BIA – 678 Big Data Technologies
Spring 2022

Prediction of Flight Delay using Flight Indicators

Team 12

Huilin Zhang(Methodology)

Juan Guo(Data and Data Preparation, Measuring the impact of scale on quality of analysis and performance on local)

Xiaorui Wang(Measuring the impact of scale on quality of analysis and performance on cluster, Evaluation)

Zhaoyi Xu(Proposal, meeting arrangement, Introduction and Conclusion)

Introduction

After the covid-19 pandemic, international flight tickets were very limited, and most airlines cut off their flights. On the other hand, due to various border inspection requirements and different countries' centers for disease control and prevention, lots of countries need covid PCR test taken in a certain time frame. Because of these two reasons, many international students have to purchase a non-direct flight and transfer to the US to make sure they catch all the international flights and get to the border on time.

In this paper, we are going to use domestic flight information to implement our machine learning model to predict flight delay, and according to the federal aviation administration, “Flight delays of less than 15 minutes are not reported in Operations Network”, so we are going to use 15 mins after scheduled arrival time as the boundary to classify on-time and delay in this project. In this paper, we are going to use Random Forest, Decision Tree, Logistic Regression, Linear support vector machine, and Gradient-Boosted Tree Classifier.

Data and Data Preparation

Data Description

https://www.kaggle.com/datasets/tylerx/flights-and-airports-data

The data set we used for analysis contains two data files which are flights and airports. We obtained this dataset from an open data source at kaggle. The flights file includes 2702218 data with 7 columns. The airports file shows the exact address of the airportID, which includes 365 addresses with 4 columns. There are 5561 duplicate data and 5561 missing values in the flight dataset. A detailed description of data set attributes is shown as Table 1.

Table 1. Attribute description for the data set.

Attribute Name	**Description**	**Type**
Day of month	Date of flight	Integer
Day of week	Day of the week	Integer
Carrier	Carrier code that represents the carrier company	String
OriginAirportID	Departure AirportID	Integer
DestAirportID	Destination AirportID	Integer
DepDelay	Departure delay of the flight	Integer
ArrDelay	Arrive delay of the flight	Integer

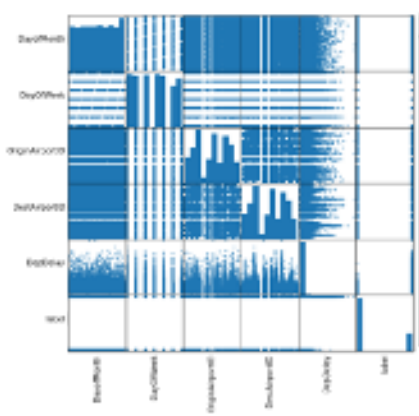
Data Processing

1. Dropping the duplicate data and missing values

Minor adjustments were made to the data set, including dropping the duplicate data and missing values. There were 5561 out of 2702218 data with duplicate data and missing values that means the deletion will just have little effect on the overall data set distribution.

2. Correlations and distribution

We use the function scatter matrix to get the correlations between independent variables shown as Fig. 1, It is obvious that there are not highly correlated numeric variables. Therefore, we will keep all of them for the model. The summary statistics for numeric variables shown as Fig. 2.



	0	1	2	3	4
summary	count	mean	stddev	min	max
DayofMonth	2696657	15.799071591233146	8.801337925743184	1	31
DayOfWeek	2696657	3.9004070595555906	1.9864668165468373	1	7
OriginAirportID	2696657	12742.454167882679	1502.0468528618916	10140	15376
DestAirportID	2696657	12742.858206290233	1502.0033937097596	10140	15376
DepDelay	2696657	10.532424034647343	36.063715485293585	-63	1863
label	2696657	0.2464013777058039	0.4309150816912807	0	1

Fig. 1 correlations between independent variables Fig. 2 summary statistics for numeric variables

3. Features and label

In our study, our goal is to predict flight delays, therefore, we select all of columns except ‘ArrDelay’ to use as features and use column ‘ArrDelay’ to create a Boolean label field named label with values 1 or 0(we consider the arrdelay time more than 15min as delay). Specifically, 1 for flights that arrived late, 0 for flights that were early or on-time.

4. Built the pipeline

In order to prepare the data for modeling and train a predictive model. we created a pipeline with seven stages: which include StringIndexer estimator that converts string values(‘Carrier’) to indexes for categorical features, VectorAssembler that combines categorical features("CarrierIdx", "DayofMonth", "DayOfWeek", "OriginAirportID", "DestAirportID") into a single vector, VectorIndexer that creates indexes for a vector of categorical features, VectorAssembler that creates a vector of continuous numeric features("DepDelay"), MinMaxScaler that normalizes continuous numeric features("DepDelay") and VectorAssembler that creates a vector for the categorical and continuous features then output a single vector "features". We get 'features' column and ‘label’ column, then we use the pipeline to combine multiple Transformers and Estimators together to specify our machine learning workflow.

5. Randomly split the data

At last, the data set is randomly divided into a training set and a validation set, in order to compare the effect on different data size, we split the dataset as 50%, 60%, 70%, 80% and 90%.

Methodology

Machine Learning Algorithms

We totally implemented 5 models in this project, they are Random Forest, Decision Tree, Logistic Regression, Linear support vector machine, and Gradient-Boosted Tree Classifier. Then we will evaluate these 5 models and choose the best result. In the following section, we will explain these 5 models.

1. **Decision Tree:** A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.(15 April 2022, Wikipedia)
2. **Random Forest:** Even though this is a binary classification problem, Random forest, a multinomial classifier, is also a suitable candidate for our modeling. There are some advantages in the Random Forest algorithm. First, it decreases the risk of overfitting. Second, the random forest provides flexibility. Third, random forest is easy to determine the feature importance.
3. **Linear SVC:** The objective of Linear SVC is fitting the data and return a best fit hyperplane that divides or classifies the data. The advantages of support vector machines are: (1). Effective in high dimensional spaces. (2). Still effective in cases where the number of dimensions is greater than the number of samples. (3). Uses a subset of

training points in the decision function (called support vectors), so it is also memory efficient. (2007 - 2022 scikit-learn developers)

4. **Logistic regression:** Since we are dealing with a binary classification, Logistic regression model with Sigmod function $\varphi(z) = (1 + e{-}z\ ){-}1$ is applied**.** Logistic regression is a simple and more efficient method. This model is easy to realize and achieves very good performance.
5. **Gradient-Boosted Tree:** Gradient boosting is used in regression and classification tasks. It gives a prediction model in the form of an ensemble of weak prediction models, and then can together make a more accurate predictor.

Performance Measures Criteria

For our model, we decided to apply three 3 appropriate criteria of accuracy rate: (1) Precision Score, Recall, F1 score, (2) Accuracy, (3) AUC score.

(1) Note that the confusion matrix provides the value of TP, FP, FN, TN, the value of recall = TP/ (TP +FN) and the value of precision = TP/ (TP + FP). The F1 score is the harmonic mean of precision and recall. Whereas the regular mean treats all values equally, the harmonic mean gives much more weight to low values. As a result, the classifier will only get a high F1 score if both recall and precision are high. F1 = TP/(TP+(FN+FP/2)).

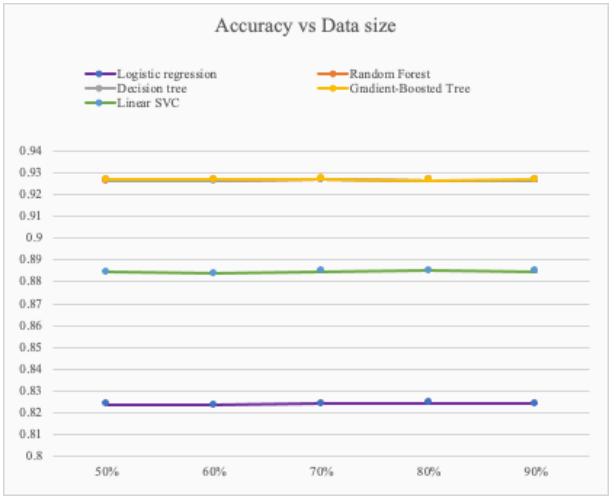
(2) Accuracy represents the number of correctly classified data instances over the total number of data instances. Accuracy = (TN+TP)/ (TN+FP+TP+FN).

(3) AUC is the area under the ROC curve. It tells how much the model is capable of distinguishing between classes. Higher the AUC, the better the model is at predicting 0 classes as 0 and 1 classes as 1. The more convex to the upper left is the curve, the better is the accuracy.

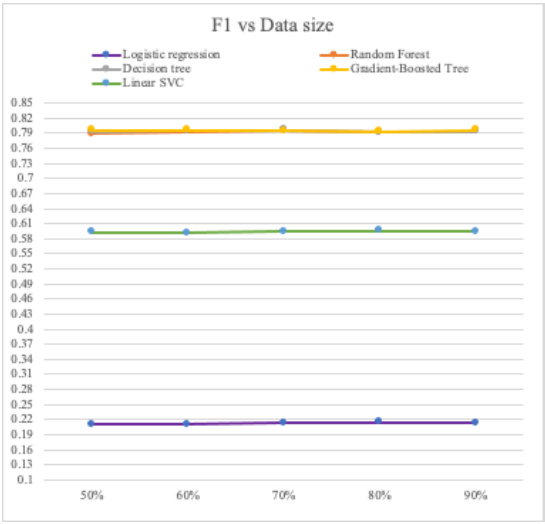
Evaluation

Measuring the impact of scale on quality of analysis

After applying the five Machine Learning Algorithms using Pyspark code on Jupyter notebook and Databrick, we got the value of evaluation measures for every algorithm on different data size. Here we separately compared F1, AUC score and accuracy under different data size. The results are presented in Fig. 3, Fig. 4 and Fig. 5.



From Fig. 3, Fig. 4 and Fig. 5 we can obtain that there is no significant change for F1, AUC score and accuracy as the data size increases. That may because the amount of our dataset is large enough, that make the data split equal or more than 50% can get a good prediction. However, Gradient-Boosted Tree performed best in all the three criteria with above 90% accuracy and AUC score. Linear SVC performed worse in F1 and accuracy compared with other four model, but still with above 90% accuracy. Decision tree performed worse in AUC score, but except data size of 60%, the rest are still with more than 80% AUC score.



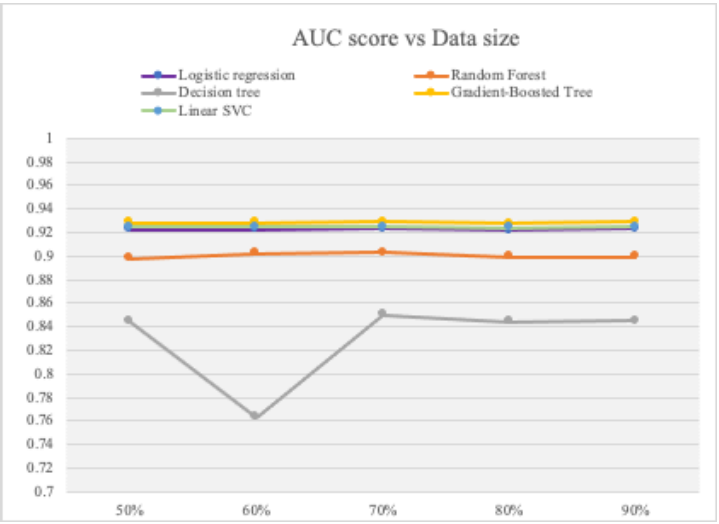


Fig. 3 F1 vs Data size (last page bottom left) Fig. 4 AUC score vs Data size (last page bottom right)

Fig. 5 Accuracy vs Data size (last page top right)

Measuring the impact of scale on performance

From Fig. 6, we can find that the running time for all the five models go down at scale 70%, then go up at scale 80%, finally, go down at scale 90%. From scale 50-70% Decision Tree has lowest running time under 60 sec. Random Forest has the highest running time above 78 sec. From scale 70-90% Logistic Regression, Linear support vector machine has the lowest running time around 60 sec.Gradient-Boosted Tree has the highest running time around 130 sec.

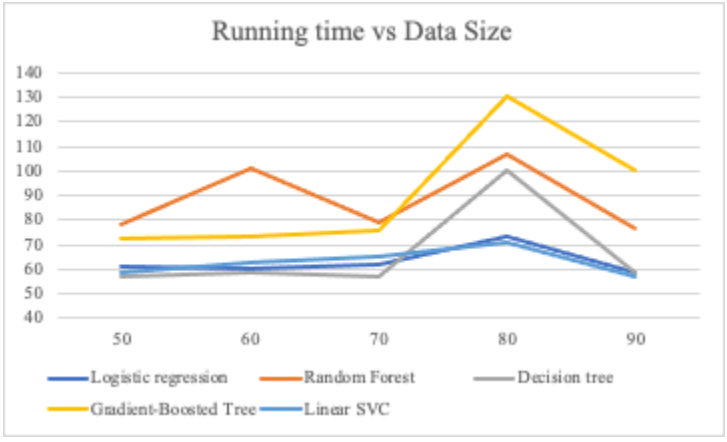
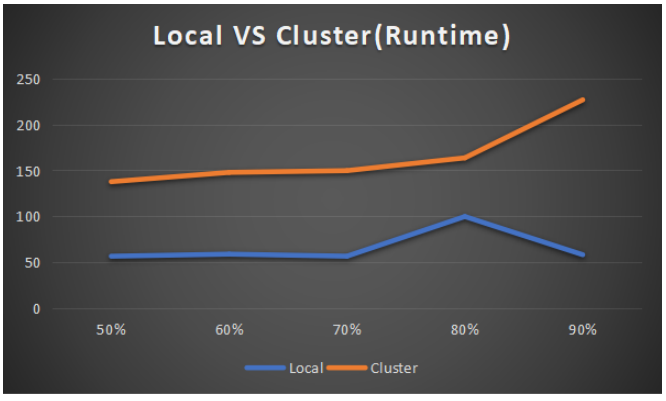


Fig. 6 Running vs Data size (the right side)

Measuring the impact of parallel computation and Local computation

Here we used Databricks with a runtime version of 10.4 LTS, 15.3 memory, 2 cores and 1 DBU(DBU, also known as Databricks Unit, is a unit of processing capability per hour, billed on a per second usage). Since we only used the community edition of Databricks, we can only create one cluster with these configurations listed above.



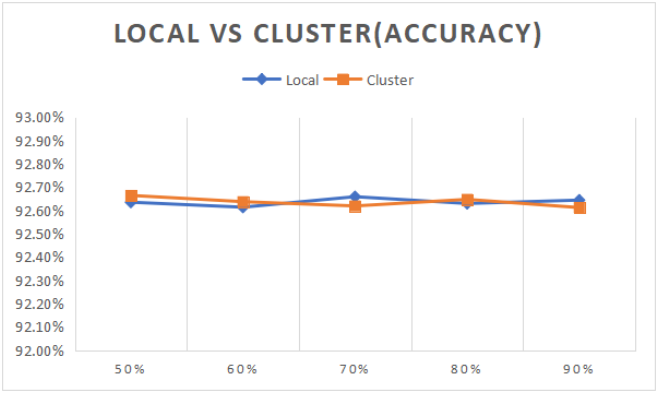


Fig. 7&8 Local vs Cluster(time) & Local vs Cluster(accuracy) (last page bottom left & right)

We will use Decision tree algorithm as an example to compare the runtime and accuracy of local and cluster. We can see that with the smaller the train size, the faster the computation. Unlike local where 80% train set has the logest runtime. Local computation runs faster than Databrick, about 100 seconds, because the community edition only has 2 cores and 1 DBU. If we double those configurations, I'm sure the run time will also be reduced by double and run faster than the local machine. We also evaluate accuracy with different size of train set on local and cluster. The difference is within 0.03%, so there aren't any different between local and cluster in accuracy. Cluster reach its highest accuracy when it is 50% train set, while local is 70% train set.

Performance under the best Data size

After comparing all the model under the different training dataset, we obtain that for most model when the data size is 70%, the performance is better than the others. Below Table 2 show the performance of each model when the training data size is 70% of the data. From Table 2, we can see that the algorithm we choose performed really well, mostly above 90% accuracy, and Logistic Regression is 82% accuracy. As for AUC, Logistic Regression and Gradient-Boosted Tree managed to reach above 90%. Based on the table below and its figures, I would say that Gradient-Boosted Tree is the best among those five algorithms, with its highest accuracy and AUC, and top-of-the-line Recall and F1 score.

Table. 2 performance of each model

Algorithm	Logistic regression	Random Forest	Decision tree	Gradient-Boosted Tree	Linear SVC
F1	0.212759	0.796962	0.796962	0.795526	0.593837
Accuracy	0.824367	0.926587	0.926587	0.927112	0.884628
AUC	0.923053	0.902662	0.850132	0.928777	0.924758

Conclusion

In conclusion, Gradient-Boosted Tree is the best model for our prediction. Because it has the best performance on all of three criteria and the running time is low when the scale is 70%. We found among the 5 models the Decision tree is the one with lowest AUC score. Logistic regression is the one with the lowest F1 score and accuracy. Random forest also has a very good performance on the three criteria. Linear SVC has a good AUC score but lower F1 score and accuracy. After comparing all the models under the different training dataset, most models perform well when the data size is 70%. The running time of Gradient-Boosted Tree and Random forest are longer than the other model.

Furthermore, to better predict flight delay, we might add more features in the future. For instance, for the airport situation, instead of airport id, airport longitude and latitude, volume of flight in the airport and landing difficulties for the airport. On the other hand, for the flight, we might able to get pilot behavior factors(eg. Previous delay/on-time records) and aircraft information. To increase the efficiency of our code, we might enhance the performance of parallel computing in the future to sort through data sets faster and save time and money. The main purpose of parallel processing is to reduce overall processing time by executing the tasks across multiple nodes or processors. And with 4 or 8 cores CPU and larger cluster size, the web-based computation should run a lot faster than the local machine.

Reference

[1]Aurélien Géron. (2019), “Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow”, O’Reilly Media.

[2]Kaggle. Machine Learning with Spark.3 May 2022, https://www.kaggle.com/code/ashenafigurmu/machine-learning-with-spark/notebook

[3]Susan Li. Machine Learning with PySpark and MLlib — Solving a Binary Classification Problem. 4 May 2022, https://towardsdatascience.com/machine-learning-with-pyspark-and-mllib-solving-a-binary-classification-problem-96396065d2aa

[4]Matteo Migliorini. Demonstration of ML Training using Apache Spark MLlib-Random forest and Gradient-boosted trees. 4 May 2022, https://swan-gallery.web.cern.ch/notebooks/apache_spark1/ML_Spark_MLlib.html

[5]Alice Sternberg, Jorge Soares, Diego Carvalho, Eduardo Ogasawara. A Review on Flight Delay Prediction. Arxiv, 2021.

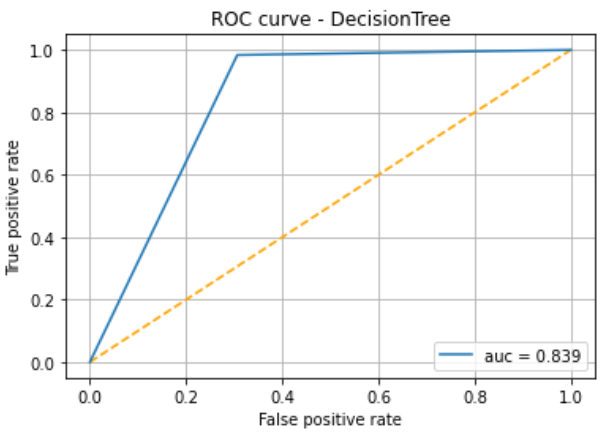
[6]By: IBM Cloud Education. (n.d.). *What is Random Forest?* IBM. Retrieved May 6, 2022, from https://www.ibm.com/cloud/learn/random-forest

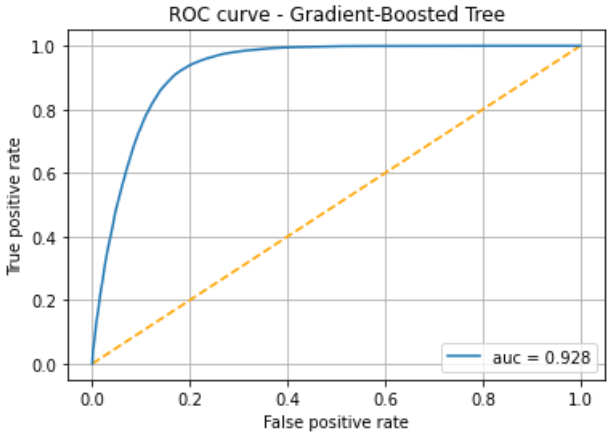
[7]*1.4. Support Vector Machines*. (2022). Scikit-Learn. https://scikit-learn.org/stable/modules/svm.html

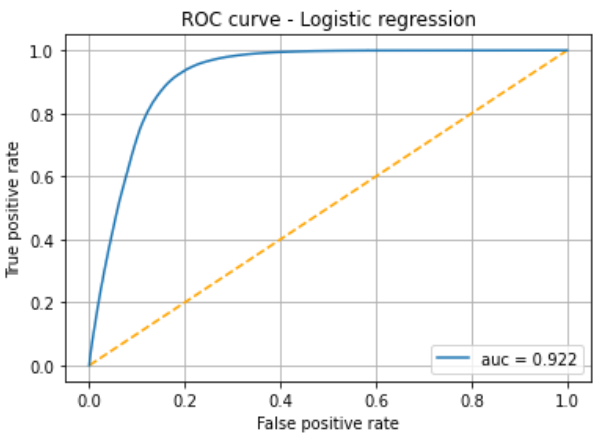
[8]*1.4. Support Vector Machines*. (2022). Scikit-Learn. https://scikit-learn.org/stable/modules/svm.html

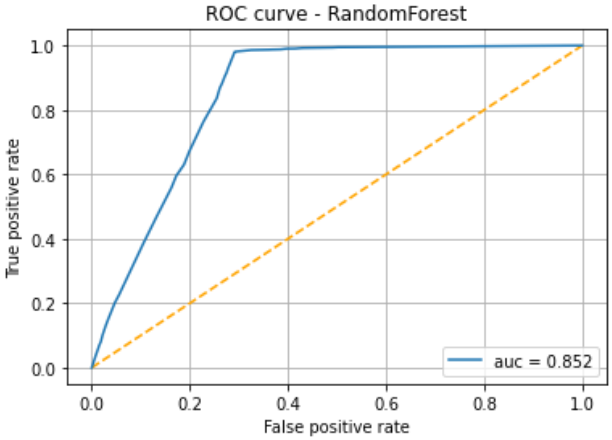
[9]Wikipedia contributors. (2022b, April 15). *Decision tree*. Wikipedia. https://en.wikipedia.org/wiki/Decision_tree#:%7E:text=A%20decision%20tree%20is%20a,only%20contains%20conditional%20control%20statements.

Appendix


ROC curve - DecisionTree
True positive rate
False positive rate
auc = 0.839


ROC curve - Gradient-Boosted Tree
True positive rate
False positive rate
auc = 0.928


ROC curve - Logistic regression
True positive rate
False positive rate
auc = 0.922


ROC curve - RandomForest
True positive rate
False positive rate
auc = 0.852

```
# File location and type
file_location = "/FileStore/tables/flights.csv"
file_type = "csv"

# CSV options
infer_schema = "false"
first_row_is_header = "true"
delimiter = ","

flight = spark.read.csv(file_location, inferSchema=True, header=True)

display(flight)
```

▸ (3) Spark Jobs

	DayofMonth	DayOfWeek	Carrier	OriginAirportID	DestAirportID	DepDelay	ArrDelay
1	19	5	DL	11433	13303	-3	1
2	19	5	DL	14869	12478	0	-8
3	19	5	DL	14057	14869	-4	-15
4	19	5	DL	15016	11433	28	24
5	19	5	DL	11193	12892	-6	-11
6	19	5	DL	10397	15016	-1	-19
7	19	5	DL	15016	10397	0	-1

Truncated results, showing first 1000 rows.

Click to re-execute with maximum result limits.

Command took 37.41 seconds -- by xwang200@stevens.edu at 2022/5/7 09:17:50 on fast