class labels exist between the four records of the node:

$$Gini_0 = 1 - \left(\left(\frac{2}{4} \right)^2 + \left(\frac{1}{4} \right)^2 + \left(\frac{1}{4} \right)^2 \right) = 0.625$$

The Gini-index for the other child node is determined:

$$Gini_1 = 1 - \left(\left(\frac{1}{2} \right)^2 + \left(\frac{1}{2} \right)^2 \right) = 0.5$$

The impurity of the entire split is then calculated:

$$Gini_{split(wake)} = \frac{4}{6} \cdot 0.625 + \frac{2}{6} \cdot 0.5 \approx 0.583$$

Finally, the *Gain* from splitting on wake turbulence category:

$$Gain_{wake} = 0.667 - 0.583 = 0.084$$

The *Gain* achieved with the wake turbulence category has to be compared against the *Gain* achieved splitting on the other features. The longitude and latitude impurity calculations would result in the exact same values, so for the sake of keeping the example short, only the calculations for longitude have been included. The procedure is the exact same:

$$Gini_{-97.6} = 1 - \left(\left(\frac{3}{4} \right)^2 + \left(\frac{1}{4} \right)^2 \right) = 0.375$$

$$Gini_{12.6} = 1 - \left(\left(\frac{1}{2} \right)^2 + \left(\frac{1}{2} \right)^2 \right) = 0.5$$

Having the Gini score for each child node, it is possible to calculate the Gini split score:

$$Gini_{split} = \frac{4}{6} \cdot 0.375 + \frac{2}{6} \cdot 0.5 \approx 0.417$$

The *Gain* from splitting on longitude can now be determined:

$$Gain_{longitude} = 0.667 - 0.417 \approx 0.25$$

This concludes that the longitude is the better feature to choose for the split. When the split is made, the same process continues recursively, and the new nodes serve as parents since the condition of step 1 is still not met. The algorithm starts over by calculating the different features' split impurities and choosing the one resulting in the largest *Gain*. Performing the calculations would result in both new nodes being split using the wake turbulence category. The finished Decision Tree is illustrated by Figure 7:

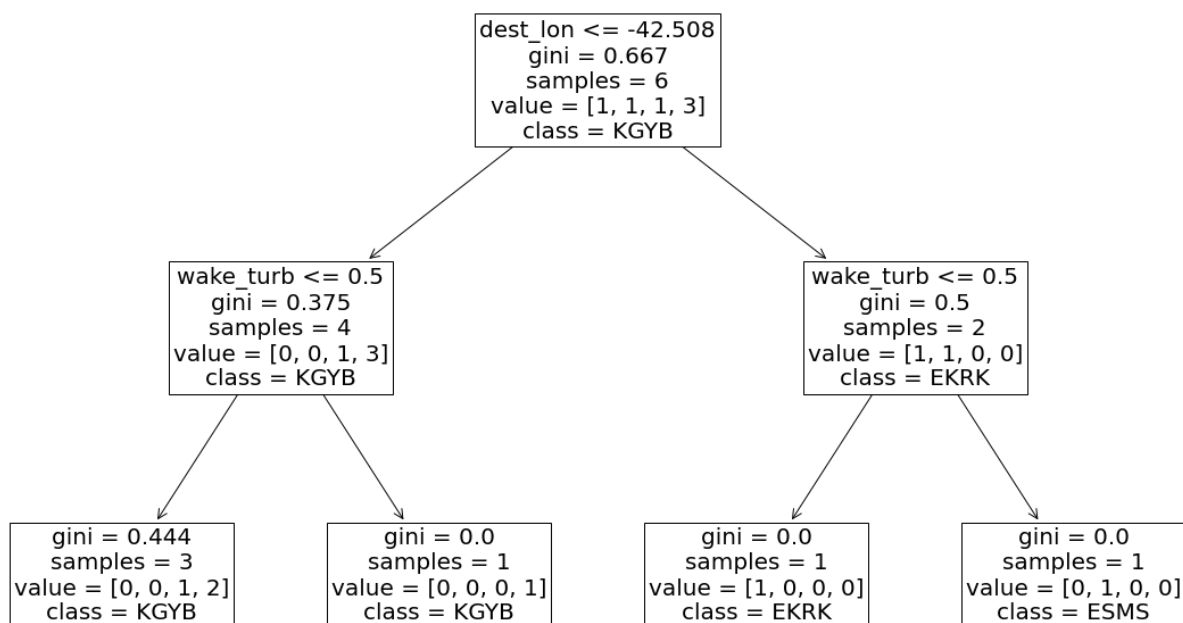


Figure 7 - Finished decision tree built from small data example

Within Figure 7 the Gini index for each node is shown, as well as the number of different records (samples) and class labels (values, sorted alphabetically). The resulting tree has three pure leaf nodes containing only one class label. However, one leaf node contains two different class labels resulting in a wrongly classified record. This record is flight plan number 2 from Table 3 where KEDC was selected as alternate. Since there exists more flight plans with the same features but with KGYB as alternate, the leaf node is assigned KGYB as class label. The resulting Decision Tree would therefore never predict KEDC as alternate.

Given the generated Decision Tree model, it is possible to try a prediction by using new data as input. Table 4 provides a previously unseen flight plan with an unknown class label.

ID	Destination Latitude	Destination Longitude	Wake cat.	Alternate Airport
6	29.65	-95.28	2	???

Table 4 - Example of unseen flight plan with unknown alternate

The new input represents a flight plan with destination of William P. Hobby Airport in Houston, Texas. The airplane is a Boeing 744 that with its 183.5 Tons is categorized as having a heavy wake turbulence represented as the number 2. When this data is inputted in the model the Decision tree will take the yellow path shown in Figure 8.

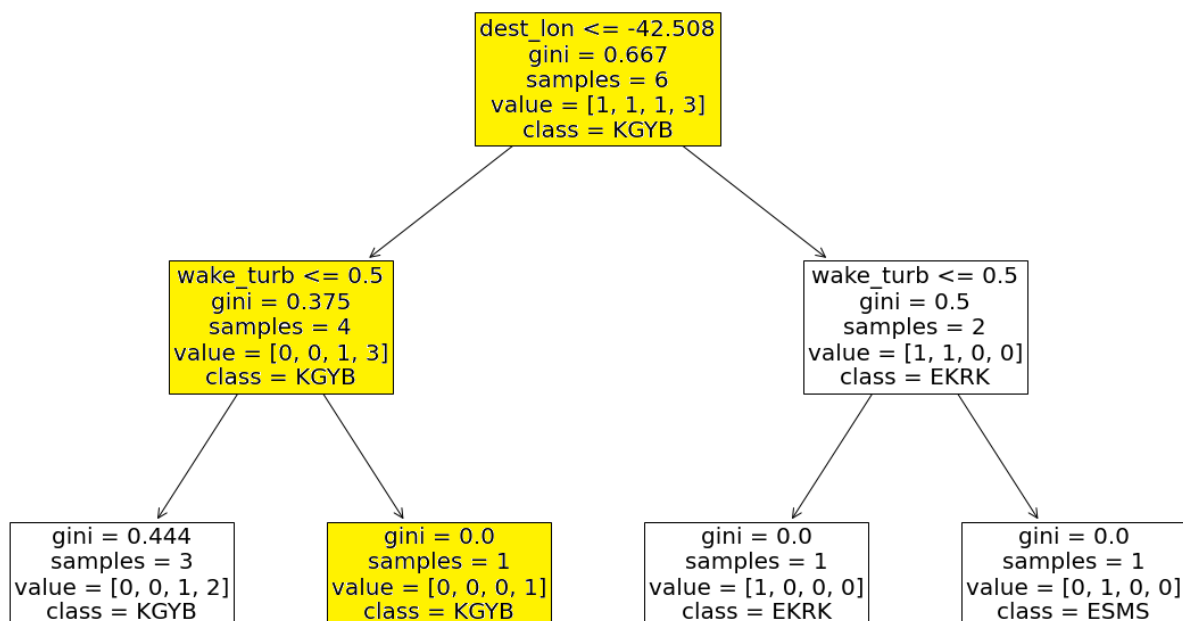


Figure 8 - Path taken by unseen flight plan from Table 4

This data has never been seen by the model before, but it is still able to predict on it because of the simple way that it splits the nodes. Starting at the root node which looks at the destination longitude parameter, and for this example the longitude value is -95.28 which is smaller than the -42.508 that the tree splits on. It then follows the left arrow and hits next node which tests the wake turbulence class of the aircraft. The attribute test results in the right path because 2 is larger than the 0.5 it splits on. The decisions ends up with a pure leaf node with the value KGYB. The predicted alternate for a flight plan with William P. Hobby Airport as destination and flying a Boeing 744 would thus be Giddings-Lee County Airport in Texas.

5.3.2 Random Forest

A Random Forest is an ensemble method which is a technique used in machine learning that takes advantage of several base models, like for example Decision Trees, in order to produce one optimal predictive model¹⁶. The idea is to utilize the majority vote of several classifiers to obtain an error rate lower than that of each individual classifier. This requires that the errors between the base classifiers are independent of each other meaning they do not all misclassify the same records. To ensure the base classifiers handle records differently, ensemble methods often take advantage of bootstrap aggregation also known as bagging. Bagging is a way of creating several subsamples from the original dataset and create a decision tree for each of the subsamples as illustrated in Figure 9. The aggregation part of the bootstrap aggregation is the algorithm that afterwards aggregates over the trees to form an efficient predictor.¹⁷

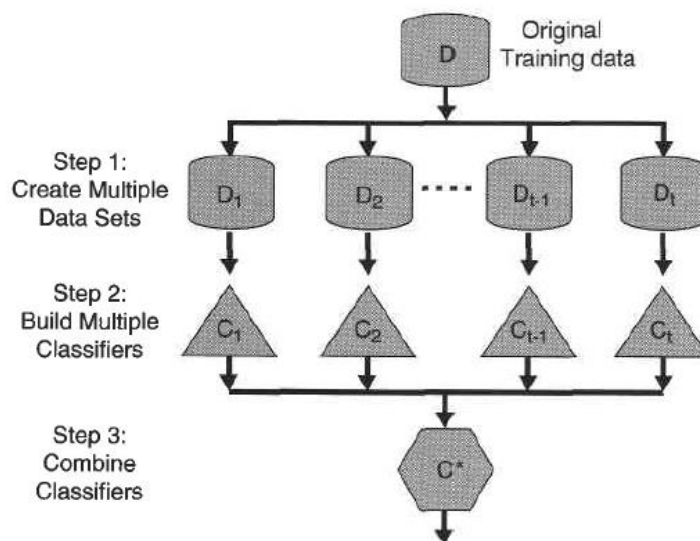


Figure 9 - A logical view of the ensemble learning method¹⁸

When the different decision trees inside the Random Forest model are created, the data used for training is done on a random sampling of the dataset. Since the sampling is done with replacement, it is possible for the same records to be present several times in the sampled dataset while some may not be included at all.

To further ensure the Decision Trees are trained to handle records differently, the Random Forest manipulates the input features used to train each Decision Tree. Normally when building a Decision Tree, all

¹⁶ (Tan, Karpadne, Kumar, & Steinbach, 2006)

¹⁷ (Lutins, 2017)

¹⁸ (Tan, Karpadne, Kumar, & Steinbach, 2006)

features of the dataset are considered when making a split. However, Random Forests only consider a fixed number of features randomly selected at each node split. The Decision Trees generated by the Random Forest are therefore not necessarily built with all features.

When new data is inputted in the model, a prediction is done for each of the decision trees inside the forest. The class label predicted by the majority of the Decision Trees becomes the final output of the Random Forest model.

An example using the same dataset as in section 5.3.1 regarding the Decision Tree will now be provided to show the different steps taken by the Random Forest classifier. Table 5 shows the input data:

ID	Destination Latitude	Destination Longitude	Wake cat.	Alternate Airport
0	30.20	-97.67	1	KGYB
1	55.62	12.65	1	ESMS
2	30.20	-97.67	0	KEDC
3	55.62	12.65	0	EKRK
4	30.20	-97.67	0	KGYB
5	30.20	-97.67	0	KGYB

Table 5 - Example flight plan data from Table 3

The first part of the process is to bootstrap the dataset by extracting a collection of random records and building a decision tree based on those. Bootstrapping the dataset could result in the following data sample. Notice how some records (0 and 2) have been selected multiple times while some (1 and 5) are not present at all.

ID	Destination Latitude	Destination Longitude	Wake cat.	Alternate Airport
3	55.62	12.65	0	EKRK
4	30.20	-97.67	0	KGYB
2	30.20	-97.67	0	KEDC
0	30.20	-97.67	1	KGYB
2	30.20	-97.67	0	KEDC
0	30.20	-97.67	1	KGYB

Table 6 - Bootstrapping subset generated from dataset in Table 5

A Decision Tree is then trained using the bootstrapped data. Instead of testing all feature splits, the Decision Tree only considers a certain number of randomly selected features. In the scikit-learn library, the number of features is determined by the hyperparameter *max_features*. Per default, *max_features* equals

$\sqrt{n_features}$ which in this case equals $\sqrt{3} \approx 1.73$. This number is rounded down to 1 by scikit-learn.¹⁹ Each split therefore only tests one randomly selected feature of the three (latitude, longitude, and wake turbulence category). The hyperparameter *max_features* can be manually increased resulting in the algorithm comparing the impurity improvements of more features.

After splitting a node, a new feature is selected at random for each of the child nodes to be split upon. In the event that a feature split does not provide any *Gain*, the algorithm allows itself to go above the *max_features* value and create a split based on another random feature. Understanding the choice of random features and the random sampling done in the bootstrapping, was key to understanding how the Random Forest creates each of the Decision Trees inside the forest.

Another parameter on the Random Forest that has not been mentioned yet is *n_estimators*. This tells the Random Forest how many base classifiers it should construct. The standard value in scikit-learn is 100, but to create an example that is not too overwhelming the number of estimators has been set to 5. Using scikit-learn, a Random Forest classifier with five estimators is trained using the data provided in Table 5. The resulting Decision Trees are shown in Figure 10.

¹⁹ (Github, 2021)

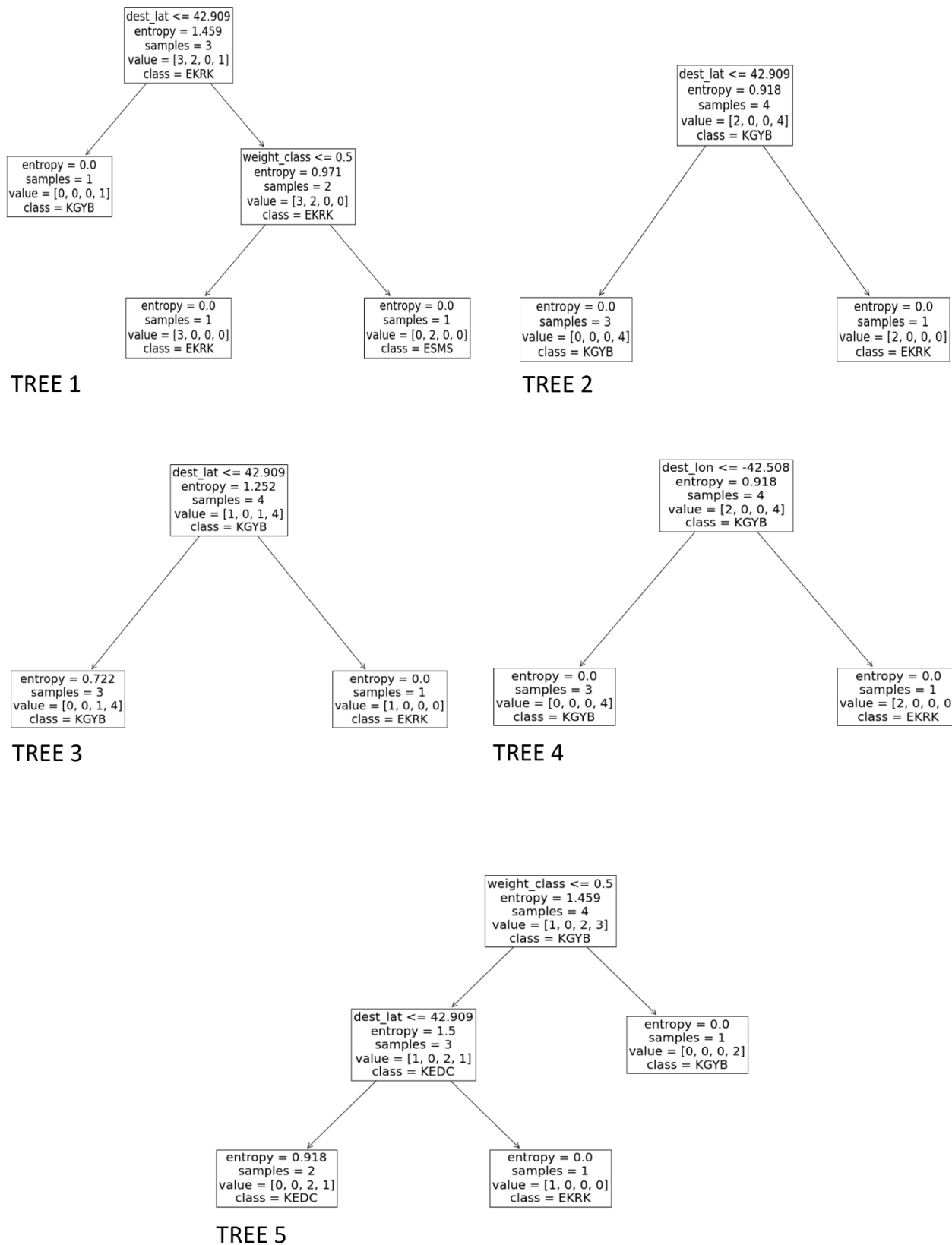


Figure 10 - Base classifiers created by Random Forest using data from Table 5

Looking at the five Decision Trees, they all split the nodes based on different features. This is both due to them being trained on different bootstrapped data samples and randomly selecting one feature when splitting nodes. This randomness results in many different variations of Decision Trees hopefully helping the classifier determine the correct prediction.

The dataset provided to tree number five fits the bootstrap example from Table 6. Looking at the root node's values, it has one record with class label EKRK, zero with ESMS, two records KEDC, and three records with KGYB. Knowing the complete dataset, the splits made in the tree are not optimal. It starts by splitting the root node based on the wake turbulence category resulting in a pure leaf node to the right. This is not a good decision since the location ideally should be taken into account before reaching a conclusion. This just shows that bootstrapping can lead to data samples not representative of the real world. This is the nature of the Random Forest and is only a problem if it was a single Decision Tree and not a combination of many. The different trees and the aggregation in a Random Forest should make up for a better prediction.

Now that the Random Forest classifier has been trained, it is possible to expose it to new flight plans to generate a suggestion for an alternate. The following flight plan has Billund airport in Denmark as destination and uses an aircraft categorized as having a large wake turbulence category.

ID	Destination Latitude	Destination Longitude	Wake cat.	Alternate Airport
6	55.74	9.15	2	???

Table 7 - Example of unseen flight plan with unknown alternate

Each of the Decision Trees are given the flight plan to generate a prediction. Afterwards, the aggregation can conclude on which label best serves as output. The Decision Trees would respectively predict the alternates ESMS, EKRK, EKRK, EKRK, and KGYB. For this flight plan, EKRK has been predicted by three of the five base classifiers and therefore becomes the final prediction of the Random Forest. It is important to mention that this is a very small data sample, and the number of estimators is very few in order to make the example easily understandable. ESMS would be a better result due to its size, but the Random Forest and the five Decision Trees were biased towards EKRK since the random sampling algorithm chose samples with EKRK more often than EKMS. The probability of this happening will diminish when creating Random Forests from larger datasets and using more base classifiers.

5.3.3 Multilayer perceptron

As an alternative to the Decision Tree and Random Forest the Multilayer Perceptron has been used. MLP is a type of feedforward artificial neural network consisting of several building blocks, like neurons, weights, and activation functions. It is an effective classifier because of its abilities to learn arbitrary functions but typically expensive to train²⁰. The multilayer perceptron used for this project is the MLP model from the scikit-learn python library.

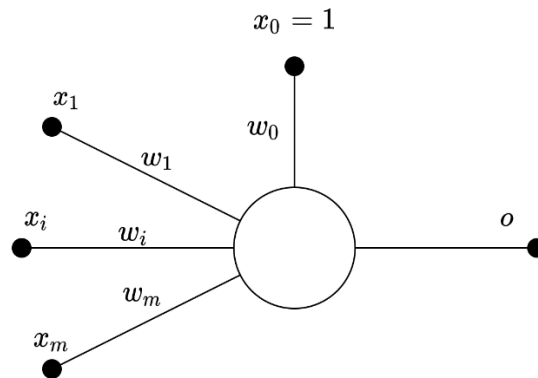


Figure 11 - Simple perceptron²¹

The concept of artificial neural networks was inspired by the billions of biological neurons connected inside a brain. In the multilayer perceptron model, a single perceptron is modelled as having several inputs x each with an assigned weight w and a single output o as seen in Figure 11. Inside the perceptron, all weighted inputs are added together and passed through an activation function f determining the output as indicated by the following formula.

$$o = f\left(\sum_{i=0}^m x_i w_i\right)$$

The activation function f can be of many types. The simplest is the step function returning either -1 or 1 when given values respectively below and above 0 . The activation function input is determined by the weighted sums of inputs x . Input x_0 with a constant value of 1 is called the bias whose weight is adjustable to determine what is often called the threshold. With a weight $w_0 = -1$, the rest of the weighted inputs

²⁰ (Heaton, 2017)

²¹ Figure inspired by (Tan, Karpatne, Kumar, & Steinbach, 2006)

must add up to at least 1 to give a positive output for the activation function. However, other than its constant value of 1, the bias is treated exactly as an ordinary input.

The simple perceptron outputs a binary value depending on the provided input and can function as a simple binary classifier. The perceptron can be trained on a set of data to adjust its weights accordingly to produce the correct results. For each piece of training data, compute the output o , and update each of the inputs' weights using the following formula:

$$\Delta w_i = \eta \cdot (d - o) \cdot x_i$$

Where η is a constant determining the learning rate, and d is the desired output for this particular piece of training data. A full iteration over the dataset is called an epoch. The algorithm usually runs as many epochs as it takes to eliminate any prediction errors, or it reaches a defined limit or tolerance.

A classifier made by a simple perceptron serves as a linear separator of classes. Figure 12 depicts a problem of separating two classes using three inputs. The decision boundary from the simple perceptron classifier linearly separates the classes.

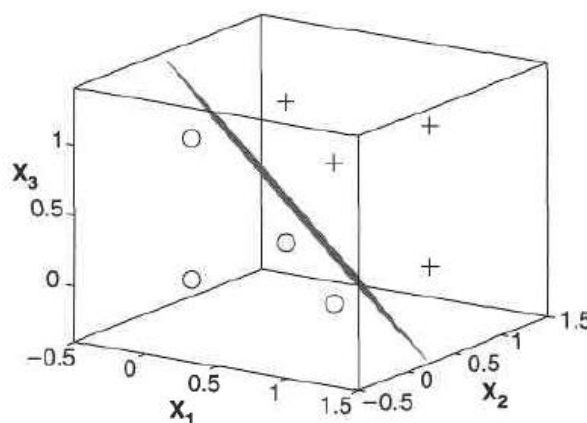


Figure 12 - Illustration of decision boundary created by simple perceptron²²

Not all problems are simple enough to be linearly separable. It is possible to make a classifier that can predict arbitrary functions by combining multiple perceptrons into a network called a Multilayer Perceptron where the output of some perceptrons serves as input for others. The multilayer perceptron can be divided into 3 types of layers: the visible input layer, the hidden layer, and the output layer. The input layer consists of the features from the dataset. The data that is used as input must be the same as later used to predict, so if the

²² (Tan, Karpapne, Kumar, & Steinbach, 2006)

data has been normalized before the model is trained, then the data inputted for prediction would also have to be normalized. After the input layer comes the hidden layers whose shapes and sizes differ from model to model. The last layer is the output layer which has the same size as the number of possible outputs. In this case, where there are over 4000 different alternates, the output layer will have over 4000 end nodes. The output can be presented in different ways, either as a binary value which gives a true or false, or as in the case of this project, an approximate probability for a given output node to be true.

As with the simple perceptron, the MLP is trained using an error function. A commonly used function is the mean squared error as seen below. n is the number of entries in the training data, d_k is the desired output for data entry k , and o_k is the network's actual output for data entry k .

$$E(w) = \frac{1}{2n} \sum_{k=0}^n (d_k - o_k)^2$$

The error is a function of the network's weights w since these directly influence the output. The goal is to determine the weight values that result in the lowest error. This can be done with gradient decent. The gradient decent method requires that the error function is differentiable in order to determine the gradient at certain weight values. The step function used by the simple perceptron should therefore be changed. A commonly used activation function is the sigmoid function:

$$f(x) = \frac{1}{1 + e^{-x}}$$

Figure 13 illustrates the error surface over two weight parameters. Using gradient decent, each weight is changed by the following amount until it converges. How quickly and if the algorithm converges depends on the learning rate and the complexity of the problem. There might be several local minima where the network gets stuck.

$$\Delta w_i = -\eta \cdot \frac{\partial E}{\partial w_i}$$

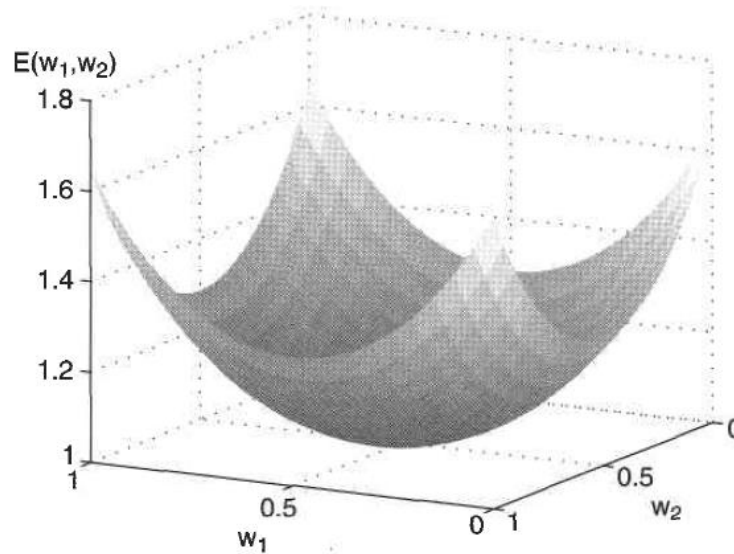


Figure 13 - Error surface over two weight parameters²³

When dealing with a multilayer perceptron, it is difficult to determine the effect the weights in the hidden layers have on the error. The desired output of the nodes in the hidden layers are not known unlike the desired outputs of the output nodes. Since the output of a layer is directly affected by the output of the previous layer, it is possible to estimate the errors for nodes using the errors from the next layer. This technique is called back propagation and consists of two phases: The feed-forward phase and backwards phase. During the feed-forward phase, a set of inputs are sent through the network to generate outputs for both the hidden and output nodes. The backwards phase then starts at the output layer by calculating δ for each node, which tells how much the weight of a particular node should be changed. When using the sigmoid activation function, δ is determined by:

$$\delta_i = o_i \cdot (1 - o_i) \cdot (d_i - o_i)$$

Where d_i is the desired output of the node, and o_i is the observed output. The δ -value for each hidden node can now be determined:

$$\delta_j = o_j \cdot (1 - o_j) \cdot \sum_i w_{ji} \delta_i$$

²³ (Tan, Karpatne, Kumar, & Steinbach, 2006)

Where o_j is the output of the hidden node, and w_{ji} is the current weight between the hidden node j and node i in the following layer. This value is calculated for each node in the entire network by propagating backwards. Afterwards it is possible to update the weights between every node i and j :

$$w_{ij} \leftarrow w_{ij} + \Delta w_{ij} \quad \text{where} \quad \Delta w_{ij} = \eta \cdot \delta_j \cdot x_{ij}$$

Here, x_{ij} is the input of node j which is also the output of node i .

The backpropagation algorithm runs several epochs until the error reaches an acceptably low value. It is often very time consuming to train an artificial neural network depending on the amount of data and size of hidden layers. It is not a good idea to include more neurons than necessary since it will increase the training time and the risk of overfitting²⁴. The number of layers should also be kept at a reasonable level. Networks only need three layers to be capable of representing any function.

Another method of improving training expense is to use stochastic gradient decent²⁵. Instead of calculating the gradient using all pieces of training data at once, it runs the feed-forward and backpropagation on one randomly selected input at a time thus making more frequent updates to the weights. One drawback, however, is that each gradient risks not representing the true optimal gradient. This is often counteracted by sampling batches of data instead of just one at a time.

²⁴ (Heaton, 2017)

²⁵ (Srinivasan, 2019)

5.4 Method of evaluation

The following section covers the method of evaluating the models. The evaluation is based on both an accuracy measurement and a visual assessment.

5.4.1 Accuracy measurement

To evaluate the performance of the models, the data is divided into a training set and a test set. The training and test set each consists of respectively 80% and 20% of the total data. Splitting the data into two partitions ensures that the model is tested on different data that it has been trained on to achieve an unbiased validation of its performance. Testing the model on the same data it has been trained with does not prove that the model works for the general case. A model that achieves high accuracy on training data, but low accuracy on test data indicates that the model is overfitted to the training data.

When the model receives a flight plan and predicts an alternate, this predicted alternate is compared to the actual choice from the flight plan. The prediction can either be right or wrong, and the entire model's accuracy is based on how many of its total predictions are correct²⁶.

$$Accuracy = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

This method of evaluating machine learning models is widely accepted for classification systems. However, it is arguable whether this method is appropriate in the context of this project. This traditional evaluation method implies that for each set of inputs, there exists only one correct output. This is not the case when it comes to selecting alternates. For one flight plan, there might be several acceptable alternates the pilot can choose from. Each of the provided flight plans only represent one possible choice of alternate, and it is unlikely that this choice is the only correct one. There may even have been better alternates in some cases. It is important to keep in mind that the alternates have been chosen by pilots, and they might not have chosen perfectly.

It has therefore been necessary to come up with an evaluation method better suited for the context of this project. As long as the actual alternate from the flight plan appears in the top ten most likely alternates according to the model, the prediction can be considered successful. In the flight plan software, when pilots fill out the flight information, they will be shown a list of alternates suggested by the machine learning model

²⁶ (Tan, Karpatne, Kumar, & Steinbach, 2006)

to choose from. Therefore, pilots will be able to see and choose the alternate even though it is not the most optimal one according to the machine learning model.

5.4.2 Visual evaluation

In addition to the accuracy measurement a graphical user interface has been created to visually evaluate the trained models. The GUI represents the system as it would be used by a pilot creating a flight plan. The purpose of the GUI has been to visually represent the output of each model making it easier to detect any invalid suggestions. This evaluation method also provides an opportunity for the pilot to evaluate the models.

The presented GUI consists of three parts. The left part is used to input the necessary data, and is provided two fields, one for the destination ICAO, and one for the aircraft ICAO. These values are then sent to the server which has an input pipeline to transform the values into correct data for the model to predict on. The server then returns the prediction together with information about the predicted alternates. The bottom part of the left column is an option for the pilot to select an alternate of his choice and see information regarding that specific alternate.

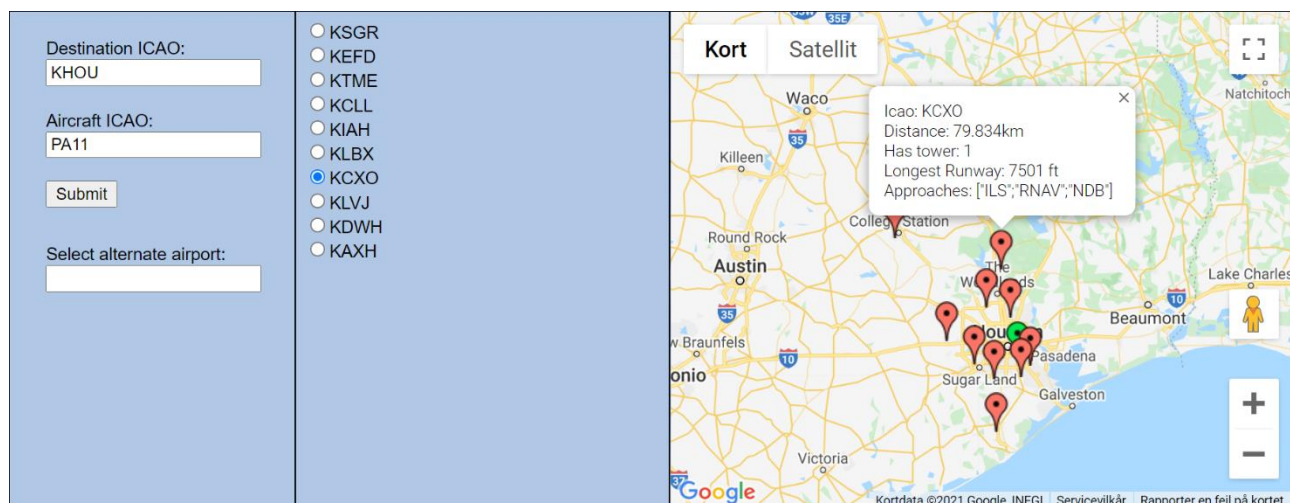


Figure 14 - Screenshot from created graphical user interface

As shown in Figure 14, the middle column is a list of the predicted alternates that is shown in the same order as the probabilities outputted by the machine learning model. An alternate selected from the list will be shown on the map in the right column of the GUI. The map is an integration of the Google Map API with red markers added for each of the alternates suggested by the model, and a green marker for the destination. When an airport is selected on either the map or the list, information is shown about its ICAO, distance to the destination, if it has a tower, the longest runway, and what type of approaches it offers.

The GUI allows for evaluating and comparing models' output visually. Since the standard and top-ten testing methods only evaluates on one item in the list, there is a need for validating the quality of the remaining suggested alternates.

5.5 Technical implementation

Pandas

For this project, the open-source library called Pandas was used to handle the dataset into manageable dataframes. Pandas offers a fast and efficient way of creating dataframe objects together with tools for loading, reshaping and much more.²⁷

Scikit-learn

The machine learning done in this project uses the scikit-learn library. This library is very useful since it works well with Pandas and has a lot of different "off-the-shelf" machine learning algorithms. The documentation for scikit-learn is easily understandable, and the scikit-learn github offers a view into the code to gain an understanding of the inner workings behind the tools provided. Scikit-learn offers tools for classification, regression, clustering, and dimensionality reduction.²⁸

²⁷ (tutorialspoint.com, 2021)

²⁸ (analyticsvidhya.com, 2015)

6 Evaluation

This section will go through each of the presented classification algorithms and describe how they performed. For each of the classification algorithms, the different pre-processing and input feature selection/engineering techniques has been tested to later compare them. As described in section 5.2.1 regarding feature selection, it is known that the destination airport and aircraft type are two important factors in the alternate decision process. Therefore, the evaluation will start by looking at different combinations and alterations of these. Afterwards, features with more uncertain outcomes will be tested on the configuration of destination airport and aircraft type proven most successful.

To further optimize the performance of the models, they will be tested with different variations of relevant hyperparameters. The accuracy tests conducted on the features beforehand will have the default hyperparameter configuration unless otherwise specified.

Both the traditional and the custom test described in section 5.4.1 will be used to assess the models. Generally, the accuracies from the two test methods follow each other proportionally, but in some cases during the experimentation one would decrease while the other increased. When evaluating the different variations of models, their size will be kept in mind. This is because it might be relevant for pilots to use them on phones and/or tablets which have limited hardware capabilities. Training times are not considered important in this use case and will not be considered in the evaluation.

At the end of this section, a qualitative evaluation is covered by getting the pilot interviewed earlier to use the finalized product with the purpose of evaluating the model from a professional point of view. Any accuracy measure and performance test are useless if the model is not usable in the eyes of a real pilot.

6.1 Experimental evaluation

6.1.1 Decision tree

According to Table 8, the best combination of the input features representing the destination and aircraft depends on which test the model is measured against. For the standard test, the model peaks at an accuracy of 49,29% when using the destination airport ICAO code and aircraft code as input. However, the configuration achieving the highest score with the top-ten test is when using the destination coordinates and the aircraft wake turbulence category. Generally, going from Aircraft code to Aircraft wake turbulence category results in a lower score from the standard accuracy test but a higher score in the top-ten test.

Input features	Test accuracy (standard)	Test accuracy (in top ten)
- Destination airport code - Aircraft code	49.29%	78.13%
- Destination airport latitude - Destination airport longitude - Aircraft code	49.20%	77.87%
- Destination airport square - Aircraft code	41.55%	78.45%
- Destination airport code - Aircraft wake turbulence category	45.23%	90.01%
- Destination airport latitude - Destination airport longitude - Aircraft wake turbulence category	45.18%	90.70%
- Destination airport square - Aircraft code wake turbulence category	34.41%	88.49%

Table 8 - Results from testing Decision Tree with different representations of destination and aircraft

The following tests have been conducted using the input features from before resulting in the most accurate Decision Tree (destination coordinates + aircraft wake turbulence category). None of the additional features improve the accuracy of the decision tree. Therefore, none of them will be included in the rest of the tests.

Additional input feature	Test accuracy (standard)	Test accuracy (in top ten)
Destination has precision approach (true/false)	45.11%	90.46%
Destination longest runway (n ft)	44.90%	90.21%
Destination weather (VFR, Marginal VFR, Low IFR, IFR)	44.06%	87.34%
Flight rule (IFR/VFR)	45.14%	89.97%

Table 9 - Results from testing Decision Tree with additional features

The hyper parameter *max_depth* was tested to explore what max depth was the optimal for the Decision Tree. Figure 15 shows that every *max_depth* value under 23 has a lower accuracy than the values above, and from the graph it can be concluded that there is no increase in performance when going over the 23 max depth value. Going over the diminishing return gives a risk of getting a large model size which can be problematic for smaller devices like if the model were to be used on smartphones.

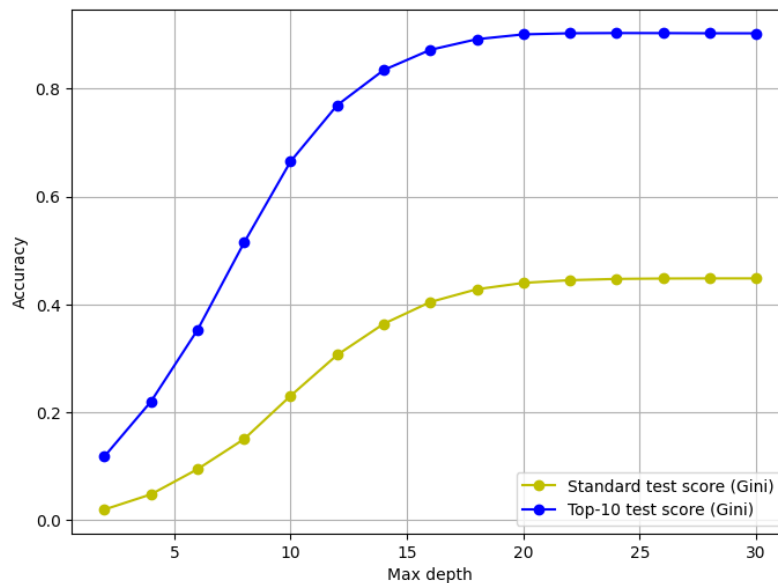


Figure 15 - Results from testing Decision Tree with various max depths

The impurity measure used by the Decision Tree can be changed through the hyperparameter *criterion*. By default, it uses the Gini impurity measure. However, it is worth testing the model using the Entropy impurity measure also described in section 5.3.1. When using default values for the other hyperparameters, the two different impurity measures do not seem to make a difference in accuracy. They both achieve accuracies of around 45% and 90% on respectively the standard test and the top-ten test. However, when visualizing the accuracies over different *max-depth* values, a difference starts to appear (see Figure 16). The model trained using Entropy impurity measure converges earlier than when trained using Gini. By using Entropy, it is possible to get a similarly accurate model with lower depth and therefore smaller in size. The model trained with Entropy peaks at 90.48% with a max depth of 12.

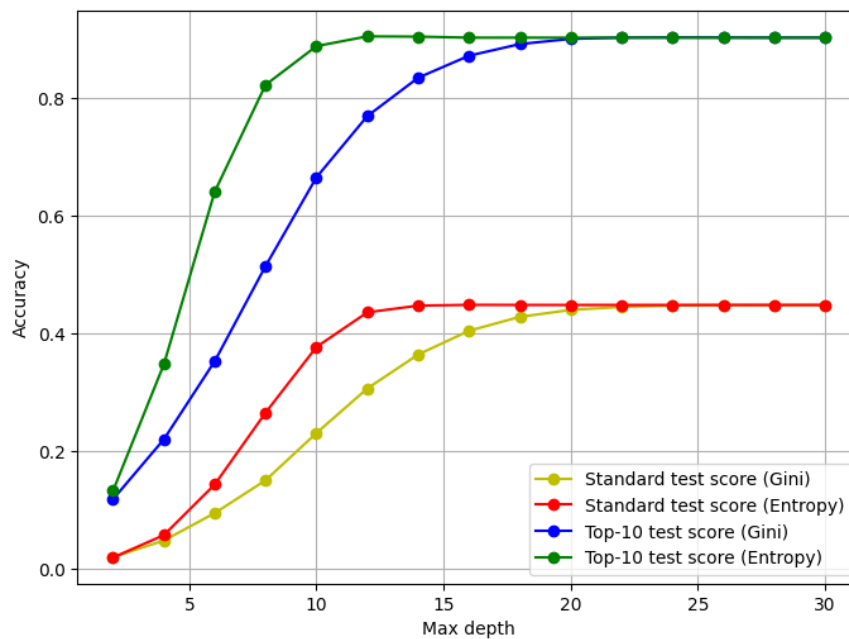


Figure 16 - Decision Tree impurity measures Gini and Entropy compared over various max depths

When training the Decision Tree with the best performing feature and hyperparameter configurations discussed in this section, it achieves an accuracy of 44.38% when using the standard test method and 93.16% when testing using the top-ten test method.

6.1.2 Random Forest

As discussed in section 5.3.2, the Random Forest classifier uses the combined results of a collection of Decision Tree classifiers to come up with a prediction. This means that the Random Forest consumes a massive amount of RAM and spends a long time predicting each test-input. To reduce the time and resources consumed, the following tests have been performed with only 20 estimators compared to the standard 100, and with a max depth of 11 compared to the unrestricted standard. The accuracies achieved will therefore not represent the best performing variant of the random forest model but only serve to compare different configurations of features and hyperparameters. Afterwards, each of the results will be combined to form the best possible Random Forest model.

Table 10 shows mostly the same patterns as for the Decision Tree classifier. The best performance was again achieved when trained on the destination coordinates and the aircraft wake turbulence category. Unlike the Decision Tree, this combination of input features achieves the highest score in both the standard test and the top-ten test.

Input features	Test accuracy (standard)	Test accuracy (in top ten)
- Destination airport code - Aircraft code	20.62%	51.35%
- Destination airport latitude - Destination airport longitude - Aircraft code	40.07%	86.16%
- Destination airport square - Aircraft code	28.77%	73.72%
- Destination airport code - Aircraft wake turbulence category	30.29%	74.47%
- Destination airport latitude - Destination airport longitude - Aircraft wake turbulence category	43.15%	89.63%
- Destination airport square - Aircraft code wake turbulence category	32.35%	86.49%

Table 10 - Results from testing Random Forest with different representations of destination and aircraft

The following tests have been conducted using the input features from before resulting in the most accurate Random Forest (destination coordinates + aircraft wake turbulence category). These tests show that the model becomes slightly more accurate when adding some of the additional features. However, upon further examination, this improvement should be taken with a grain of salt. When inputting different flight plans to the finished models, one starts to notice that it mostly ignores the wake turbulence category. This means it suggests several small airports to even the heaviest aircrafts. One reason the test accuracy has increased could be that the data only contain very few flight plans with heavy wake turbulence categories. Therefore, these will have low impact compared to the rest of the flight plans. Because of this drawback, none of the additional features will be included in the final model.

Additional input feature	Test accuracy (standard)	Test accuracy (in top ten)
Destination has precision approach (true/false)	43.28%	89.98%
Destination longest runway (n ft)	43.89%	90.69%
Destination weather (VFR, Marginal VFR, Low IFR, IFR)	42.54%	89.62%
Flight rule (IFR/VFR)	42.73%	90.17%

Table 11 - Results from testing Random Forest with additional features

Section 5.3.1 regarding theory behind Decision trees mentions Gini and Entropy as two ways of measuring impurity. The functions measure the impurity/quality of a split to determine the best feature to split on. In the comparison shown by Figure 17, the model trained using the Entropy impurity measure climbs more quickly and reaches its highest accuracy faster than the Gini model, when trained over different max_depths. Using Entropy, the model starts seeing diminishing returns at around the max depth of 11. Restricting the depth of the model ensures it does not take up more space than necessary. Furthermore, the graph shows the two models eventually reaches the same accuracy. However, the Entropy model peaks a bit higher but lowers towards the end of max-depth values. This is an indication of overfitting on the training when using deeper trees.

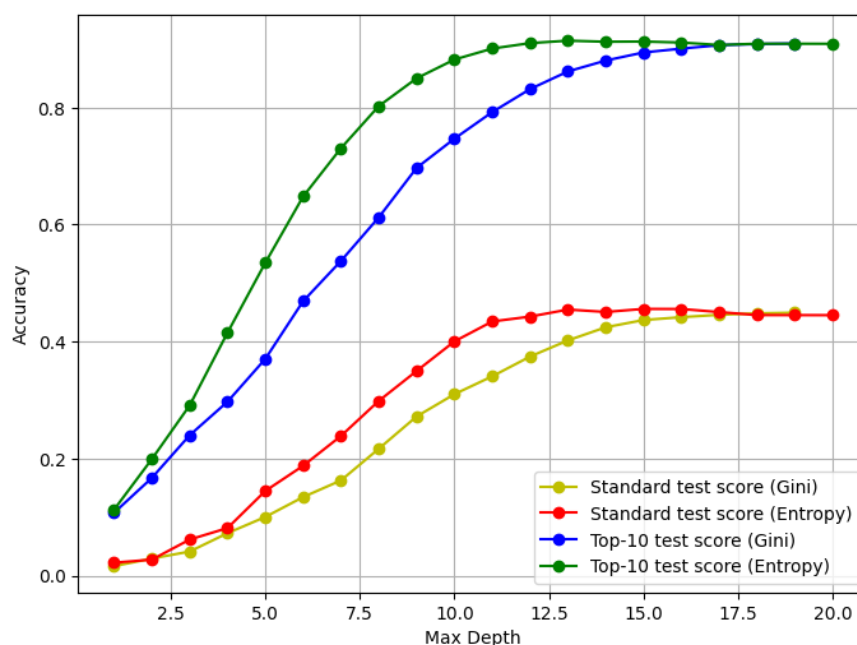


Figure 17 - Random Forest impurity measures Gini and Entropy compared over various max depths

Another meaningful hyperparameter is *max_features*. As described in section 5.3.2, this controls the amount of features considered with each decision tree split. As default, this hyperparameter equals $\sqrt{n_features}$ which in the case of the model trained on destination latitude, longitude, and wake turbulence category is $\sqrt{3}$ which is rounded down to 1. Figure 18 shows the accuracies when trained with *max_features* of 1, 2 and 3. Even though small, the model experiences an improvement with *max_features*=2.

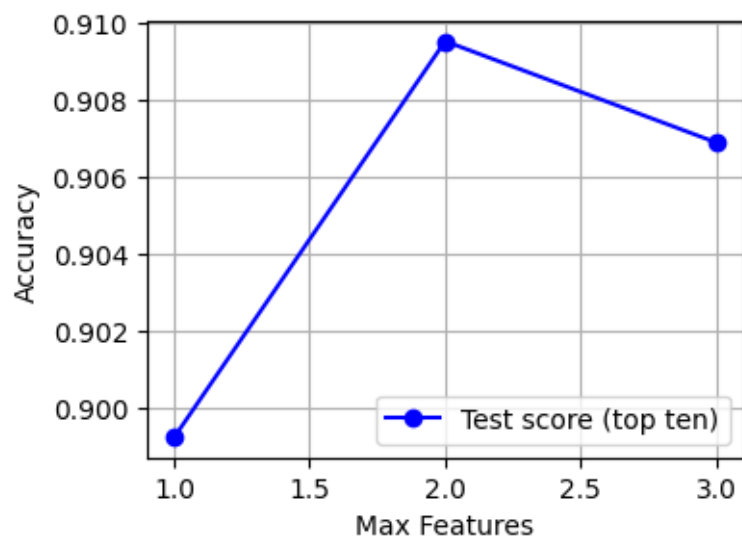


Figure 18 - Accuracy of Random Forest over various max features

Figure 19 shows test results for different values of $n_estimators$. As described in section 5.3.2, this parameter decides the number of decision trees trained to make up the random forest. The accuracy increases with the number of estimators but stabilizes around 40-ish. The amount of memory consumed by the random forest increases linearly with the number of estimators. Being able to train the model using only 40 estimators compared to the standard of 100, the size of the model is drastically lowered.

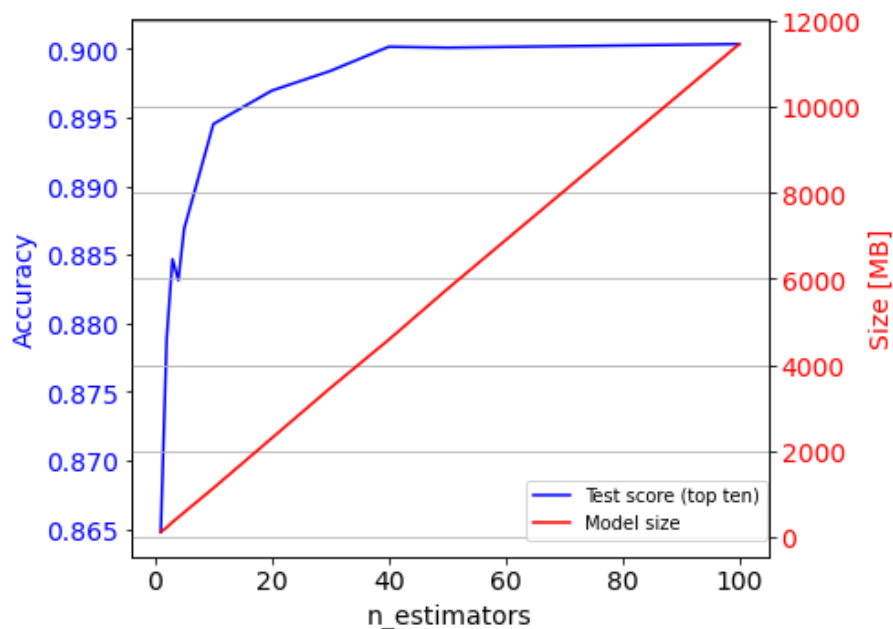


Figure 19 - Accuracy (blue) and size (red) of Random Forest over various $n_estimators$

When training the random forest with the best performing feature and hyperparameter configurations discussed in this section, it achieves an accuracy of 45.92% when using the standard test method and 94.13% when testing using the top-ten test method.

6.1.3 Multi-Layer Perceptron

The creation of hidden layers and complex connections between neurons, together with calculation and adjusting of weight and bias makes the multi-layer perceptron a computational heavy model to create. The test done in this section is done with the following hyper parameters:

- Solver: Adam
- Hidden layer size: (100,100,100,)
- Alpha: 0.0001
- Learning initiation rate: 0.005
- Activation function: relu
- Max iterations: 20

These parameters have been adjusted from earlier testing with this model and do not reflect the best accuracy that the model can perform if given unlimited time, but to make a comparison between different features it works well enough.

Table 12 shows that just like the Random Forest and Decision Tree, the combination of latitude, longitude and aircraft wake turbulence category provides the best accuracy for the features that the pilot stated were relevant. All other feature combinations generally show low accuracies compared to the aforementioned combination.

Input features	Test accuracy (standard)	Test accuracy (in top ten)
- Destination airport code - Aircraft code	13.41%	44.10%
- Destination airport latitude - Destination airport longitude - Aircraft code	22.57%	77.90%
- Destination airport square - Aircraft code	11.91%	50.49%
- Destination airport code - Aircraft wake turbulence category	13.37%	46.55%
- Destination airport latitude - Destination airport longitude - Aircraft wake turbulence category	25.28%	80.72%
- Destination airport square - Aircraft code wake turbulence category	10.21%	48.84%

Table 12 - Results from testing MLP with different representations of destination and aircraft

Table 13 shows that there is a small percentage increase from adding the longest runway to the feature list. When testing the model using the graphical user interface, it was revealed to be more likely to suggest alternates too far away from the destination. This was especially apparent when exposing the model to heavier aircrafts. The features therefore were kept at destination coordinates and wake turbulence category.

Additional input feature	Test accuracy (standard)	Test accuracy (in top ten)
Destination has precision approach (true/false)	23.69%	76.78%
Destination longest runway (n ft)	35.78%	85.75%
Destination weather (VFR, Marginal VFR, Low IFR, IFR)	21.26%	78.89%
Flight rule (IFR/VFR)	24.21%	79.35%

Table 13 - Results from testing MLP with additional features

Several hyper parameters were tested to find the best combination for the MLP model. For the first test, the different activation functions were tested to evaluate how each affected the accuracy. The result illustrated in Figure 20 shows that the relu function performs 10% better than the identity, logistic and tanh functions.

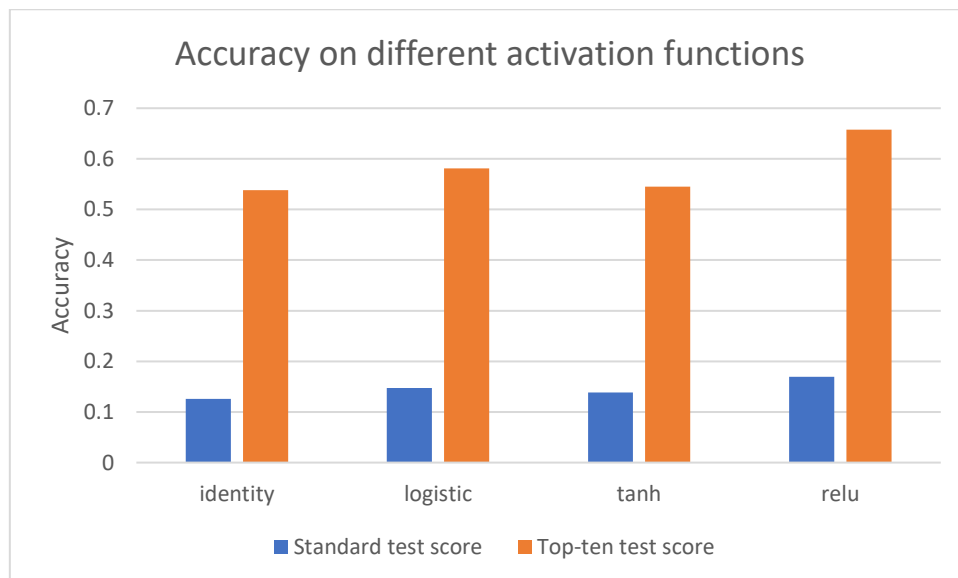


Figure 20 - Accuracy of MLP using various activation functions

The standard batch size is 200 unless the number of samples is lower. An evaluation was done on different batch sizes to find the relation between accuracy and batch size. Figure 21 below shows that a smaller batch size gave a higher accuracy, with 2% difference between a batch size of 100 and 200. The downside of using a smaller batch size, was the increase in training time.

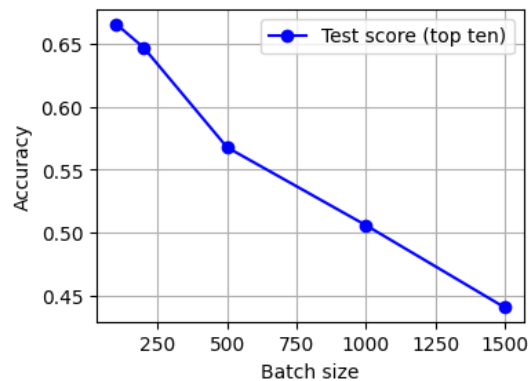


Figure 21 - Accuracy of MLP over various batch sizes

Since the MLP is a neural network, it was essential to analyze the relationship between different layers' structures and accuracy. In Figure 22 below it is shown that a model trained with a single layer had a lower accuracy than the ones with multiple hidden layers. It is also shown that the number of hidden layers have a higher impact on accuracy than the size of each of the hidden layers. The combination of hidden layers resulting in the most accurate model was with four layers each with 100 nodes. Compared to having three layers of 100 nodes, it took longer to train but only increased the accuracy marginally. The configuration with three layers was therefore chosen.

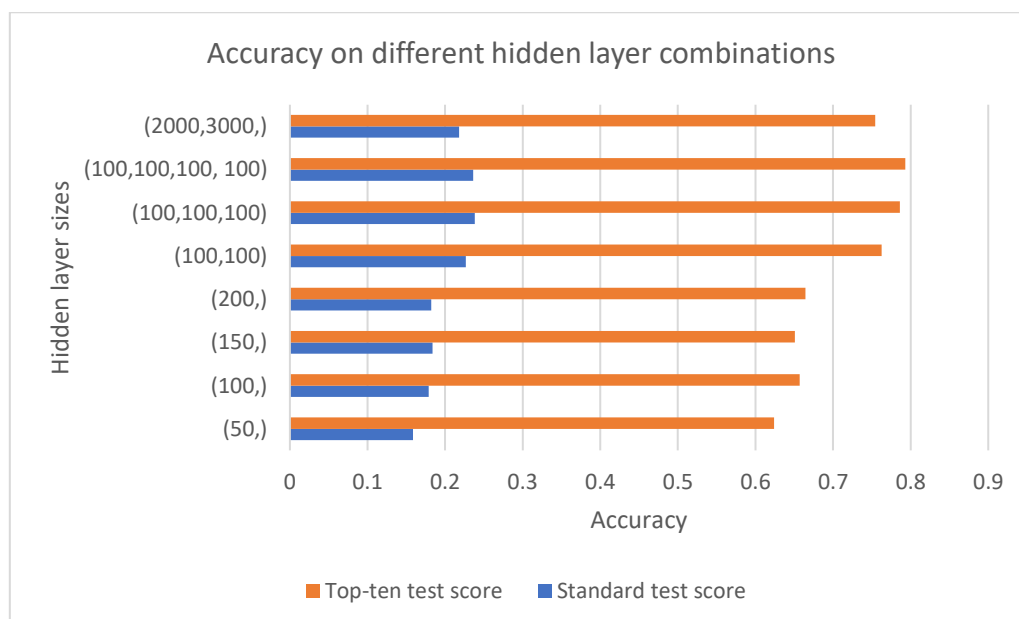


Figure 22 - Accuracy of MLP with various combinations of hidden layers

The MLP trained with latitude, longitude, and aircraft wake category as features was the best overall model. The model was tested using various normalization techniques including Standard Scaler and Min Max Scaler. Using normalization did increase the accuracy slightly but showed problems in specific use cases by suggesting invalid alternates placed in different continents. The final model achieves a score of 26.75% in the standard test and a score of 83.56% in the top-ten test.

6.2 Qualitative evaluation

Using the graphical user interface described in section 5.4.2, each classifier was compared against each other. Afterwards, the pilot tested the model to evaluate the quality of the predicted alternates.

6.2.1 Visual model comparison

Section 6.1 compared the accuracies on different models with different features. This served as a guideline for the theoretical best model. To find the best practical model for the pilots, the three different classifiers were given the same input, and then evaluated on the quality of the output.

Decision tree

As seen in Figure 23, generally the suggested alternates are close to the destination and are all valid for the Boeing 737 aircraft. Also shown in the suggestions are the KATW airport that is over 600 km North of the destination and is therefore invalidated. Alternates with large distances to the destination were a recurring theme of the Decision Tree classifier.

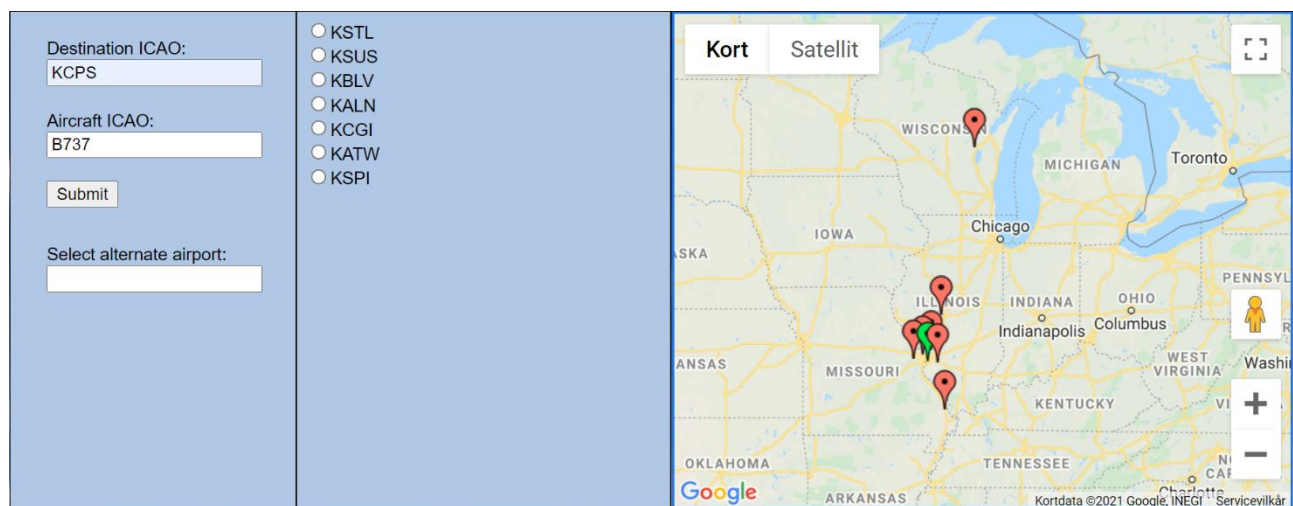


Figure 23 - Final Decision Tree tested in graphical user interface

Random Forest

The Random Forest classifier yields results similar to those of the Decision Tree. The generated suggestions are generally near the destination, but it will occasionally suggest a few alternates too far away to be considered valid. Furthermore, the random forest does not adjust its suggestions very much to fit the wake turbulence category of the flight.

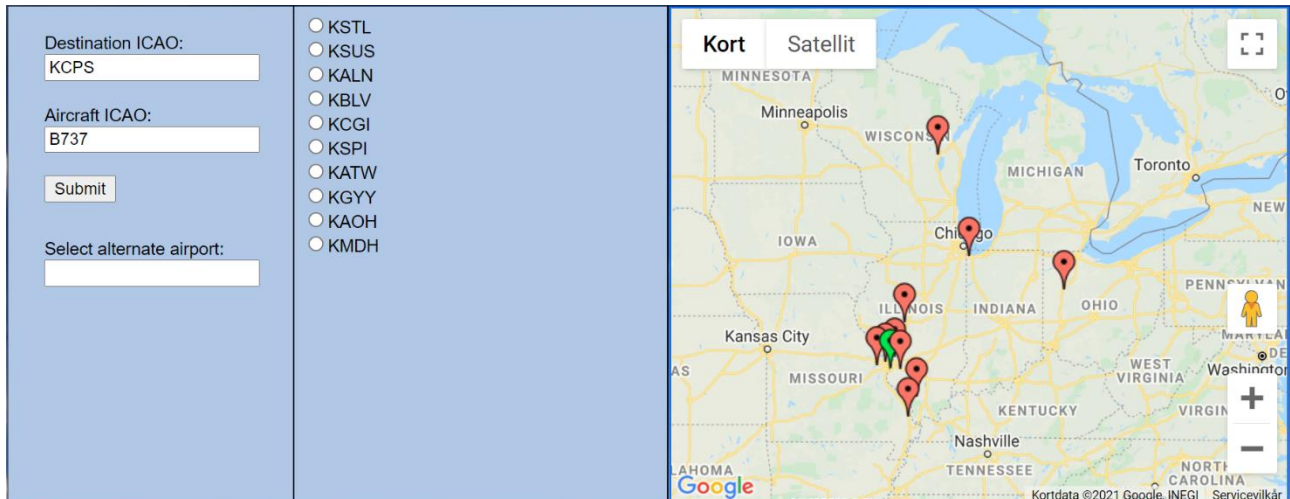


Figure 24 - Final Random Forest tested in graphical user interface

Multi-Layer Perceptron classifier

Figure 25 shows that the suggested alternates are more centered around the destination airport. The alternate furthest away is KEVV with a distance of 230km. According to the pilot interviewed earlier, this might not be the first choice, but it is still an acceptable distance to an alternate. However, the MLP classifier still occasionally suggest alternates too far away to be valid when exposed to larger aircrafts.

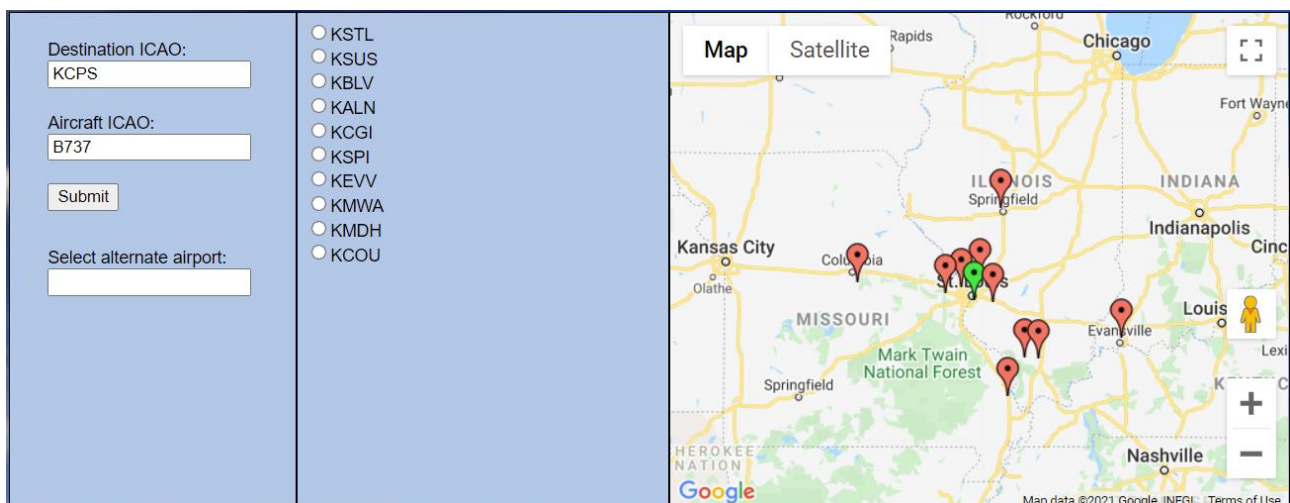


Figure 25 - Final MLP tested in graphical user interface

6.2.2 Pilot user testing

The point of this test was to have the pilot use the machine learning model and give his professional view on the quality of the alternates predicted. The test did not evaluate on the usability of the GUI. The pilot tested the GUI with input matching flight plans previously created by himself. Each alternate predicted by the machine learning model was shown to the pilot who then evaluated on the validity. This was done for each of the three different machine learning models. Overall, the pilot showed great satisfaction with the models and expressed most of the alternates to be valid with few outliers being over 1000 km away. He also mentioned the importance of showing the characteristics for each of the airports that the GUI presents, since they are a big part of the pilots' decision on the alternate. Some alternates predicted with the highest probability were not the first choice of the pilot, but this was due to company specific choices, like service agreements and specific customer requests when he flies the private jets. These are parameters that cannot be considered on the current model due to missing data.

Having tested the model on one pilot gave a limited evaluation since each pilot has their own preferences. Deploying the product in production would provide an option to store the index that the pilot chose from the list of predicted alternates and with that information evaluate the model based on the how high on the list it was. The recorded choices could help create a company specific model to predict the correct order of alternates and provide a better user experience for the pilots. This would serve as a better evaluation but was not possible within the scope of this project.

7 Discussion

Out of the three different classification algorithms evaluated, the most accurate turned out to be the Random Forest classifier which achieved an accuracy of 45.92% when using the standard test method and 94.13% when testing using the top-ten test method. This is a few percent improvement over the Decision Tree classifier and about ten percent more than the MLP classifier achieved on the top-ten test. However, after testing the models using the graphical user interface, it became apparent that the Random Forest classifier more often suggested invalid alternates by either being too far away or too small for aircrafts to land on. The MLP classifier was the model with fewest such cases. Therefore, it was considered the best model if one should be deployed to the final system in their current states. However, the Random Forest's high accuracy in the top-ten test opens the possibility of post-processing the output to remove any invalid suggestions while still including a valid airport in at least 94% of cases. Introducing this filter could change the conclusion on which model performed the best.

Having confirmed that machine learning can be used to predict valid alternates, it should now be considered whether the model trained throughout this project is better than ForeFlight's existing Alternate Advisor. According to ForeFlight, their current system does not take the flown aircraft into account. This has been achieved in the models developed in this project by training on the aircrafts' wake turbulence category allowing the model to distinguish flights requiring different runways.

Another problem with ForeFlight's Alternate Advisor is its inability to predict alternates outside the US. This is partially accomplished by this projects' model since it has been trained on data in Europe. If given enough data, nothing would prevent it from being trained to predict alternates from around the world. However, this model still suffers from the weakness of only being able to predict alternates that it has seen before. Classification models normally assume there exists a limited number of possible classes. This is also the case of alternates since the world contains a finite number of airports. However, it is not realistic to expose a classifier to every one of them. Furthermore, it is not unthinkable for new airports to be built making the classifier deprecated after a while. Instead of it predicting specific airport ICAO codes, a possible solution could be to build a classifier outputting certain abstract features expected by the alternate to have. Afterwards, a list of fitting airports could be found using the features. This would create an abstract model that for any given place in the world would be able to output a description of a suitable alternate, and a lookup in the database would give the specific alternates to show to the pilot. One of the most difficult parts of such a method would be to come up with a way of describing airports abstract enough to cover all airports but not too abstract making them indistinguishable from each other.

The Alternate Advisor is made for ForeFlight's mobile client. To accommodate the need for the software to work on different type of devices, it was chosen to go with a client-server model and the use of JSON to make requests to the model on a Python backend. Even though machine learning can be done in different programming languages, then python is the most popular choice because of the popularity in the language and the wide variety of machine learning libraries²⁹. A client-server model also helps with the distribution of a newly trained or updated machine learning model, instead of having each device needing to download and maybe decompress it. One of the biggest problems of the client-server model is if the pilot requires the model to be accessible offline. For example, to look up possible alternates mid-flight because both the destination and alternate airport becomes unavailable.

A feature ForeFlight wished to add to their Advisor was the possibility of updating it over time. Both the Random Forest and the Multilayer-Perceptron model can do what is called a "warm start", which means that it has the option to continue the training on the existing fit. This requires a pipeline that can filter and verify correct data before it is automatically fit into the existing model, to make sure it does not ruin the existing model. The warm start does not allow for the new data to outweigh the old one, which would have been preferred since it reflects the latest and most updated choices from the pilots. One might imagine that a given airport would get upgraded or downgraded and therefore becomes a better or worse choice. This change could be reflected in the new data and help the model for a better prediction. Foreflight registered 577,245 new flight plans in 2020, this averages to 48,103 flight plans per month. Meaning that every month foreflight would be able to update their model with over 48,000 new data points.

A test was created to evaluate how the MLP model compared to the Alternate Advisor. Each of the provided flight plans included the suggested alternates from the Alternate Advisor from when the flight plan was created. These suggestions have been extracted from the database for the test. The data contained several rows of null values and a response from foreflight confirmed that the null values were due to the system being incapable of providing a suggestion. Since ForeFlight's system only gives a list of five suggestions, the top-ten testing method was altered to fit a top-five test instead, and both systems were tested. The Alternate Advisor reached a score of 63.19% where the MLP classifier reached a score of 68.67% based on the exact same data. This means that the MLP classifier performs 5,48% better than the Alternate Advisor. This rather small improvement in accuracy shows its significance when compared to the approximately 570,000 flight

²⁹ (in.springboard.com, 2020)

plans handled by ForeFlight in 2020 resulting in more than 31,000 cases where the MLP model would succeed, and Alternate Advisor would not.

The test described in section 5.4.1 was created since there is not a single correct alternate for a given flight. This test evaluates on the top ten predictions outputted from the model, but it does not evaluate on the quality of those outputs. The model would score a positive point in accuracy if the correct alternate was in the list, even if it was at the bottom of the list. More problematic is the test not evaluating on the rest of the predictions in the list, which means the list could have outliers being too far away, the runway being too short or have some other problem. Even though this is the pilot's problem and responsibility in the end, then it is a severe problem for the quality of the model. A potentially more fitting accuracy measure would be the precision and recall at k evaluation method³⁰. This would determine the ratio of relevant elements in the system's top k recommendations. For each flight plan, this ratio can be stored to get an understanding about the quality of each list of suggestions. To distinguish between relevant and irrelevant suggestions, each alternate would be evaluated on the length of their runways and distance from the destination.

In summary, the models created throughout this project improved upon the existing Alternate Advisor on various measures. The model succeeds at predicting valid alternates in more cases while considering the aircraft. The model leaves room for improvement especially when considering its performance on larger aircrafts.

³⁰ (Malaeb, 2017)

8 Conclusion

Throughout this project, several different classification models were evaluated on their ability to predict valid alternate airports based on the criteria important to pilots. A pilot from Air Alsie was interviewed to gain insight into the process of selecting alternates. A flight plan's destination and aircraft type were important factors determining the validity of an airport as an alternate for a given flight. The flight plans provided by ForeFlight additionally contained features whose importances were experimentally evaluated. Analyzing the data beforehand proved essential to gain an understanding about how the machine learning would behave to later allow tweaking the data to best fit the problem. This understanding built the foundation for engineering the input to generalize it and make it more meaningful for the model by, for example, using coordinates instead of airport ICAO codes.

Testing showed that using the destination coordinates together with the aircraft wake turbulence category as input features resulted in the best model when evaluated on the top-ten test and qualitative assessment in the GUI. Including additional features sometimes resulted in higher accuracies but overall reduced the quality of the predictions by suggesting more outliers.

Of the three different classifiers, the Multilayer Perceptron was most successful. It achieved a score of 83% in the top-ten test. This seems low compared to other classifiers, but the final verdict was based on a visual evaluation showing that it most consistently created a complete list of valid suggestions.

Further testing was done comparing the MLP classifier to ForeFlight's existing Alternate Advisor. ForeFlight's solution achieves an accuracy of 63.20% while the MLP classifier has an accuracy of 68.67% when based on the same test. As mentioned in the discussion section, this rather small improvement shows its significance on the large amount of flight plans that Foreflight handles. Furthermore, the new model takes the size of the aircraft into account by changing suggestions based on weight. A limitation in both ForeFlight's and the models built in this project is their inability to predict alternates it has not been trained on making the new model non-functional outside Europe and USA.

9 Perspectivation

Throughout this project, it has been proved that it is possible to recommend alternates based on flight data using various classification models. However, this method leaves room for improvement. For the future of this project, it would be interesting to explore different approaches to building machine learning models and recommender systems.

One of the hurdles that the current models does not overcome is the ability to suggest alternates not seen during training. An interesting approach would be to engineer the input to contain features of the destination airport and aircraft, and the output to contain features reflecting the requirement for the alternate. Since the dataset no longer needs the specific ICAO, hopefully it would make the prediction usable for every airport in the world.

If the project were to continue, it would be a good idea to come up with a better method of evaluating the models. The top-ten method described in this report does not evaluate on the quality of each item in the list. For future work, it would be interesting to look into more relevant methods of evaluation such as the precision and recall at k method discussed earlier.

10 References

- analyticsvidhya.com*. (2015, 01 05). Retrieved from analyticsvidhya:
<https://www.analyticsvidhya.com/blog/2015/01/scikit-learn-python-machine-learning-tool/>
- AviationCloud. (2021, January 26). Retrieved from AviationCloud: <http://www.aviationcloud.aero/>
- Borgermann, A. W. (2021, 03 03). Pilot. (T. S. Laursen, & M. K. Jensen, Interviewers)
- Brownlee, J. (2019, December 14). *A Gentle Introduction to Imbalanced Classification*. Retrieved from Machine Learning Mastery: <https://machinelearningmastery.com/what-is-imbalanced-classification/>
- Brownlee, J. (2020, June 30). *Why One-Hot Encode Data in Machine Learning?* Retrieved from Machine Learning Mastery: <https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/>
- Federal Aviation Administration. (2021, May 10). *ICAO FLIGHT PLANS*. Retrieved from Federal Aviation Administration Web site:
https://www.faa.gov/air_traffic/publications/atpubs/fs_html/appendix_a.html
- ForeFlight. (2019, June 24). *blog.foreflight.com*. Retrieved from Foreflight Blog:
<https://blog.foreflight.com/2019/06/24/track-log-review-alternate-advisor-and-more-in-foreflight-11-5/>
- Github*. (2021, May 14). Retrieved from Github.com: https://github.com/scikit-learn/scikit-learn/blob/main/sklearn/tree/_classes.py#L242
- Hansen, H. (2021, 02 26). VP, Flight Planning Engineering. (T. S. Laursen, & M. K. Jensen, Interviewers)
- Heaton, J. (2017, June 1). *The Number of Hidden Layers*. Retrieved from Heaton Research:
<https://www.heatonresearch.com/2017/06/01/hidden-layers.html>
- in.springboard.com*. (2020, 08 31). Retrieved from Springboard Blog: <https://in.springboard.com/blog/best-language-for-machine-learning/>
- Kumar, S. (2021, March 02). *Stop One-Hot Encoding your Categorical Features - Avoid Curse of Dimensionality*. Retrieved from Medium: <https://medium.com/swlh/stop-one-hot-encoding-your-categorical-features-avoid-curse-of-dimensionality-16743c32cea4>
- Lutins, E. (2017, August 2). *Towards Data Science*. Retrieved from towardsdatascience.com:
<https://towardsdatascience.com/ensemble-methods-in-machine-learning-what-are-they-and-why-use-them-68ec3f9fef5f>
- Malaeb, M. (2017, August 13). *Recall and Precision at k for Recommender Systems*. Retrieved from Medium: https://medium.com/@m_n_malaeb/recall-and-precision-at-k-for-recommender-systems-618483226c54
- Open Data Science. (2019, October 1). *Missing Data in Supervised Machine Learning*. Retrieved from Medium: <https://medium.com/@ODSC/missing-data-in-supervised-machine-learning-b6df0f02a731>

- Rocca, B., & Rocca, J. (2019, June 03). *Introduction to recommender systems*. Retrieved from Towards Data Science: <https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada>
- Scikit-learn. (2021, May 15). *Scikit-learn LabelEncoder*. Retrieved from Scikit-learn documentation: <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html>
- Scikit-learn. (2021, 05 20). *Scikit-learn StandardScaler*. Retrieved from Scikit-learn documentation: <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html>
- Shaikh, R. (2018, October 28). *Feature Selection Techniques in Machine Learning with Python*. Retrieved from Towards data science: <https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e>
- Srinivasan, A. V. (2019, September 7). *Stochastic Gradient Descent — Clearly Explained !!* Retrieved from Towards data science: <https://towardsdatascience.com/stochastic-gradient-descent-clearly-explained-53d239905d31>
- Tan, P.-N., Karpatne, A., Kumar, V., & Steinbach, M. (2006). *Introduction to Data Mining*. Boston: Pearson.
- Tripathi, M. (2020, June 16). *Knowing all about Outliers in Machine Learning*. Retrieved from DataScience Foundation: <https://datascience.foundation/sciencewhitepaper/knowning-all-about-outliers-in-machine-learning>
- tutorialspoint.com*. (2021). Retrieved from Tutorialspoint: https://www.tutorialspoint.com/python_pandas/python_pandas_quick_guide.htm

List of Figures

Figure 1 - ForeFlight's Alternate Advisor	10
Figure 2 - The distribution of differently represented alternates in the provided flight plans	13
Figure 3 - The geographical distribution of the provided flight plans	14
Figure 4 - Illustration of dividing airports into squares	17
Figure 5 - Distribution of changes in alternate representations after removal of null values.	22
Figure 6 - Sorted distances between destination and alternate for all flight plans	23
Figure 7 - Finished decision tree built from small data example	27
Figure 8 - Path taken by unseen flight plan from Table 4	28
Figure 9 - A logical view of the ensemble learning method	29
Figure 10 - Base classifiers created by Random Forest using data from Table 5	32
Figure 11 - Simple perceptron	34
Figure 12 - Illustration of decision boundary created by simple perceptron	35
Figure 13 - Error surface over two weight parameters	37
Figure 14 - Screenshot from created graphical user interface	40
Figure 15 - Results from testing Decision Tree with various max depths	44
Figure 16 - Decision Tree impurity measures Gini and Entropy compared over various max depths	45
Figure 17 - Random Forest impurity measures Gini and Entropy compared over various max depths	47
Figure 18 - Accuracy of Random Forest over various max features	48
Figure 19 - Accuracy (blue) and size (red) of Random Forest over various n_estimators	48
Figure 20 - Accuracy of MLP using various activation functions	50
Figure 21 - Accuracy of MLP over various batch sizes	51
Figure 22 - Accuracy of MLP with various combinations of hidden layers	51
Figure 23 - Final Decision Tree tested in graphical user interface	52
Figure 24 - Final Random Forest tested in graphical user interface	53
Figure 25 - Final MLP tested in graphical user interface	53

List of Tables

Table 1 - Example data before One Hot Encoding	19
Table 2 - Example data after One Hot Encoding	20
Table 3 - Small example of flight plan data	25
Table 4 - Example of unseen flight plan with unknown alternate	28
Table 5 - Example flight plan data from Table 3	30
Table 6 - Bootstrapping subset generated from dataset in Table 5	30
Table 7 - Example of unseen flight plan with unknown alternate	33
Table 8 - Results from testing Decision Tree with different representations of destination and aircraft	43
Table 9 - Results from testing Decision Tree with additional features	43
Table 10 - Results from testing Random Forest with different representations of destination and aircraft	46
Table 11 - Results from testing Random Forest with additional features	46
Table 12 - Results from testing MLP with different representations of destination and aircraft	49
Table 13 - Results from testing MLP with additional features	50

11 Appendix

11.1 Appendix 1 - ICAO Model Flight Plan Form (Federal Aviation Administration, 2021)

Department of Transportation Federal Aviation Administration		International Flight Plan	
PRIORITY FF →		ADDRESSEE(S) EHAZQEX EBURZQZX EDDYZQZX LFFFZQZX LFRRZQZX LFBBZQZX LECMZQZX LPFCZQZX	
FILING TIME 1 9 0 8 3 6 →		ORIGINATOR E H A M Z P Z X →	
SPECIFIC IDENTIFICATION OF ADDRESSEE(S) AND/OR ORIGINATOR			
3 MESSAGE FPL	7. AIRCRAFT IDENTIFICATION A C F 4 0 2	8 FLIGHT RULES I	TYPE OF FLIGHT N
5. NUMBER -	TYPE OF AIRCRAFT E A 3 0	WAKE TURBULENCE CAT. H	10 EQUIPMENT S / C
13 DEPARTURE AERODROME E H A M	TIME 0 9 4 0		
15 CRUISING SPEED 0 8 3 0	LEVEL F 2 9 0	ROUTE LEK 2B LEK UA6 XMM/MO78F330	
UA6 PON URION CHW UAS NTS DCT 4611N00412W			
DCT STG UAS FTM FATIMIA			
16. DESTINATION AERODROME L P P T		TOTAL EET HR. MIN. 0 2 3 0	ALTN AERODROME L P P R
18 OTHER INFORMATION REG / FBVGA SEL / EJFL		2ND. ALTN AERODROME	
EET / LPFCO158			
SUPPLEMENTARY INFORMATION (NOT TO BE TRANSMITTED IN FPL MESSAGES)			
19 ENDURANCE HR. MIN. E / 0 3 4 5	PERSONS ON BOARD P / 3 0 0		EMERGENCY RADIO UHF VHF ELDA R / U V E
SURVIVAL EQUIPMENT S / POLAR DESERT MARITIME JUNGLE X X X X		LIGHT FLOURES UHF VHF J / L F X X	
DINGHIES NUMBER CAPACITY COVER COLOUR D / 1 1 3 3 0 C →		YELLOW	
AIRCRAFT COLOUR AND MARKINGS A / WHITE			
REMARKS X /			
PILOT-IN-COMMAND C / DENKE			
FILED BY AIR CHARTER INT.	ACCEPTED BY	ADDITIONAL INFORMATION	

FAA Form 7233-4 (5-93) Supersedes Previous Edition