2022 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT) | 978-1-6654-6030-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/COMNETSAT56033.2022.9994427

A Practical Real-Time Flight Delay Prediction System using Big Data Technology

Minh-Tri Vo
Faculty of Information Science and Engineering.
University of Information Technology.
Ho Chi Minh City National University.
Ho Chi Minh city, Vietnam.
trivm851@gmail.com

Trieu-Vu Tran
Faculty of Information Science and Engineering.
University of Information Technology.
Ho Chi Minh City National University.
Ho Chi Minh city, Vietnam.
nitsvutt@gmail.com

Duc-The Pham
Faculty of Information Science and Engineering.
University of Information Technology.
Ho Chi Minh City National University.
Ho Chi Minh city, Vietnam.
phamthect2001@gmail.com

Trong-Hop Do
Faculty of Information Science and Engineering.
University of Information Technology.
Ho Chi Minh City National University.
Ho Chi Minh city, Vietnam.
hopdt@uit.edu.vn

***Abstract*—Flight delay is an unexpected incident in the field of aviation in particular and transportation in general. Predicting the possibility or delay of flights plays a vital role in proactively arranging a time for the airline as well as increasing the reputation of the airline among users. This study presents an implementation of a real-time flight delay prediction system. To ensure the practicality, the entire system is built using big data technology. Apache Kafka is used to stream the flight data to trained machine learning models integrated inside Apache Spark to output real-time prediction results, which will be displayed through a dashboard and stored in Cassandra database simultaneously. Consequently, the system can process a huge amount of input data and produce prediction results in real-time.**

***Index Terms*—Flight Delay Prediction, Machine Learning, Spark, Kafka, Streaming, Cassandra.**

I. Introduction

One of the worst experiences of flying is delays or even cancellations. To be fair, about 80% of flights are on time - that is, within 15 minutes of the scheduled time. The remaining 20% is often delayed. Indeed, the delay makes any of us frustrated because we must wait a long time and miss many planned plans. There are many reasons why a flight departs late compared to the originally scheduled time, such as late arrival, weather problems, security issues, technical problems, etc.,. The delay also takes a toll on airlines' business strategies as they rely on customer trust and return to support their loyal customers, and users will be affected by its reliable performance to users.

Flight delay prediction is a problem based on analyzing information provided before departure to extract useful information and then using mathematical models to predict flight delays. The problem is researched to help airlines predict the possibility and delay of a flight to be ready to handle special situations, which is to notify customers so that they can be flexible in their own time, helping to increase the user experience and compete with other airlines.

This study focusses on building a practical flight delay prediction system. The practicality requires the system to be capable of processing large amount of data and produce the prediction results in real-time. To this end, the entire real-time flight delay prediction system in this study is built using big data technology. The flight data is streamed via Kafka to a trained machine learning model integrated in Spark, which is a powerful big data framework capable of processing huge amount of data and produce results in real-time. The real-time prediction results are displayed through dashboard and stored in Cassandra database.

The structure of the paper is as follow. We present related works on this topic in section II. The process of constructing flight dataset and dataset analysis are presented in Section III. The approach for building the real-time flight delay prediction system is described in detail in Section IV. In Section V, We proceed to build an experimental pipeline of Machine Learning models on the dataset and analyze the results to find the best model and its limitations. Next, we proceed to build a timed flight delay backup simulation system implemented in Section VI. Finally, we draw conclusions in Section VII.

II. Related works

[9] This paper provides valuable research in the field of flight delay analysis in places like US, Asia, Brazil, Europe. The analysis on many aspects of data

such as planning (filght plan, Airline schedule, Airport Schedule), temporal (Season, month, day of week, time of day), weather (Visibility, Ceiling, Convective weather, Surface weather), spatial (Airport, City, Region), operations (Capacity, Demand), features (Airline Status, Airport infrastructure, Aircraft model, Aircraft occupancy, Fares, Frequency), state of the system (Prior delay levels, Operational conditions) makes the data more grounded and the indications of the problem gradually reveal. Plus analysis of different methodologies such as machine learning, Operational Research, Network Representation, Probabilistic Models, Statistical Analysis. The difficulty that the author mentions is that the data is getting bigger and bigger, and the challenge for data scientists to study airlines and airports will require a combination of in-domain knowledge and scientific skills data learning to explore the trove of flight big data.

With the knowledge of Bigdata, we will build a real-time simulation system for flight delay prediction so that more research can be done to help deploy other Bigdata systems into real applications economy is more specific.

III. Flight Dataset Construction and Analysis

A. Flight Dataset

The dataset used in the flight delay prediction problem is collected from website Bureau of Transportation Statistics (BTS). BTS is a part of Department of Transportation (DOT) is the preeminent source of statistics on commercial aviation, intermodal freight operations, and transport economics, and provides context for planners and the public to understand traffic statistics carriage. BTS provides data information of flights from 1987 to present. Flight information data is provided on a monthly basis with an average of nearly half a million flight information a month. Because the amount of data is quite large and hardware is limited, we only use a few months representing the quarters of the year to generate the model training dataset. We use flight information data of 6/9/12–2021, 3/2022 for model training and flight information data April 2022 to stream emulated real-time for the system. We have collected the data of the above 4 months and obtained a dataset containing 2,200,913 flight information data and many flights have missing values. Since the dataset has missing values, we perform preprocessing steps to handle missing values. For flights with missing values, we will remove that flight's information. Feature DEP_DELAY used by us as the output for the problem, it contains continuous values that are the difference in minutes between the expected and actual departure times of the flights. To match the original goal and simplify the problem, we normalize feature DEP_DELAY into 3 labels as follows: for flights with actual early departure time or on time will be labeled "0", flights with actual departure time less than 30 minutes late will be labeled "1", remaining flights with actual departure time more than 30 minutes late or subject to cancellation will be labeled "2". Feature DEP_DELAY is also renamed LABEL to fit the data and problem. After the processing is done, we get the dataset *Flight Delays Dataset* contains 2,157,737 data lines and 11 attributes are flight information. Similarly, we create a dataset to stream the real-time emulator for the system containing 542,117 data lines and 11 feature.

TABLE I
Details of properties

Index	Feature	Meaning
0	ID	ID of flight.
1	QUARTER	Quarter of the year (1-4).
2	MONTH	Month of the year (1-12).
3	DAY_OF_MONTH	Day in month (1-31).
4	DAY_OF_WEEK	Day in week (1-7).
5	OP_UNIQUE_CARRIER	Service carrier code.
6	ORIGIN	Departure Airport.
7	DEST	Destination Airport.
8	DISTANCE	Distance between airports (mile).
9	CRS_DEP_TIME	Estimated departure time.
10	LABEL	Output, includes 3 labels: – 0: Not delay. – 1: 30 or less minutes late. – 2: 30 or more minutes late or cancel the flight.

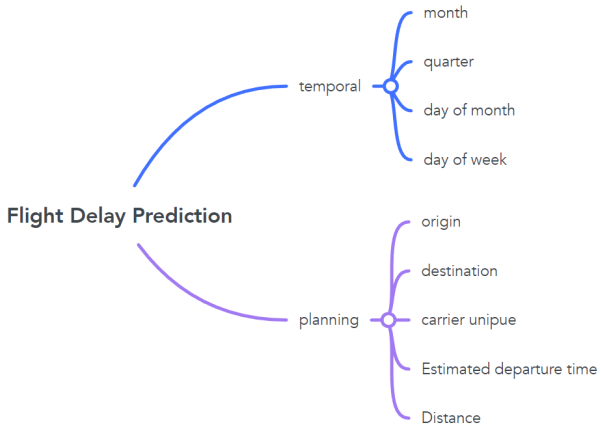

Fig. 1. Data model of the flight delay prediction

We use the properties that favor the planning and temporal branches as shown in Fig. 1, with the desire to use an optimal data set to bring out the full potential of the whole system.

Details of the attributes of the dataset are shown in Table I.

B. Dataset Analysis

Fig. 2 shows the distribution and proportions of the three labels in the train/test set of our Flight Delays Dataset. We can see that the distribution of labels on train and test sets is similar with the label ratios 0 - 1 - 2 respectively 61.5% – 25.6% – 12.9%. Our datasets

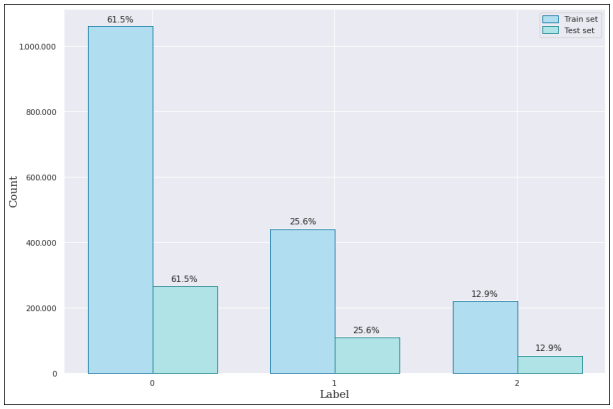

Fig. 2. Statistics of Train/Test set.

tend to have data imbalances when the number of labels 0 occupies 61.5% of the dataset, 2.4 times more than label 1 (25.6%) and 4.8 times more than label 2 (12.9%). This is also a challenge of our dataset.

IV. Real-time flight delay prediction system

From the Flight Delays Dataset we created in Section III, we perform data normalization steps before encoding data as String Indexer, One Hot Encoding, Vector Assembler, Data Standardization (are covered in more detail in Part IV-A). To create a flight delay prediction model, we proceed to build models *Machine Learning* as Logistic Regression, Decision Tree Classifier, Random Forest Classifier, Naive Bayes supported by Machine Learning Library (MLlib) is a machine learning library of Spark. To evaluate the model, we use 4 degrees of measurement: Precision, Recall, Accuracy and F1–score (Part details IV-C).From the model evaluation, we will select the best and most viable model to proceed with building our flight delay prediction real–time simulation system.

A. Data Normalization

Data Normalization is a general concept that refers to the act of converting the original values of a data set into new values. The new values are usually encoded relative to the data set itself and scaled in some way.

1) String Indexer: Encode a label string column into a label index column. String Indexer can encode multiple columns. If the input column is numeric, it will be cast to string type and index the string's values. Indicators are in [0, numLabels). By default, this is sorted by label frequency so the most frequent label gets index 0.

2) One Hot Encoding: Mapping a categorical feature, expressed as an index label into a binary vector with at most one unique value indicating the presence of a particular feature value among all the feature values. This encoding allows algorithms to expect continuous features such as Logistic Regression using categorical features. For string input, it is common to encode the categorical features using the String Indexer first. OneHotEncoder can convert multiple columns, returning a one-hot-encoded output vector column for each input column. Usually, merge these vectors into a single feature vector using Vector Assembler.

3) Vector Assembler: Is a transformation that combines a given list of columns into a single vector column. It is useful to combine raw features and features generated by different feature transformations into a single feature vector, to train ML models like Logistic Regression and Decision Trees. Vector Assembler accepts the following input column types: all numeric, boolean and vector types. In each row, the values of the input columns will be concatenated into a vector in the specified order.

4) Data Standardization: It's a specific type of normalization technique. It is sometimes called z-score normalization. z-score is also known as standard score, it's the transformed value for each data point. To normalize the dataset using standardization, we take all x_i value inside the data set and transform it to the corresponding z_i value using (1):

$$z_i = \frac{x_i - \mu_k}{\sigma_k} \tag{1}$$

Where μ_k and σ_k are the mean and standard deviation of each k .th attribute, respectively. This standardization converts the mean of the data set to 0 and its standard deviation to 1.

B. Machine Learning Algorithm

1) Logistic Regression: Logistic Regression (LR) is a process of modeling the probability of a discrete outcome for an input variable. This is a special case of Generalized Linear models which is used to predict the probability of the outcomes. In logistic regression spark.ml, binomial logistic regression can be used for binary classification results using. To predict multiple class, multinomial logistic regression can be used [1].

Multiclass classification supported through polynomial logistic regression (softmax). In polynomial logistic regression, the algorithm generates **K** sets of coefficients or matrices of size **K** × **J** where **K** is the number of result classes and **J** is the number of features. If the algorithm fits a term *intercept* then the length **K** vector of the intercepts is available.

TABLE II
Confusion matrix

		Predicted annotation	
		Positive	Negative
Gold annotation	Positive	True Positive	False Negative
	Negative	False Positive	True Negative

The conditional probability of the resulting classes $k \in 1, 2 \ldots,$ **K** is modeled using the softmax function shown in (2).

$$P(Y = K|\mathbf{X}, \beta_k, \beta_{0k}) = \frac{e^{\beta_k \cdot \mathbf{X} + \beta_{0k}}}{\sum_{k'=0}^{K-1} e^{\beta_{k'} \cdot \mathbf{X} + \beta_{0k'}}} \quad (2)$$

2) Decision Tree Classifier: Decision Tree (DT) is a popular method of classification and regression methods. Decision trees are widely used because they are easy to interpret, handle categorical features, extend to multiclass classification implementations, do not require object scaling, and can capture object interactions and nonlinear.

A decision tree is a greedy algorithm that performs recursive binary partitioning of the feature space. The tree predicts the same label for each bottom (leaf) partition. Each partition is chosen greedily by choosing the best split from a set of possible splits, to maximize information gain at a tree node. In other words, the split is selected at each tree node selected from the set as shown in (3):

$$\arg\max_x IG(D, s) \quad (3)$$

Where $IG(D, s)$ is the information gain when a division s is applied to the data set D.

3) Random Forest Classifier: Random Forest (RF) is a popular group of classification and regression methods. Random forest combines many decision trees, but each decision tree will be different (with random factor). The prediction results are then aggregated from the decision trees, the output's label is the most predicted (majority) label of all decision trees to reduce the risk of overfitting. The spark.ml implementation supports random forests for binary and multiclass classification and for regression, using both numerical and categorical features.

4) Naive Bayes: Naive Bayes (NB) is a classification algorithm based on computing probabilities applying the Bayes theorem. This algorithm belongs to the group supervised learning. This is the classification approach according to the probabilistic model. Predict the probability that a new object belongs to a member of the class in question. Naive Bayes can be trained very effectively. With one training over the data, it computes the conditional probability distribution of each feature for each label. For prediction, it applies Bayes theorem to calculate the conditional probability distribution of each label for an observation.

According to Bayes theorem, we have (4) for calculating the random probability of event Y when X is known as follows:

$$P(y|\mathbf{x}) = \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})} \quad (4)$$

Suppose we split an event **x** into n different components $x_1, x_2, \ldots, x_n$. Naive Bayes, as the name suggests, relies on the naive assumption that $x_1, x_2, \ldots, x_n$ are independent components. From there we can calculate as in (5):

$$P(\mathbf{x}|y) = P(x_1|y)P(x_2|y)\ldots P(x_n|y) \quad (5)$$

Hence we have (6):

$$P(\mathbf{x}|y) \propto P(y)\prod_{i=1}^{n} P(x_i|y) \quad (6)$$

In practice, it is rare to find data whose components are completely independent of each other. However, this assumption makes the calculation simple, the training data fast, and brings unexpected results with certain classes of problems.

C. Evaluation Metrics

1) Precision & Recall: Two metrics are used to measure the performance of a retrieval system: *precision* and *recall*.Precision is defined as the ratio of correctly classified positive samples (True Positive) to a total number of classified positive samples, see 7. The recall is calculated as the ratio between the numbers of Positive samples correctly classified as Positive to the total number of Positive samples. The recall measures the model's ability to detect positive samples. The higher the recall, the more positive samples detected as shown in (7) and (8) [2].

$$Precision : P = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (7)$$

$$Recall : R = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (8)$$

2) F1–score: F–score is defined as the weighted mean of precision and recall depending on the weight function β, see (9). F1–score means the harmonized mean between precision and recall, see (10), when it is written as F–score, it usually means F1–score. F–score is also known as F–Measure. F1–score can have different metrics that give different weights for precision and recall [2].

$$F - score : F_\beta = (1 + \beta^2) \times \frac{P \times R}{\beta^2 \times P + R} \quad (9)$$

For β = 1, the standard F–score is obtained, see Equation 9.

$$F - score : F_1 = F = 2 \times \frac{P \times R}{P + R} \quad (10)$$

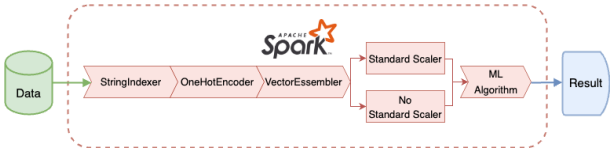

Fig. 3. Spark ML pipeline.

3) Accuracy: Another metric defined as the proportion of true cases retrieved, both positive and negative, out of all retrieved cases. Accuracy is a weighted average of inverse precision and precision. The higher the Accuracy, the more efficient the model. However, this is not true for problems with unbalanced datasets, see (11) [2].

$$Accuracy : A = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \qquad (11)$$

D. BigData Frameworks

Nowadays more and more data is generated by indirect or direct ways. Along with the development of technology devices, we can increasingly collect data in many different ways, typically from social networks, IoT devices in daily life silently collecting data. ours makes the data more and more bloated. Big data now has a lot of different applications, from banking and securities to media or education, but how to use big data is an issue that traditional tools and techniques cannot can handle. This is why some frameworks are born to work with big data. One of the powerful frameworks is Apache Spark which we use for this problem: flight delay prediction.

Apache Spark is a large-scale open source data processing framework which has huge computing power and is easy to handle big data. Spark also provides many easy-to-use APIs to reduce the burden on developers in distributed computing or big data processing.

V. Experiments and Results

We build an experimental pipeline of Machine Learning (ML) models: Logistic Regression (LR), Decision Tree (DT), Random Forest (RF), Naive Bayes (NB) using Spark ML Framework (Fig. 3).

To avoid the randomness of the experimental results, we performed 5 times of training and recorded the results, then averaged the results. Table III presents the performance comparison results of ML datasets on training and testing sets when the data normalization method is not used and the z-score normalization method is used. Because the dataset tends to have data imbalance, to compare the performance of the models we rely on the F1-macro measure to evaluate. The LR model gives the same results when data normalization is not applied and data normalization is applied using z-score, the results on the training and testing sets are quite similar (F1-macro train = 36.23% and F1-macro test = 36.25%). DT model when not applying data normalization gives higher results when applying data normalization using z-score (higher than 1.7% in training set and 0.69% in testing set), The difference between training and testing sets is quite large, from 2.67% – 3.77%. RF model when not applying data normalization gives lower results when applying data normalization by z-score (lower than 1.52% in training set and 1.54% in testing set), the result difference between training and testing set is not significant, from 0.01% – 0.03%. Although the RF model is trained on all three labels "0", "1" and "2", the model cannot predict the label "2" of the dataset, i.e. is the prediction result of the model labeled only "0" and "1" in both training and testing sets. The NB model gives superior results when applying the z-score data normalization method (higher than 13.92% in the training set and 13.64% in the testing set), the result difference between training and testing set is not significant, from 0.03% – 0.25%. In summary, the DT model does not apply data normalization for the best results (F1-macro test = 45.27%), followed by the NB model with z-score normalization applied (F1-macro test = 38.50) and the RF model which was the worst model (F1-macro test = 27.14% and could not predict the labels "2").

Detailed results on each label of the DT model without data normalization (best model) are shown in Table IV. The results are descending in order of label "0", label "1", label "2". Only one label has a high F1-score above 70% (label "0" – 78.60%), and the other two labels have a relatively low F1-score (label "1" – 30.22% and the label "2" – 26.99%). This result explains the data imbalance when the number of labels "0" accounts for 61.5% (labels "2" accounts for only 12.9% in the Flight Delays Dataset, detailed in Fig. 2).

VI. System Architecture

The system architecture is divided into three main phases, shown in Fig. 4. Phase 1 (blue) synthesizes, processes data and builds machine learning models for the system, phase 2 (red) streams data to predict flight delays, phase 3 (green) outputs results to the screen and model update.

A. Model Selection Building Phase

In fact, data is aggregated in many places and for Bigdata, the data is more distributed, so the application of Bigdata processing tools like Kafka to create a data stream for the system to operate stably and efficiently high.

In the initial phase, the aim is to use machine learning methods supported by spark to create the best model for this system. Through steps such as processing data IV-A, encoding columns and then conducting training using many different methods.

TABLE III
AVERAGE RESULTS OVER 5 TRAINING

	Scale = None				Scale = StandardScaler			
	Train set		Test set		Train set		Test set	
	Accuracy	F1-macro	Accuracy	F1-macro	Accuracy	F1-macro	Accuracy	F1-macro
LR	63.34	36.23	63.31	36.25	63.34	36.23	63.31	36.25
DT	**67.24**	**49.04**	**64.95**	**45.27**	**66.56**	**47.34**	**64.86**	**44.58**
RF	61.53	25.63	61.49	25.60	61.90	27.15	61.86	27.14
NB	24.60	24.83	24.64	24.86	44.96	38.75	44.73	38.50

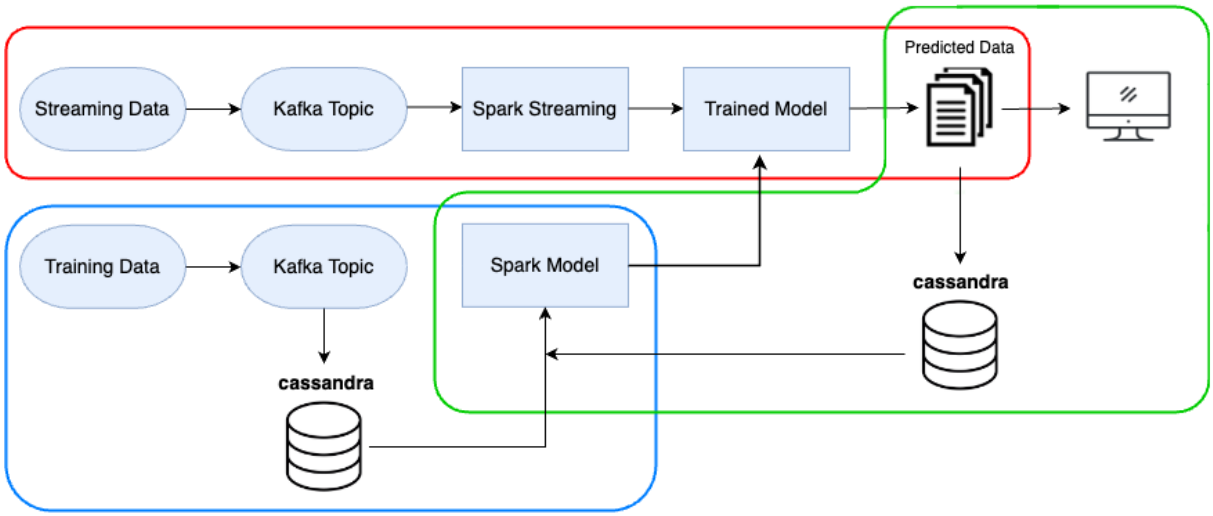

Fig. 4. General architecture of the Flight Delay Prediction system.

TABLE IV
THE RESULTS PER CLASS OF THE BEST MODEL ON TEST SET

LABEL	Precision	Recall	F1-score
0	68.32	92.52	78.60
1	47.30	22.20	30.22
2	50.08	18.47	26.99

Fig. 5. Model Selection Building Phase.

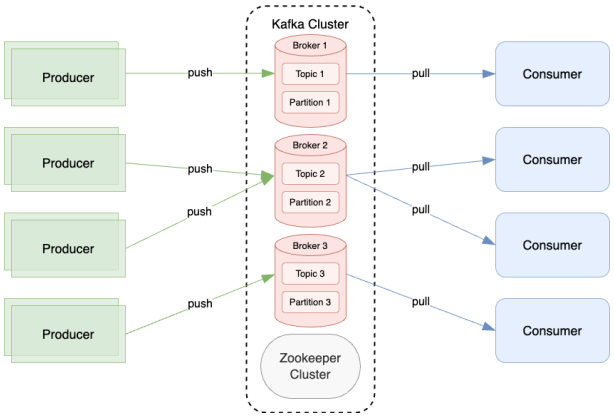

Fig. 6. Kafka's communication system.

Training data compiled from many different sources will be sent by Kafka Producer to Kafka topic. Consumer then takes on the role of data transfer and transfers data directly to the Cassandra database for storage. Next, the data is loaded from Cassandra to train the models and choose the best model to use as the main model of this system. Fig. 5 fully shows the data in small steps. The steps of data processing, model training experiments in Section V show that the Decision Tree model has the best results, so we will use the Decision Tree model for this system.

1) Apache Kafka [5]: Kafka can be understood simply as a tool for streaming distributed data (data from many different sources). Specifically, Kafka is an open source, fully packaged, fault tolerant and instant messaging system. It is because of its reliability that Kafka is gradually being replaced by the traditional messaging system. It is used for common messaging systems in different contexts. This is a consequence when horizontal scalability and reliable data transfer are the most important requirements. Some Kafka use cases: Stream Processing, Log Aggregation, Website Activity Monitoring, Metrics Collection.

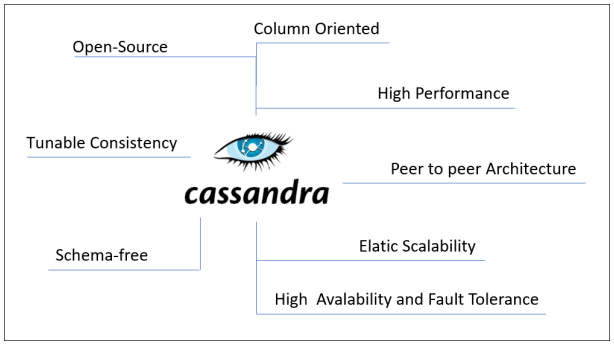

Fig. 7. Overview about Apache Cassandra.

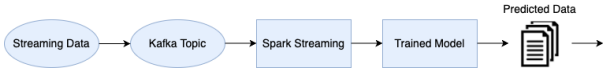

Fig. 8. Data Stream Phase.

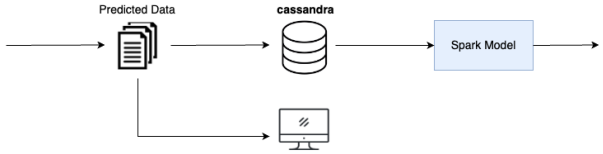

Fig. 9. Update the model and display the results.

Departures — Time: 10:06:44

ID	ORIGIN	DEST	CRS_DEP_TIME	PREDICTION
4/6/2022-OH-RIC-CLT-5475	RIC	CLT	13.0	Not Delay
4/7/2022-OH-RIC-CLT-5475	RIC	CLT	13.0	Not Delay
4/8/2022-OH-RIC-CLT-5475	RIC	CLT	13.0	Not Delay
4/9/2022-OH-RIC-CLT-5475	RIC	CLT	13.0	Not Delay
4/10/2022-OH-RIC-CLT-5475	RIC	CLT	13.0	Not Delay
4/11/2022-OH-RIC-CLT-5475	RIC	CLT	13.0	Not Delay
4/12/2022-OH-RIC-CLT-5475	RIC	CLT	13.0	Not Delay
4/13/2022-OH-RIC-CLT-5475	RIC	CLT	13.0	Not Delay
4/14/2022-OH-RIC-CLT-5475	RIC	CLT	13.0	Not Delay

Fig. 10. Dashboard displaying real-time prediction results

Kafka's structure: based on Fig. 6, it can be seen that: Producer has the role of a message sender, the message will be included in a specific topic. Consumer acts as a message receiver, messages to be passed from topic to customer to make subsequent requests from Kafka users. Topic is a feed name where the data from the Producer is stored. Partitions are paragraphs divided by topics. Broker: A Kafka cluster is a collection of servers, each of which is called a broker. ZooKeeper is used to manage and arrange Kafka Brokers.

2) Apache Cassandra [4]: Speaking of NoSQL middleware leading the way are Google BigTable and Amazon Dynamo, but many others also appear in the open source world. Among them, Apache Cassandra (Cassandra) is attracting special attention recently.

Cassandra was originally created by Facebook (now renamed Meta). It was then donated to the Apache Foundation in February 2010 and upgraded to the flagship Apache project. Cassandra is a distributed database that combines the data model of Google Bigtable with a distributed system design like the Amazon Dynamo clone. Fig. 7 shows characteristics of cassandra.

The superior features of Cassandra include the following points: Suitable for real–world use, highly fault tolerant, SPOF (Single point of failure)–free architecture, consistent degree of control freedom, rich data model, Fast Linear-scale Performance, Cassandra increase throughput because they facilitate us to increase the number of nodes in the cluster. As a result, Cassandra maintains fast response times. High availability, support for different languages as client code, easy to capture the internal state of the server using JMX (Java Management Extensions).

Because Bigdata has very large volume it becomes difficult to store and manage data, thanks to the outstanding features of Cassandra we use it to manage this big data. Our purpose of using Cassandra in this system is also because we want to take advantage of multi–node capabilities, when two or more machines contain the required data, we can access P2P (peer–to–peer) to query data. back and forth, from which it is possible to get data from many different nodes, that is the nature of big data because it has a very large volume, so the storage capacity is highly distributed, data is difficult to concentrate in 1 node.

B. Streaming Data Phase

In the second phase, with the model for the system in place at the previous phase, the next step is to start the system from the first step of data transfer. Real–time data is the key to the system being put into practice, and it is also a challenge for data scientists and experts in the field.

Kafka will continue to do its job, which is to put real–time data into the system. This process is similar to how the training data was fed into the database in the previous stage. However, this time the data will be put into Kafka Topic and temporarily stored to wait until Spark Streaming uses it.

The purpose of combining Kafka and Spark Streaming is you can ensure minimal data loss through Spark Streaming while synchronously saving all received Kafka data for easy recovery, read data from single topic or multiple topics quickly. Using Spark and Kafka in parallel can also achieve high scalability, throughput and fault tolerance, and a host of other benefits.

Based on Fig. 8, it can be seen that using Spark Streaming to load streaming data directly from the corresponding Kafka Topic to process and predict the flight delay immediately. Here we use Kafka in combination

with Spark Streaming to model the speed of receiving messages and continuous processing.

Spark Streaming helps process real–time data from various sources such as Kafka, Flume, and Amazon Kinesis. After processing, this data can be pushed out to the file system or to a database. Spark Streaming is used for real–time data that can be unstructured data like images or text.

C. Model Update Phase

At this phase, we update the model with the predicted real–time data with the original label. Accompanied by displaying an intuitive interface for users to have an overview of the system's results. Cassandra has the role of storing and managing all data after processing, with the advantages mentioned in part VI-A2, if the system operates on many different computers (nodes), the data will be distributed so Cassandra will be the key tool for the system to work easily. The predicted data will be saved to Cassandra to update the model again. At the same time, prediction data will also be displayed on the application screen so that users can visually see the results. Fig. 9 simulates the architecture at this phase. Since the system depicts the flight status screen at the airport, we have set the streaming time to 5 minutes for a data stream and 6 hours for a model update.

We used Python's Skinter library to design an interface that displays the results as a list of flights when the prediction is complete. The interface shown in Fig. 10 is inspired by the flight status display at the airport and reproduced by us.

VII. Conclusion

This study presents a practical real-time flight delay prediction system based on big data technology. Compared to other studies, which focus on building the prediction model, this paper concerns the practicality of the system, which requires the capability of processing a huge amount of input data and producing prediction results in real-time. To this end, the entire system in this paper is built using big data technology. Apache Kafka is used to stream the flight data to trained machine learning models integrated inside Apache Spark, which is a powerful big data processing framework. The real-time prediction results are displayed on a dashboard and also stored in Cassandra database at the same time.

Acknowledgment

This research was supported by The VNUHCM-University of Information Technology's Scientific Research Support Fund.

References

[1] Singh, P., "Logistic Regression. In Machine Learning with PySpark," Apress, Berkeley, CA, pp. 65-98, 2019.

[2] Dalianis, H., "Evaluation metrics and evaluation. In Clinical text mining," Springer, Cham, pp. 45-53, 2018.

[3] Zhang, Z., "Artificial neural network. In Multivariate time series analysis in climate and environmental research," Springer, Cham, pp. 1-35, 2018.

[4] Avinash Lakshman, Prashant Malik, "Cassandra - A Decentralized Structured Storage System," 2010.

[5] Jay Kreps, Neha Narkhede, Jun Rao, "Kafka: a Distributed Messaging System for Log Processing," 2011.

[6] Shelke, M.S., Deshmukh, P.R. and Shandilya, V.K., "A review on imbalanced data handling using undersampling and oversampling technique," Int. J. Recent Trends Eng. Res, 3(4), pp.444-449, 2017.

[7] Wang, S.C., 2003, "Artificial neural network. In Interdisciplinary computing in java programming," Springer, Boston, MA, pp. 81-100, 2003.

[8] Dai, J.J., Ding, D., Shi, D., Huang, S., Wang, J., Qiu, X., Huang, K., Song, G., Wang, Y., Gong, Q. and Song, J., "BigDL 2.0: Seamless Scaling of AI Pipelines from Laptops to Distributed Cluster," In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 21439-21446, 2022.

[9] Alice Sternberg, Jorge Soares, Diego Carvalho and Eduardo Ogasawara, "A Review on Flight Delay Prediction," 2017.