FP_Section1_Group2_Phase4_Report

August 4, 2022

0.1 Phase 4 and Final Report

0.1.1 Prepared by Brian Moon, John Stilb, Jonas Degnan, and Shuhan Yu

0.2 Abstract

In this project, we are helping travel insurance underwriters to predict flight delays. Having the prediction on whether a flight would be delayed, which is defined as a flight arriving 15min (or greater) later than planned or the flight being diverted to another location or flight being canceled with Flight data from Airline On-time from US Department of Transportation (USDOT), and weather data from National Oceanic and Atmospheric Administration (NOAA) repository between 2015 and 2021. With extensive feature engineering and random undersampling to address class imbalanced, we have trained a number of model including logistic regression, linear support vector classifiser, random forest and neural network through cross validation and hyperparameter tunning. The best performing model is a random forest with 18 features with a f1 score of 0.402.

0.3 Project Description

0.3.1 Business Case

In 2019, the Federal Avaition Adminitration (FAA) estimated trip delays cost passengers $18.1 billion in lost time and productivity due to delays, cancellations, and missed connections. These losses have been rising around 10 percent per year since 2016 [11].

Given the breadth of travel insurance products available, our team focused strictly on Trip Delay policy products. These products vary, but typically cover expenses such as meals, hotel accomodations, transporation to/from hotel and airport, and an incidentals allowance. Further complicating this product category are the varied contractual agreements between carriers and passengers, which offset passenger trip delay costs depending on the reason for delay.

A delay classification model would benefit actuarial analyses used to set premium prices by providing a risk profile to associate with each policy that is not easily captured by existing data metric pipelines.

The key actuarial metrics used in industry and their definitions are listed below. - **Exposure**: the quantity used to determine relative level of risk. - **Premium**: exposure/time-dependent product price (generally tied to exposure criteria) - **Loss**: claims paid

Operationalization A classification model can operationalize key components of these metrics, most importantly exposure, as the model F1 metric directly supports exposure determination. Each policy premium (within the forecast window) can have a more appropriate delay-risk driven pricing; this applies to individual sales (a specific filght route is known to have higher delay risk during certain months) and distributor wholesale pricing (higher premiums on bundled policies on flights matching model parameters).

The loss metric can also be supported given the forcasting component of the model. Typically, distibutors (e.g., Kayak) will sell Travel Delay policies bundled with their primary travel products. In many instances, the distributors do not report policyholder data to the insurer in a timely fashion, resulting in considerable difficulties managing capital reserves for potential claims (in some instances claims are filed *before* an insurer knows of that policyholder)[10]. An insurer can mitigate claim reserve estimation variance by analyzing future flight data against this model to understand potential aggregate flight delay risk, and adjust claim reserves accordingly, freeing captial for other business opportunities.

Another benefit of assessing this particular product, in contrast to a Trip Cancellation policy, is that the customer behavior variances are not confounding variables in our model. Given the datasets provided, this product is well suited for delay classification modeling.

One limitations of this model includes round-trip flights where the return trip falls ouside the forecast window.

0.3.2 Dataset

To create a model that would help travel insurance underwriters predict flight delays, we have used a dataset consisting of 168 columns and containing information on flight, hourly weather, departure and arrival airports, airline, and etc. Such data was constructed by the data engineering team using three underlying datasets, which are described in more detail below.

The Airline On-time Performance Data ("Flights Data") was obtained from the US Bureau of Transportation Statistics. The data is generated through carrier reporting on-time data for the flights they operate. Data dates back to 1987. However, given the rapid change in the air transportation industry landscape, we would focus on the subset of the data from 2015 to 2021 as the more recent data would reflect the current landscape.

Another dataset was obtained from National Oceanic and atmospheric Administration repository ("Weather Data"). This is a subset of data from the Integrated Surface Database (IDS), which consists of hourly and synoptic surface observations. It is an integrated dataset from more than 100 original data sources.

In addition, the team has relied on data from the US Department of Transportation as well ("Station Data"). This dataset contains information for nearby geographic locations (including major airports) and the distance between the two.

0.4 Exploratory Data Analysis and Feature Engineering

0.4.1 Exploratory Data Analysis

We have first performed a set of Exploratory Data Analyses ("EDA") on the dataset provided by the data engineering team to check for data issues, determine helpful features, and determine the need for feature engineering.

Missing values

- Hourly Precipitation: missing for 7.97% and 8.02% of the time. Precipitation information seems to be missing as precipitation is not reported in certain locations. For example, in Rickenbacker International, Stewart International, and Hilton Head Island Airport, precipitation is never reported, and in 35 airports, precipitation is not reported more than 90% of the time. This may be due to different reporting standards by each airport, or lack of equipment to measure such information.
- Wind Direction: missing for 3.92% and 3.96% of the time. Wind direction information is more often missing during the night and early morning. Between 00:00 and 10:00am, DEP_HourlyWindDirection is missing for 8.63%, whereas between 10:00am and 11:59pm, it is missing for only 1.98%. This may be due to measurement of wind direction involving human judgment or observation, and the lack of staff during such hours.

Imbalanced Classification According to our `DELAY_FLAG`, only 20.18% of the data is classified as having an issue (due to either cancellation, significant delay, or diversion). As one class is four times more frequently occurring than the other, it may lead to imbalanced classification and may require rebalancing to obtain accurate results. We deal with such issue by using random undersampling, which we will discuss in more detail in later sections.

Correlation between Explanatory Variables As we can see in the correlation heatmap below, some explanatory variables such as station pressure and dry bulb temperature, or precipitation and humidity have strong correlations. We erred on the side of including only one of the variables between the two that demonstrate strong correlations.

Distribution As shown in the histograms below, most features have a left-skewed distribution. Some variables such as wind speed and precipitation have outliers, which may be representative of natural disasters. We create an additional feature to capture the effects of natural disasters or extreme weather conditions in our feature engineering section.

Explanatory vs. Fitted Variables Scatter plots below demonstrate the relationship between delay time. We can immediately note that there are more flight delays with more extreme weather conditions. We have attempted to account such effects of extreme weather on flight status changes through an engineered feature, as mentioned previously.

0.4.2 Feature Engineering

To better represent what we aim to predict, we have engineered our fitted variable `DELAY_FLAG` to have value of 1 if a flight satisfies one of the following conditions: (1) delayed by more than 15 minutes; (2) cancelled; or (3) diverted.

To enhance the predicting power of our model and reduce feature space and model runtime, we have engineered the following explanatory variables:

Natural disasters and more weather related features

- Present extreme weather condition flag
  - This variable takes value of 1 if the current origin or destination weather is experiencing *either* (1) ice crystals, (2) snow, (3) hail, (4) fog, (5) smoke, (6)storm, *or* (7) haze.
  - Under such extreme weather conditions, we are hypothesizing that there will be more changes to the flight status.

- Icy runway
  - This variable flags when there was (1) any precipitation over the past 6 hours *and* (2) the dry bulb temperature was below freezing point.
  - We have hypothesized that another common cause for flight delays or cancellations is frosty runways and airplane wings.
  - Interacting this term with the airport type variable may further increase predicting power of our model, as smaller airports may not have the resources and equipment to defrost runways or deice plane wings, as much as larger airports do.

Time-based features

- Prior extreme weather condition flag
  - This variable takes value of 1 if the current origin or destination weather has experienced *either* (1) ice crystals, (2) snow, (3) hail, (4) fog, (5) smoke, (6)storm, *or* (7) haze over the past 6 hours.
  - This flag intends to capture weather trends and immediately recent weather conditions that would be factored in the decision-making for flight status changes.
- Average `DELAY_FLAG` over different time windows (3 months, 2 weeks, 1week, 1 day, 6 hours, and 1 hour)
  - This feature will capture different meanings depending on the length of the time it represents. The prior 3-month average may capture cyclical factors, such as labor union movements or labor shortages, that affect flight cancellations and delays over a longer horizon. The prior 1-hour average on the other hand, may capture recent weather trends.

Event-based features

- Holiday indicator, which takes value of 1 if the date falls within $\pm$ 1 week range of major US Holidays (Christmas, New Year's Eve, Labor Day, July 4th, Memorial Day, Spring break, Superbowl, New Year's).
  - Flight schedules over Holiday seasons may tend to be more or less busy than those over non-Holidays, and thereby lead to more or less flight status changes.
- Weekend indicator, which takes value of 1 if it is either Saturday or Sunday
  - Flight schedules over the weekend may tend to be more or less busy than those over weekdays, and thereby lead to more or less flight status changes.

Graph-based features

- Airport Congestion PageRank
  - Flight cancellations, delays, and diversions may also be affected by the traffic level of the origin/destination airports.
  - This variable ranks the airports by flight volume.

Airline related features

- Size of the airline, based on number of flights
  - Smaller airlines may tend to have more staffing constraints, or smaller aircrafts, which may be more likely to be delayed or cancelled.
  - This variable takes value of 1 if the airline lies in the lower 25 percentile in terms of flight count

To assess which explanatory variables would be particularly more helpful to include, we have compared the correlation to the fitted variables.

Explanatory Variable	Correlation to DELAY_FLAG
MONTH	-0.03656742439160803
DAY_OF_WEEK	-0.007518705075137238
extreme_weather	0.10355900309614259
small_carrier	0.0021535053274980512
AIR_TIME	-0.04029889769854789
DISTANCE	0.008113633979236668
DEP_HOUR	0.12397310582086053
ARR_HOUR	0.10777435954643888
origin_weather_HourlyWindSpeed	0.026819942709151796
dest_weather_HourlyWindSpeed	0.0326785311424282
origin_weather_Avg_HourlyVisibility	-0.07867630285428849
dest_weather_Avg_HourlyVisibility	-0.08206545776556229
num_flights_past_30_days	0.0013049
icy_runway	0.06608226803934573
num/avg flights past 7 days	0.00143
avg delay past 7 days	0.181566
weekend	-0.007211151473871182
holiday_month	-0.021093165425109094

We can notice that extreme_weather, arrival/departure hours, origin/destination visibility, icy_runway, and average delays in past 7 days demonstrate strong negative or positive correlations with the DELAY_FLAG.

0.5 Data Leakage

Data leakage is when outside information is used in training that wouldn't be available at the time of prediction and is a serious problem for any machine learning project. For example, if our model was to use some pilot information from an outside dataset that wouldn't be available during production, our model wouldn't work. In order to prevent this, we have only used information that will be available in production and have created transformations to ensure that the production data matches our models' requirements. In addition to data leakage there are many other concerns for a machine learning project: overfitting, bad data, and overgeneralization of model results.

To combat overfitting, we separated our data into training and tests sets so that we weren't optimizing our models upon the test set. This gives us a much more accurate view of our expected performance. Additionally, we incorporated regularization into many of our pipelines to introduce noise, thus mitigating overfitting. Lastly, we used cross validation to tune our hyperparameters which limits the risk of overfitting by repeatedly partitioning random samples of our data and holding one out to test performance and then taking the average of our results.

Bad data comes in many forms including missing values, confounding variables, and label imbalance. We performed a thorough EDA to understand which features had many missing values and adjusted our pipeline to handle them. Additionally, we were careful in our feature selection to avoid confounding variables or variables with incorrect/unhelpful records. Lastly, we used undersampling to handle the major class imbalance we saw.

Our model has limitations (discussed at depth in a later section) and we by no means recommend its use in any other scope than that which we propose: to predict whether flights will experience a "delay"—up to a certain degree of confidence—for underwriting purposes.

0.6 Algorithm Summary

When it comes to the modeling choices, given the dimension of the data set, a key concern is the speed performance of the modeling pipeline, especially on the prediction side that we would like to ensure a prediction can be generated in time. Therefore, we did not consider models such k-NearestNeighbours but the ones scale well with dimension. The list of the models we experimented and associated loss functions are listed below:

0.6.1 Models & Loss functions

- Loistic regression model with "Log Loss" loss function also known as cross entropy loss.

$$L(w; x, y) = -[ylog(p) + (1-y)log(1-p)$$

- Linear support vector machine model with the "Hinge Loss" loss function to maximise classification margin.

$$\ell(y) = \max(0, 1 - t \cdot y)$$

- Random forest model with "Gini Impurity" loss function, which indicates the likelihood of new, random data being misclassified if it were given a random class label according to the class distribution in the dataset.

$$\sum_{k \neq i} p_k = 1 - p_i$$

- Multilayer perceptron model (the neural network model) was also exaimed upon stakeholder request, which also uses the log loss function has shown above.

0.7 Pipeline Discussion

We have constructed the data pipeline to transform the training and testing data into a format that the modeling pipeline can consume. The categorical variables has been encoded through One-HotEncoders to transform them into binary vectors to support model training. Then all variables are normalized via MinMaxScaler. The data is normalized to prevent models biasing one specific set of variables as opposed to others due to the scale. Specifically, we chose a MinMaxScaler to scale the features to a range of [0,1] as opposed to StandardScalers so we can preserve the shapes of the features during training.

We also addressed the imbalance in the dataset. As seen in the EDA, the delayed vs non-delays shows an 8:2 ratio across the data set. In previous experiments on unbalanced data, our machine learning models have favored delayed predictions leading to a poor recall score. We first examined the recent popular Synthetic Minority Oversampling Technique (SMOTE), which oversamples the minor class by synthesizing new examples from the minor class. A fully functioning SMOTE pipeline was built with reference to [6]. However, upon examining SMOTE methods, we saw significant performance degradation with larger datasets as it relies on identifying the nearest neighbors of each sample. With increase in the number of samples and data dimension, SMOTE takes a considerably long time to perform.

Upon further literature review, a number of papers show that SMOTE does not perform so well on high-dimension big data problems (e.g., [7], [8], [9].) Specifically, [8] has suggested that Random Undersampling has outperformed other methods when addressing class imbalance issues. Therefore, we have implemented random undersampling in the training process.

0.7.1 Input Features

Previously, we ran into issues with the large number of categorical variables that had been inflating our feature space. This led us to summarize as much of the categorical information into boolean and numeric variables, by creating our icy_runway, small_carrier, holiday, weekend, extreme_weather, and pagerank variables.

For our set of experiments we used two different families of features: our limited set, which consists of our most valuable features determined through EDA, and our extended set which includes several additional weather variables and our month variable. We chose these two sets, because we believe that the limited set would reduce the risk of our feature space exploding while the extended set allowed us to explore how our models handled more information.

Raw	Derived Features	Created Features
`AIR_TIME`	`Avg_delay_past_7_days`	`DELAY_FLAG`
`origin_weather_HourlyWindSpeed`	`icy_runway`	
`dest_weather_HourlyWindSpeed`	`small_carrier`	
`origin_weather_Avg_HourlyVisibility`	`DEP_HOUR`	
`dest_weather_Avg_HourlyVisibility`	`ARR_HOUR`	
`OP_UNIQUE_CARRIER`	`holiday`	
	`weekend`	
	`extreme_weather`	

Limited Features

Raw	Derived Features	Created Features
`AIR_TIME`	`Avg_delay_past_7_days`	`DELAY_FLAG`
`origin_weather_HourlyWindSpeed`	`icy_runway`	
`dest_weather_HourlyWindSpeed`	`small_carrier`	
`origin_weather_Avg_HourlyVisibility`	`DEP_HOUR`	
`dest_weather_Avg_HourlyVisibility`	`ARR_HOUR`	
`OP_UNIQUE_CARRIER`	`holiday`	
`origin_weather_Avg_HourlyStationPressure`	`weekend`	
`dest_weather_Avg_HourlyStationPressure`	`pagerank`	
`DISTANCE`	`extreme_weather`	

Raw	Derived Features	Created Features
`MONTH`		

Extended Features

0.7.2 Cross Validation and Hyperparameter Tuning

Each model we have chosen has a number of hyperparameters such as regularization that could impact the model performance. Therefore, the cross validation technique has been adopted so that we can fine tune the hyperparameters. However, given the embedded time series nature of the flight data, we implemented blocking cross validation as opposed to popular k-fold cross validation. In each iteration of the cross validation, one year of data is taken as the training set and the subsequent year is used as validation set. Specifically for the last year in the training set, a 8:2 split is used for training and validation. The cross validation will identify the best set of hyperparameters for each model. In terms of the best model, the cross validation will return the model with best hyperparameters trained with the last set of train data to reflect the shifting landscape through time. And finally the 2021 data is used as blind test data to provide test score.

Below is a list of hyperparameters tuned for each model. | Model | Parameters |Value | | ——— | ——— |——— | | Logistic Regression | regParam |0.1, 0.01| | Linear Support Vector Classifier | regParam |0.1, 0.01| | Random Forest | maxDepth |5,10 | | |numTrees|50,70| | Neural Network | layers |[input, 100,50,2], [input,50,50,50,2]|

In addition to the hyperparameter tuning, to avoid overtraining, we also implemented early stopping with a maximum number of iterations as all models are trained through an iterative approach.

0.7.3 Experiments - Jm & Jonas(to incorporate final features and run last set of experiments)

- Number of experiments conducted
- Experiment table with the following details per experiment:
  - Baseline experiment
  - Any additional experiments
  - Final model tuned
  - best results (1 to three) for all experiments you conducted with the following details
    - **Computational configuration used** (go to Event Log in Spark UI: timestamps of cluster up/downscaling events)
    - **Wall time for each experiment**

Model	Parameters	Best Value
LR	regParam	0.01
LSVC	regParam	0.01
RF	maxDepth	10
	numTrees	70
NN	layers	[input,50,50,50,2]

Need to add below info to experiments table

num 1 Model: Random Forest
Precision = 0.339
Recall = 0.586
F1 = 0.429
time = 1132.5422

num 2 Model: LSVC
Precision = 0.322
Recall = 0.537
F1 = 0.402
time = 110.1324

num 3 Model: Log Reg
Precision = 0.314
Recall = 0.558
F1 = 0.402
time = 110.0524

Prev Best Model: Random Forest
Precision = 0.927
Recall = 0.087
F1 = 0.160
time = ?

model test_precision test_recall test_f1 train_precision train_recall train_f1 train_time test_time 0 lr 0.337340 0.432272 0.378951 0.319801 0.437673 0.369566 95.386972 0.099186 1 lsvc 0.349976 0.383192 0.365831 0.331449 0.378282 0.353320 95.700415 0.066915 2 rf 0.607294 0.161293 0.254889 0.606024 0.194687 0.294701 889.120048 0.076450 3 nn 0.281564 0.382854 0.324488 0.271128 0.366535 0.311694 1527.420244 0.078564

0.8 Neural Network

As stakeholder requested, we trained a neural network model during the cross validation model training. We have exprimented varying depth and width of the neural network with intermediate layers using sigmoid (logistic) function:

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

and the output layer use softmax function:

$$f(z_i) = \frac{e^{z_i}}{\sum_{k=1}^{N} e^{z_k}}$$

Specially, during cross validation, we have exmained 2 or 3 hidden layers. The best performing neural network has an architecuture of 18-50-50-50-2, where the input layer matches the size of input vectir and the last layers matches the binary predition classes. This best model achieved a test f1, which is better than the baseline model. While the model performance is better than the baseline model. it has not been the best overall model among the ones we trained, and the training time has been noticebly longer than the other models, which would have impact the whether the model should have been put into production. There are few aspects we can potentially improve the neural network model: - experiment more complicated architecture with other activation function and layer structure such as using RELU and convolution. - test deeper nerual network - allow more iterations in the optimisation. To control the overall training time, an early stop has been employed in our training process. However, neural network may benefit from more training iterations.

0.9 Novel Approaches

We tried two novel approaches: smote and a voting classifier.

Synthetic Minority Oversampling Technique (SMOTE), oversamples the minor class of a dataset and is used to balance data which has an uneven predictor class distribution. We built several different iterations of a SMOTE pipeline which worked on small subsets of data. However, as we began to try and balance our entire training dataset, we quickly ran into computational and memory issues. After some research we instead opted to use random undersampling, which has proven to be more effective in data balancing while avoiding the memory and computation issues. Random undersampling works by randomly selecting samples to delete from the majority class, which leads to fewer training records, but a more balanced dataset.

In an effort to improve our predictive performance, we explored using a voting classifier. This type of classifier is an ensemble method, which takes the majority decision of a given set of classifiers. There is no native voting classifier in pyspark and the sklearn object requires the data to exist in an array form of x & y variables. We tried several transformations, but ultimately couldn't come up with a solution in our limited time that didn't require us reworking our entire data pipeline.

0.10 Evaluation Metrics (Jm/Jonas)

Discuss evaluation metrics in light of the business case

We used three key metrics to evaluate the performance of our models: `precision`, `recall`, and an `F-beta` score (`beta=1`).

`Precision` indicates the quality of a positive prediction made by the model. In our case it represents the percentage of correct "delay" predictions out of all "delay" predictions.

$$Precision = \frac{TP}{TP + FP}$$

`Recall` measures our models ability to detect positive samples. In our case it represents our models' ability to detect "delays".

$$Recall = \frac{TP}{TP + FN}$$

`F-beta` scores combines both precision and recall to summarize the overall predictive performance of the model. By selecting `beta=1`, our primary measure for our overall's model performance gave equal weighting to `precision` and `recall`, because it balances our model's ability to correctly classify "delays' ' while ensuring our model doesn't only predict"delay" since it is the majority class.

$$F_1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

Each metric is bounded between 0 and 1, higher the value, better the score.

Our stakeholder values both `F1` components similarly as they can impact their pricing model and decision on when and where or if to allocate resources to customer acquisition or product advertising campaigns.

`Recall` impacts false negatives rates and, if low, undervalues the risk associated with the incorrect delay classifications, thus products priced based on the model will be lower than than should be, resulting in potential revenue losses. Low `precision` results in false positives that overestimate flight delay risk and raise product prices accordingly, but low precision may hurt the brand and overprice their products in the market, allowing competition to undercut and take market share.

Given that the majority of product sales are done during a bundled travel purchase, customers that skip travel insurance typically do not seek the product after the bundled transaction. Regarding advertising, using a high `F1` delay classification model, customers that skipped the initial purchase can be more accurately targeted for a potential product sale as their flight date approaches the model forecasting window. Additionally, absent specific customer sales channels, target markets can be served product advertisments when favorable conditions are predicted by the model.

0.11 Performance and Scalability Concerns

Throughout the experiment, we have seen several scalability issues: **SMOTE**: As discussed in the Pipeline Discussion section, SMOTE does not perform well as dimensions grow larger due to the need to compute distances between samples. While having a fully functioning SMOTE pipeline but constraining resources, the run time is long. Upon literature review, it has not proven to be the best strategy to tackle an imbalanced dataset, therefore has not been incorporated into the full pipeline. **CV**: it has been computationally demanding given the large number of samples as we need to iterate over the data for multiple aspects. First, through feature engineering, we have condensed the information held in the data set with a smaller number of features. For example, we extract weekends as binary variables as opposed to using days of the week as categorical variables. Secondly, when handling an imbalanced dataset, we adopted random undersampling to reduce the number of training data. Lastly, given the time series nature of the flight data, we adopted blocking cross validation instead of the k-fold validation, which also reduces the amount of data passing through training each time. Overall, although training time for some of the models is high, the test prediction time is acceptable on the blind fold set. Therefore, we consider this issue not concerning.

0.12 Limitations, Challenges, and Future Work - Jm

We ran into many challenges and faced many limitations during this project including a lack of domain expertise, time, and computational resources as well as computational inflexibility. Given our limited time, the remote nature of our team, and the size of the data, rapid experimentation was difficult. We weren't able to adjust our cluster to allow for multiple people to be working while an experiment was running. We overcame much of this through clever partitioning of our data, undersampling, and running experiments on smaller sets of data with select features. We weren't entirely able to overcome the lack of domain expertise, but we mitigated its effects through a thorough exploration of our features and a fair amount of external research.

Given our limitations and challenges, our model has only achieved an f1 score of 0.429, indicating it has limited predictive power. Additionally, our model is limited in its ability to generalize beyond similar data and will start to decay without re-training. Our first next step is to optimize model performance, through additional feature engineering, advanced model architecture development, and extended hyperparameter tuning. Once we have met an acceptable performance range, we will develop a plan to put our model into production and ensure we have proper data decay tracking and retraining systems in place. Lastly, we push the model into production and provide maintenance and oversight services for an agreed upon period of time.

0.12.1 Model Choices

We initially chose several simple linear models, logistic regression and support vector machine, to understand our baseline predictive ability. While these models performed generally well, we knew that there were likely some nonlinear patterns that they wouldn't be able to recognize thus limiting our predictive ability.

Given our lackluster performance, we decided to implement two new models: random forest and a neural network. Both of these are nonlinear models which we believed would help overcome the limitations of our existing models. Random forest is an ensemble method for combining many decision trees to vote on a particular classification. It also had interesting hyperparameters to tune, max depth and the number of trees, which we were able to optimize via cross validation giving us our best performance across all of our models. The neural network we developed used the MultiLayerPerceptronClassifier object from pyspark. While the pre-developed object offered many benefits: ease of use, ease of integration, and optimized for PySpark it also limited our ability to experiment with architectures and activation functions that would be available in a platform such as TensorFlow. In our next steps, we plan to experiment much more with neural network architectures to maximize our performance.

0.12.2 Unbalanced Data

The flight data is heavily unbalanced wuth a rough 8:2 split, meaning that only 20% of the training data are flagging delays. This can be a disadvantages in the model trainning as models can adversely favours the non-delay class as a blind assignment of non-delay to all data points can gives a decent testing result. Altough the stakeholder has indicated preference of using SMOTE in addressing inbalanced dataset, upon large amount of literiture review, we recogonised that SMOTE is not the best performing technique when in comes to high-demision data, and tend to worse the pipeline training time. Therefore, we have decided to adopt random undersampling to address this particular issue.

0.12.3 Feature engineering

We started inital modelling attempt using the raw data points available including flight meta data such as time and weather data. Howeber, soon we realise such selection will result in a large feature space. For example, when looking at MONTH, each month will be encoded as a binary class which essentially increased one categotical variables to 12 for modelling purposes. The large feature space has extended the model training time significantly, therefore we started to encapsulate the information through feature engineering. For example, we extract the holiday related period as a binary flag from all date related variables. In the weather data space, we were able to identify the factors that will impact the flighttime such as icy runway. We had also attempt to derive further information such as airport popularity/ business from page rank of the airport. Through feature engineering, we are able to retain/ extend the information in the data set using less number of features, which descreases the trainning time while boosting the test score. If time allows we shall further explore other possible features.

0.13 Bibliography

[1] Dou, Xiaotong. 2020. “Flight Arrival Delay Prediction and Analysis Using Ensemble Learning.” In 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), 1:836–40. https://doi.org/10.1109/ITNEC48623.2020.9084929. [2] Hasanin, Tawfiq, Taghi M. Khoshgoftaar, Joffrey L. Leevy, and Richard A. Bauder. 2019. “Severely Imbalanced Big Data Challenges: Investigating Data Sampling Approaches.” Journal of Big Data 6 (1): 107. https://doi.org/10.1186/s40537-019-0274-4. [3] Nigam, Rahul, and K. Govinda. 2017. “Cloud Based Flight Delay Prediction Using Logistic Regression.” In 2017 International Conference on Intelligent Sustainable Systems (ICISS), 662–67. https://doi.org/10.1109/ISS1.2017.8389254. [4] Rong, Fei, Li Qianya, Hu Bo, Zhang Jing, and Yang Dongdong. 2015. “The Prediction of Flight Delays Based the Analysis of Random Flight Points.” In 2015 34th Chinese Control Conference (CCC), 3992–97. https://doi.org/10.1109/ChiCC.2015.7260255. [5] Yazdi, Maryam Farshchian, Seyed Reza Kamel, Seyyed Javad Mahdavi Chabok, and M. Kheirabadi. 2020. “Flight Delay Prediction Based on Deep Learning and Levenberg-Marquart Algorithm.” Journal of Big Data 7 (1): 106. https://doi.org/10.1186/s40537-020-00380-z. [6] hwang018, 2020, “spark-pipelines”, https://github.com/hwang018/spark_pipelines. [7] Blagus, Rok, and Lara Lusa. 2013. “SMOTE for High-Dimensional Class-Imbalanced Data.” BMC Bioinformatics 14 (1): 106. https://doi.org/10.1186/1471-2105-14-106. [8] Hasanin, T., Khoshgoftaar, T.M., Leevy, J.L. et al. Severely imbalanced Big Data challenges: investigating data sampling approaches. J Big Data 6, 107 (2019). https://doi.org/10.1186/s40537-019-0274-4, [9] Leevy, J.L., Khoshgoftaar, T.M., Bauder, R.A. et al. A survey on addressing high-class imbalance in big data. J Big Data 5, 42 (2018). https://doi.org/10.1186/s40537-018-0151-6 [10] Roth, et. al. 2018. “Travel Insurance an Actuarial Perspective.” American Academy of Actuaries, September, 28. https://www.actuary.org/sites/default/files/files/publications/TravelInsuranceMonograph_09052018.pdf. [11] Lukacs, M. 2020. “Cost of Delay Estimates.” Federal Aviation Administration, July, 8. https://www.faa.gov/data_research/aviation_data_statistics/media/cost_delay_estimates.pdf.