Hotel Reservations Classification: Apache Spark with Deep Learning Algorithm Using BigDL DLlib

Nguyen.T.C.Duc[1,2] and Le.V.Sy[1,3] and Nguyen.T.H.Hung[1,4]

[1] University of Information Technology, Vietnam
[2] 20521199@gm.uit.edu.vn
[3] 20521854@gm.uit.edu.vn
[4] 20521370@gm.uit.edu.vn

Abstract This paper explores the application of deep learning models, specifically Logistic Regression, Deep Neural Network (DNN), and Random Forest, which is in Deep Learning Library (BigDL DLlib), for hotel reservation classification. We analyze a real-world hotel reservation dataset, comparing the performance of these models against traditional machine learning approaches. Leveraging the capabilities of Apache Spark for distributed computing, our approach aims to enhance accuracy and efficiency in predicting booking outcomes, such as cancellations and successful reservations. The results demonstrate the potential of deep learning models to optimize hotel reservation systems, facilitating data-driven decision-making and improving guest experiences.

Keywords: Apache Spark, Deep Learning, BigDL Dllib, Logistic Regression, Deep Neural Network (DNN), and Random Forest.

1. Introduction

The hotel industry grapples with the challenges of effectively managing hotel reservations and providing exceptional guest services amid the burgeoning volume of reservation data. Accurate classification of reservations, distinguishing between successful bookings and potential cancellations, plays a pivotal role in optimizing room occupancy rates and revenue streams for hotels. In this paper, we investigate the application of deep learning models, including Logistic Regression, Deep Neural Network (DNN), and Random Forest, for hotel reservation classification. Each of these models offers unique strengths in handling diverse booking scenarios and predictive analysis in the hospitality sector. We aim to evaluate the effectiveness of these deep learning approaches by comparing their performance against each other and traditional machine learning techniques. Leveraging the capabilities of Apache Spark for distributed computing, our approach seeks to enhance the accuracy and efficiency of predicting reservation outcomes, facilitating data-driven decision-making and elevating guest experiences. With a focus on providing insights into the potential integration of deep learning models into hotel reservation systems, we envision our research to be a steppingstone for optimizing operations and guest satisfaction in the hotel industry. By harnessing the power of big data and deep learning, our study contributes to the advancement of reservation management practices and supports data-driven decision-making for hoteliers. As the hospitality sector continues to embrace technological innovations, our findings can pave the way for a more data-driven and efficient approach to hotel reservation classification, ultimately elevating the overall quality of guest services and maximizing revenue generation.

2. Related Work

Junlong Xiang et al. (2014) [1], they have used Extreme Learning Machine (ELM) algorithm which achieve a relatively high Overall Accuracy and can decrease the time of the training phase. For large data or big they proposed a massively parallel algorithm for ELM, that is MR ELM and is a MapReduce variant of ELM.Their experiment results shows that MR ELM have a high speed performance. In [2], Samuel Marchal et al. (2014) has proposed a

solution to cope large data to analyze for security observing insights. They introduced a security observing architecture of networks for intrusion detection and prevention and forensic analysis. They mined different types of data like honey-pot data. It provides the big data solution in a distributed system. They proposed Data correlation schemes and calculated their work in Hadoop and Spark. They introduced a new scalable NIDS design which collects and stores honeypot data, DNS data. Five big data frameworks were evaluated for security monitoring. After their performance analysis, they found that Shark and Spark were the on top performers in all scenarios and so they were suitable to implement the solution. Sung-Hwan Ahn et al. (2014) [3] author believed that intelligent new threats are growing and earlier unknown attacks cannot be detected using existing pattern matching methods. They anticipated bigdata analysis solution which is a solution for detecting these kinds of unknown attacks. In future works, author suggests the researches which must be done to classify the data by context of intrusion detection. This can lead to implement the data relation study methodology and unusual behavior detection approach. Author is hopeful for quantitative and qualitative calculation of suggested model and performance estimation in future. Kleber M.M. Vieira et al. (2014) [4], they proposed IRAS, an Intrusion Response Autonomic System, using Big Data techniques for data analytics for decision taking. Also, proposed a model for autonomic intrusion detection system based on the autonomic loop, known as MAPE-K (Monitor, Analyze, Plan, Execute and Knowledge Base). MohiuddinSolaimani et al. (2014) [5], they performed experiments on a real-time, Chi-square test based anomaly detection framework using Bigdata tool Apache Spark.

3. Methodologies

In this section, some pre-processing and feature engineering techniques are illustrated. In addition, information about big data platform and some ensemble learning methods are also given in this section.

3.1. Pre-processing techniques

Data cleaning is the process of removing and modifying data that is incorrect, incomplete, irrelevant, duplicated, or improperly formatted. For instance, the format of date and datetime data in a column is not correct, so it should be corrected to only one format. Moreover, attributes that have a value missing rate of more than 90% should be removed from the data set.

Data transformation is the process of changing the format, structure, or values of data. The transformed data is better-organized and easier to use than the raw data. Properly formatted and validated data improves data quality and protects applications from potential landmines such as null values, unexpected duplicates, incorrect indexing, and incompatible formats. Data transformation facilitates compatibility between applications, systems, and types of data. Data used for multiple purposes may need to be transformed in different ways.

Missing values imputation is simply the process of filling in missing data. But it is not easy to find out an appropriate value. Data imputation takes a lot of effort to gain 'just' the acceptable result. There are many techniques to impute the data as using mean, median value or taking the predicted value from a trained model, etc.

Data normalization is a technique that structures the data for storage efficiently. The data is collected from many resources; hence, the range of numeric values is different between the attributes. This may lead to the poor-quality performance of the models built on the data. Subsequently, the data have to be normalized to the same distribution. In the categorical feature, if a large information is stored in just one feature, it is better to split the data into specific features. So that, the data is managed logically, and the storage space is used economically.

Data scaling is a vital data preparation step in machine learning, especially for deep learning models. It involves transforming input feature values to a common scale, typically [0, 1], to ensure fair comparisons and faster training convergence. By using Min-Max scaling, we achieved standardized and normalized data, leading to improved model performance with faster training and better generalization capabilities. The MinMaxScaler successfully handled outliers and contributed to stable model performance, making it a valuable technique for enhancing the accuracy of deep learning predictions.

3.2. Feature Engineering

Feature engineering is a vital process in preparing data for machine learning models, and it involves various techniques to enhance the quality and relevance of features. Three common methods used in feature engineering are OneHotEncoder, StringIndexer, and VectorAssembler [6].

OneHotEncoder: OneHotEncoder is a method used to convert categorical variables into a binary vector representation. It creates new binary columns for each category present in the original categorical feature, with a value of 1 in the corresponding column indicating the presence of that category and 0 otherwise. This transformation enables machine learning algorithms to handle categorical data effectively.

StringIndexer: StringIndexer is another technique used for processing categorical variables. It assigns a unique numerical index to each distinct category in the categorical feature. This numerical representation helps the machine learning algorithms to interpret and process categorical data as numeric values, which is necessary for many algorithms.

VectorAssembler: VectorAssembler is a feature transformation method that combines multiple features into a single vector. It takes a list of input columns and creates a new feature column containing all the specified features in a dense vector format. This step is often necessary before feeding the data into machine learning algorithms that require input features in a single vector form.

3.3. Big Data Platform

Apache Spark [7] initially started in 2009 and it became a part of the Apache Software Foundation in 2013. Apache Spark is an open-source cluster computing framework for real-time processing. Spark can be run in many clusters, which utilize the hardware of many devices for resolving a problem. Spark provides an interface for programming entire clusters with implicit data parallelism and fault-tolerant [8]. Thanks to saving data in multi clusters, when a cluster is broken, data can be retrieved from other clusters so it provides an efficient fault-tolerant. Spark has some important components that supply users with all the necessary tools from collecting data from streaming or other sources to processing data and modeling data. Spark Core is the base engine for large-scale parallel and distributed data processing. Further, additional libraries which are built atop the core allow diverse workloads for streaming, SQL, and machine learning. Spark Streaming helps to process real-time data from various sources such as Kafka, Flume, and Amazon Kinesis. After processing, this data can be pushed out to file systems or databases. Spark Streaming is used for real-time data which can be unstructured data such as images or text and Spark also provide a tool for dealing with structured data, which is Spark SQL[9]. In modeling, Spark has MLlib [10] is a machine learning framework for Spark. This framework provides a lot of choice for building predictive model from regression algorithms to classification algorithms.

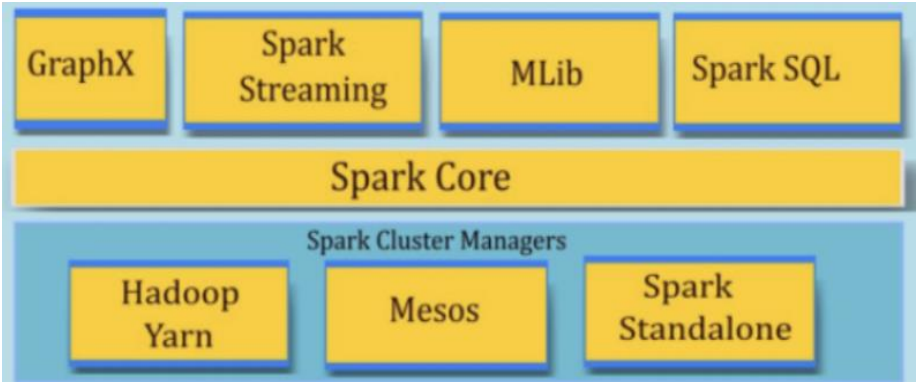

Figure 1. *Spark Components*

3.4. Bigdata Deep Learning

BigDL is a distributed deep learning library for Apache Spark; with BigDL, users can write their deep learning applications as standard Spark programs, which can directly run on top of existing Spark or Hadoop clusters.

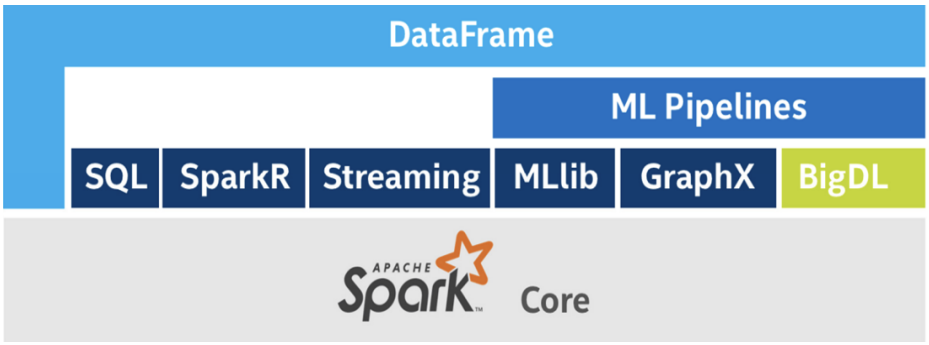

Figure 2. *BigDL in the Apache Spark*

3.5. Deep learning distributed

BigDL library, which integrates the powerful deep learning capabilities of Keras with the distributed computing environment of Apache Spark. Keras is a high-level neural network API that simplifies the process of building and training deep learning models. In this code, a neural network model is defined using the Sequential API, which allows for easy stacking of layers. [11]
The implementation of algorithms for distributed
deep learning to aggregate gradients from multiple nodes in a distributed system during the back-propagation step of deep learning training

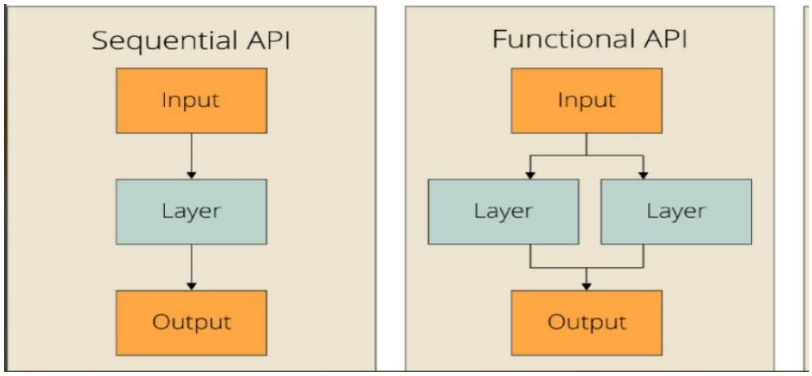

Figure3. *Deep learning distributed*

3.6. Model building

3.6.1. DNN models

A Deep Neural Network (DNN) is a machine learning technique that allows a computer, by training it, to do tasks that would be very difficult to do using conventional programming techniques. Neural network algorithms were inspired by the human brain and its functions: like our human mind, it is designed to work not only by following a preset list of rules, but by predicting solutions and drawing conclusions based on previous iterations and experiences. [12]

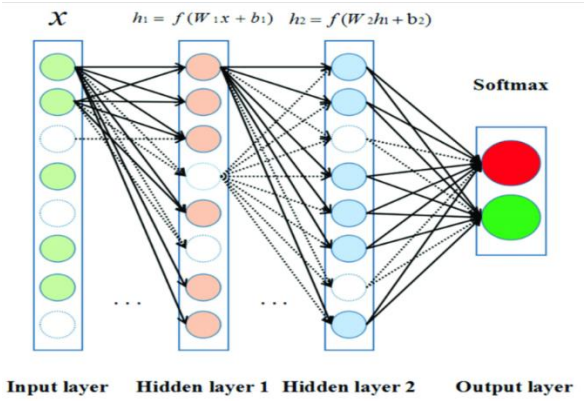

Figure4. *DNN models*

3.6.2. Random Forest

A Random Forest Algorithm is a supervised machine learning algorithm that is extremely popular and is used for Classification and Regression problems in Machine Learning. We know that a forest comprises numerous trees, and the more trees it will be robust. Similarly, the greater the number of trees in a Random Forest Algorithm, the higher its accuracy and problem-solving ability. Random Forest is a classifier that contains several decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset. It is based on the concept of ensemble learning which is a process of combining multiple classifiers to solve a complex problem and improve the performance of the model. [13]

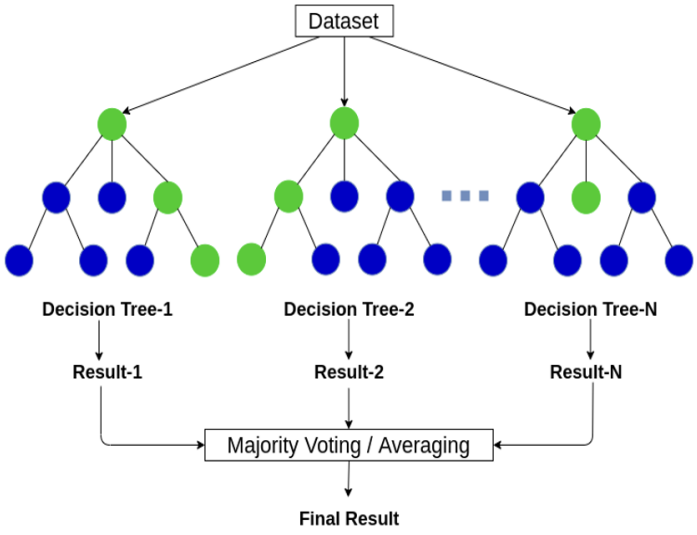

Figure 5. *Random Forest model*

3.6.3. Logistic Regression

Logistic Regression is one of the most important analytic tools in the social and natural sciences. Furthermore, logistic regression is a predictive analysis method to predict a binary outcome. The model delivers a binary or dichotomous outcome limited to two possible outcomes: yes/no, 0/1, or true/false based on prior observations of a data set. A logistic regression model predicts a dependent data variable by analyzing the relationship between one or more existing independent variables. [14]

The odds are simply the ratio of the proportions for the two possible outcomes.

$$ODDS = \frac{\hat{p}}{1 - \hat{p}}$$

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n$$

And instead of we are taking the probability . But there is an issue here, the value of will exceed 1 or go below 0 and we know that range of is (0 – 1).

$$y = \beta_0 + \beta_1 x => \frac{\hat{p}}{1 - \hat{p}} = \beta_0 + \beta_1 x$$

We know that odds can always be positive which means the range will always be. (0, +∞). The logistic regression solution to this difficulty is to transform the odds (using the natural logarithm). We use the term ***log odds*** or ***logit*** for this transformation.

$$\log\left(\frac{\hat{p}}{1 - \hat{p}}\right) = \beta_0 + \beta_1 x => \hat{p} = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

Sigmoid function is used to map predictions to probabilities. The sigmoid function has an S shaped curve. It is also called sigmoid curve.

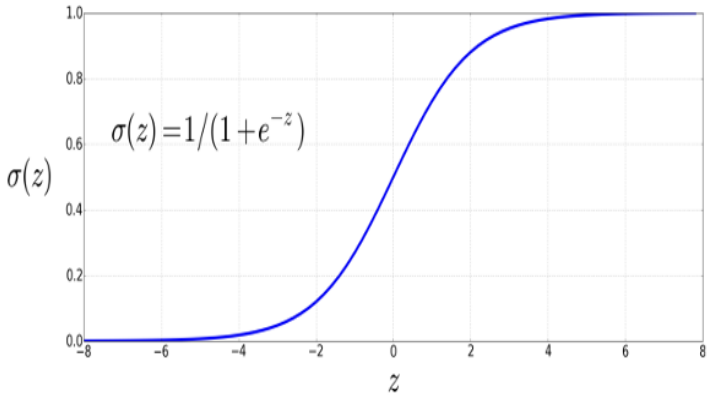

Figure 7. *Sigmoid model*

The sigmoid function from the prior section thus gives us a way to take an instance x and compute the probability P (y = 1|x). How do we make a decision about which class to apply to a test instance x? For a given x, we say yes if the probability P (y = 1|x) is more than .5, and no otherwise. We call .5 the decision boundary:

$$decision(x) = \begin{cases} 1 \; if \; P(y = 1|x) > 0.5 \\ 0, otherwise \end{cases}$$

4. Experiment

4.1. Experimental procedure

The experimental procedure involves several steps to develop a predictive model for hotel reservation booking status. Firstly, we load and explore the dataset to understand its structure and check for any missing data. Next, we preprocess the data by handling missing values, converting categorical variables into numerical representations, and scaling numerical features for fair modeling. Feature engineering is then performed to identify and transform important features into a suitable format. For parameter tuning, we select optimal hyperparameters for the model to enhance its performance. Techniques like Grid Search and Random Search are utilized to explore different hyperparameter combinations and find the best configuration. Lastly, we evaluate the model using deep learning based NNClassifier with Dense layers. The model's performance is assessed using various evaluation metrics such as precision, recall, and

F1-score. By comparing different hyperparameter settings, we aim to identify the best combination for building an accurate and effective predictive model for hotel reservation booking status.

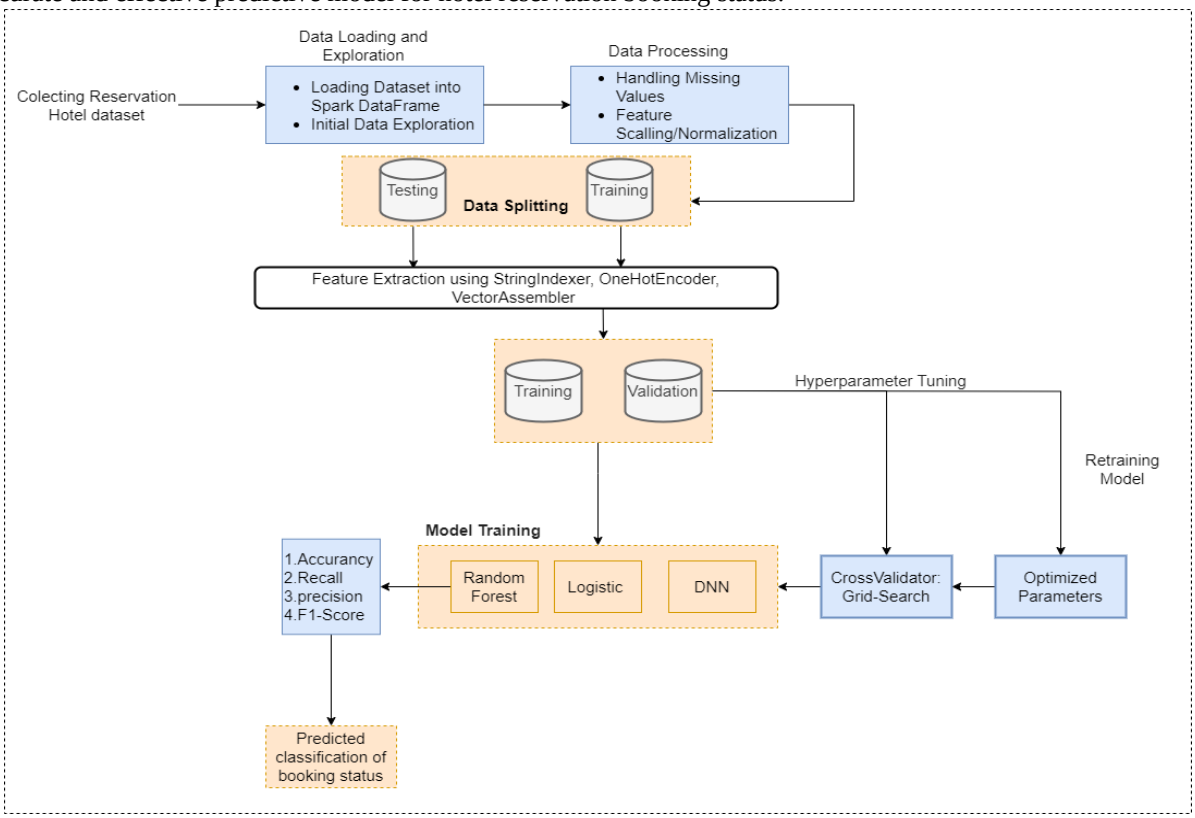

Figure 8. *System Architecture*

4.2. Data Collection

The Hotel Reservations Classification Dataset from Kaggle, Github, Real-World [15] provides data pertaining to hotel reservations, which includes a variety of factors such as booking changes, deposit type, days on the waiting list, and other customer details. With the advent of online hotel reservation channels, the booking behaviors of customers have significantly evolved. However, a substantial number of these reservations get canceled or result in no-shows due to various reasons like change of plans or scheduling conflicts. While the option to cancel free of charge or at a low cost is advantageous for hotel guests, it often results in potential revenue loss for hotels.

Booking_ID	no_of_adu	no_of_chi	no_of_we	no_of_we	type_of_meal_plan	required_c	room_type_reserved	lead_time	arrival_ye	arrival_mc	arrival_dat	market_se	repeated_	no_of_pre	no_of_pre	avg_price_	no_of_spe	booking_status
INN00001	2	0	1	2	Meal Plan 1	0	Room_Type 1	224	2017	10	2	Offline	0	0	0	65	0	Not_Canceled
INN00002	2	0	2	3	Not Selected	0	Room_Type 1	5	2018	11	6	Online	0	0	0	106.68	1	Not_Canceled
INN00003	1	0	2	1	Meal Plan 1	0	Room_Type 1	1	2018	2	28	Online	0	0	0	60	0	Canceled
INN00004	2	0	0	2	Meal Plan 1	0	Room_Type 1	211	2018	5	20	Online	0	0	0	100	0	Canceled
INN00005	2	0	1	1	Not Selected	0	Room_Type 1	48	2018	4	11	Online	0	0	0	94.5	0	Canceled
INN00006	2	0	0	2	Meal Plan 2	0	Room_Type 1	346	2018	9	13	Online	0	0	0	115	1	Canceled
INN00007	2	0	1	3	Meal Plan 1	0	Room_Type 1	34	2017	10	15	Online	0	0	0	107.55	1	Not_Canceled
INN00008	2	0	1	3	Meal Plan 1	0	Room_Type 4	83	2018	12	26	Online	0	0	0	105.61	1	Not_Canceled
INN00009	3	0	0	4	Meal Plan 1	0	Room_Type 1	121	2018	7	6	Offline	0	0	0	96.9	1	Not_Canceled
INN00010	2	0	0	5	Meal Plan 1	0	Room_Type 4	44	2018	10	18	Online	0	0	0	133.44	3	Not_Canceled
INN00011	1	0	1	0	Not Selected	0	Room_Type 1	0	2018	9	11	Online	0	0	0	85.03	0	Not_Canceled
INN00012	1	0	2	1	Meal Plan 1	0	Room_Type 4	35	2018	4	30	Online	0	0	0	140.4	1	Not_Canceled
INN00013	2	0	2	1	Not Selected	0	Room_Type 1	30	2018	11	26	Online	0	0	0	88	0	Canceled
INN00014	1	0	2	0	Meal Plan 1	0	Room_Type 1	95	2018	11	20	Online	0	0	0	90	2	Canceled
INN00015	2	0	0	2	Meal Plan 1	0	Room_Type 1	47	2017	10	20	Online	0	0	0	94.5	2	Not_Canceled
INN00016	2	0	0	2	Meal Plan 2	0	Room_Type 1	256	2018	6	15	Online	0	0	0	115	1	Canceled
INN00017	1	0	1	0	Meal Plan 1	0	Room_Type 1	0	2017	10	5	Offline	0	0	0	96	0	Not_Canceled
INN00018	2	0	1	3	Not Selected	0	Room_Type 1	1	2017	8	10	Online	0	0	0	96	1	Not_Canceled
INN00019	2	0	2	2	Meal Plan 1	0	Room_Type 1	99	2017	10	30	Online	0	0	0	65	0	Canceled
INN00020	2	0	1	0	Meal Plan 1	0	Room_Type 1	12	2017	10	4	Offline	0	0	0	72	0	Not_Canceled
INN00021	2	0	2	2	Meal Plan 1	0	Room_Type 1	99	2017	10	30	Online	0	0	0	65	0	Canceled
INN00022	1	0	0	1	Meal Plan 1	0	Room_Type 1	122	2018	11	25	Corporate	0	0	0	67	0	Not_Canceled
INN00023	2	0	2	4	Meal Plan 1	0	Room_Type 1	2	2018	3	20	Offline	0	0	0	85	0	Not_Canceled
INN00024	2	0	0	3	Meal Plan 1	0	Room_Type 1	37	2018	10	13	Offline	0	0	0	105	0	Not_Canceled
INN00025	2	0	2	1	Not Selected	0	Room_Type 1	130	2018	5	22	Online	0	0	0	94.5	1	Not_Canceled
INN00026	2	0	0	2	Meal Plan 1	0	Room_Type 1	99	2018	4	28	Online	0	0	0	114.3	1	Not_Canceled

Figur9. *Dataset from Kaggle*

The "Hotel Reservation" dataset contains information related to hotel bookings, comprising various features describing the characteristics of each reservation. The dataset includes details such as the booking ID, the number of adults and children included in the reservation, the number of weekend and week nights booked, the type of meal plan selected, the requirement for car parking space, the reserved room type, lead time before the reservation, arrival year, month, and date, the market segment type through which the booking was made (e.g., offline or online), whether the guest is a repeated visitor, the number of previous cancellations and bookings not canceled, the average price per room, the number of special requests made by the guest, and the current booking status (whether it's canceled or not). This dataset could be valuable for analyzing patterns and trends in hotel reservations and understanding factors that influence booking decisions. It may be used to gain insights into the preferences of guests, the effectiveness of different marketing strategies, the impact of lead time on booking rates, and how certain features or services offered by hotels affect booking outcomes. Additionally, this dataset could be used to build predictive models for hotel booking cancellations or to optimize pricing and service offerings to enhance guest satisfaction and maximize hotel revenue.

4.3. Data Processing

We use several techniques to process the data to improve performance. Our primary focus is on handling missing values and feature scaling to prepare the data for model training. To address missing values, we utilize Spark's DataFrame API's fillna function. For numerical columns *'no_of_adults', 'no_of_children', 'no_of_weekend_nights', 'no_of_week_nights','required_car_parking_space','lead_time','arrival_year','arrival_month','arrival_date','repeated_guest','no_of_previous_cancellations','no_of_previous_bookings_not_canceled','avg_price_per_room','no_of_special_requests'*, we replace missing values with the mean of their respective columns, and for categorical columns *'type_of_meal_plan', 'room_type_reserved', 'booking_status'* , we replace missing values with the mode of their respective columns. To feature scaling, By normalizing the features, we ensure they are on a consistent scale, leading to faster convergence during model training and reducing the risk of getting stuck in local minima. To achieve this, we employ the MinMaxScaler from the pyspark.ml. feature module, which scales the features to the range between 0 and 1. Next, We split dataset with 70% for Training and 30% for Testing.

4.4. Feature Engineering

In order to train the models, the input must be compatible to the algorithm to understand. Therefore, the values in features must be converted to a single vector so to implement the conversion, the MLlib [16] in Spark is used with estimators such as StringIndexer, OneHotEncoder, StandardScaler, VectorAssembler. There are different conversion methods for each data type. There are two types of data that need to be converted: category and numeric. The categorical data group includes data in the form of string, boolean and some other properties of ordinal For this group, StringIndexer is used to mark index each value in each attribute depending on the number of unique values, then we use OneHotEncoder to convert the index into a one hot vector(1). For the numerical data group, which includes all the remaining attributes. To transform this type of data, Skew and Kurtosis coefficients are used to filter out the attributes that need to be normalized according to the normal distribution. For attributes that do not need to be normalized, VectorAssembler aggregates them together with the vector of the categorical attribute group into a single vector (2). StandardScaler applies on the attributes to be normalized, then aggregates the conversion to vector(3) using VectorAssembler. Finally, executing the VectorAssembler again for (1),(2),(3) to aggregate into a large vector to feed into the model.

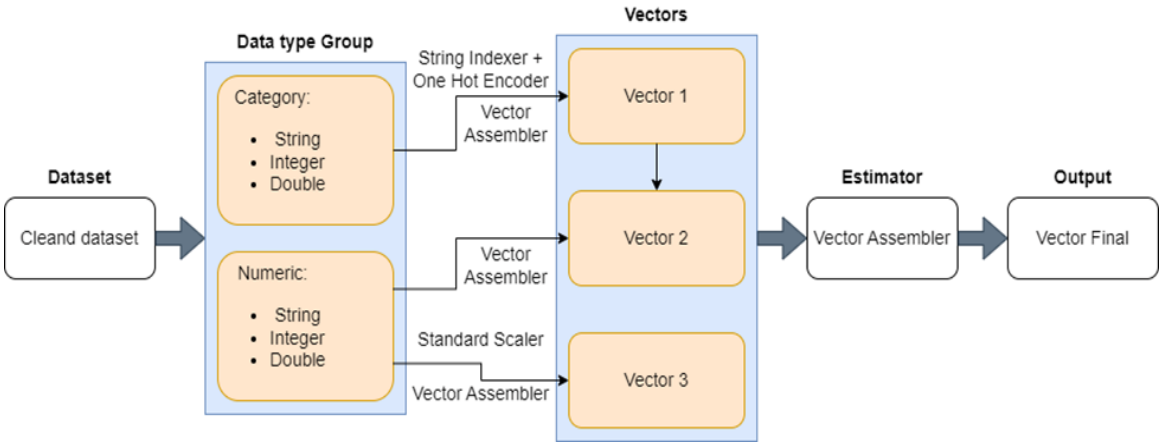

Figure 10. *Transformation pipeline*

4.5. Parameter Tuning

```
Model Summary:
_________________________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #       Connected to
=================================================================================================================
Inputc55c414a (Input)        (None, 20)                0
_________________________________________________________________________________________________________________
Densed978f592 (Dense)        (None, 64)                1344          Inputc55c414a
_________________________________________________________________________________________________________________
Dense18ebe94a (Dense)        (None, 32)                2080          Densed978f592
_________________________________________________________________________________________________________________
Dense80c97aad (Dense)        (None, 2)                 66            Dense18ebe94a
_________________________________________________________________________________________________________________
Total params: 3,490
Trainable params: 3,490
Non-trainable params: 0
```

Figure 11. *Model Summary.*

The process of optimization involves finding a tuple that provides an optimal model and minimizes loss function. There are several approaches to hyperparameter tuning. For our study, we have relied on the method of grid search, which consists of exhaustive searching through a subset of hyperparameter space of the algorithm, followed by a performance metric. In the grid searching process, data is scanned to configure optimal parameters of a specific model. Parameters are specific to the type of model considered. Grid searching is not confined to one type of model, but can be applied throughout machine learning, such that the best parameters can be identified for the model. The process of grid searching builds a model on each parameter combination possible, leading to multiple iterations. These model combinations for every parameter are stored, and hence, grid searching is computationally expensive. For a machine learner, parameter space may either include real valued or unbounded value spaces; hence, there may be a need to perform discretization [17] In this proposed work, The subsequent layer in the model is the max pooling layer, which performs a pooling operation to extract the maximum value within each patch of each feature map. Dropout is then applied to prevent overfitting during training. The fully connected layer follows, serving as the final layers in the network. The output from the previous pooling layer is “softmax” and passed into the fully connected layer as its input is “relu”. This arrangement eventually determines the sentiment of the texts.

```
|-- Booking_ID: string (nullable = true)
|-- no_of_adults: integer (nullable = true)
|-- no_of_children: integer (nullable = true)
|-- no_of_weekend_nights: integer (nullable = true)
|-- no_of_week_nights: integer (nullable = true)
|-- type_of_meal_plan: string (nullable = true)
|-- required_car_parking_space: integer (nullable = true)
|-- room_type_reserved: string (nullable = true)
|-- lead_time: integer (nullable = true)
|-- arrival_year: integer (nullable = true)
|-- arrival_month: integer (nullable = true)
|-- arrival_date: integer (nullable = true)
|-- market_segment_type: string (nullable = true)
|-- repeated_guest: integer (nullable = true)
|-- no_of_previous_cancellations: integer (nullable = true)
|-- no_of_previous_bookings_not_canceled: integer (nullable = true)
|-- avg_price_per_room: double (nullable = true)
|-- no_of_special_requests: integer (nullable = true)
|-- booking_status: integer (nullable = false)
|-- features: vector (nullable = true)
```

Figure 12*. Feature Selection*

To optimize the model's performance, hyperparameter tuning is employed. This process involves selecting the best combination of preset hyperparameters before training the model. The techniques for hyperparameter tuning are grid search. In grid search, the model is trained and evaluated for each combination of preset hyperparameter values, allowing us to choose the best combination based on the cross-validation score.Then, Choose these important features such as figure. 12. We performed hyperparameter tuning using cross-validation to find the optimal values for the

hyperparameters of our deep learning model. The hyperparameters we tuned are the batch size and the maximum number of epochs. The batch size determines how many samples are used in each training iteration, and the maximum number of epochs sets an upper limit on the number of training iterations. Grid search was used to try different combinations of batch sizes and maximum epochs, and the combination with the best performance based on cross-validation metrics was selected. In this specific model, a grid of hyperparameters is defined to search for the optimal settings. The hyperparameter grid includes options for batch size and the maximum number of training epochs. Cross-validation is then performed using this grid to evaluate the model's performance. The best model, determined through cross-validation, is obtained and the optimal hyperparameters are extracted. With these optimal hyperparameters, a new instance of the model is created and fitted to the data. In this case, a grid search to find the optimal hyperparameters for our model. We are interested in tuning the batch size [64, 128] and number of epochs [50, 100]. Training the Optimal Model: With the "optimal_classifier" model set up with the best hyperparameters, we proceeded to train it on the entire dataset. The training data, which includes both features and the corresponding target variable "booking_status," was used to fit the optimal model. During training, the model iteratively updated its weights and biases based on the optimization method (Adam in this case) and the defined loss function (CrossEntropyCriterion). The aim of training the model is to find the optimal set of weights and biases that minimize the loss and make accurate predictions.

4.6. Evaluation Metric

Precision, Recall and Accuracy are three metrics that are used to measure the performance of a machine learning algorithm[18].

4.6.1. Precision

The Precision is the ratio of true positives over the sum of false positives and true negatives. It is also known as a positive predictive value.

$$Precision = \frac{True\ Positive}{True\ Positive + \ False\ Positive}$$

Precision is a useful metric and shows that out of those predicted as positive, how accurate the prediction was.
If the cost of a false sensitivity error is high, precision may be a good measure of accuracy. For example, email spam detection. In email spam detection, if an email that turns out not to be spam gets flagged as spam, then it's as bad as it could get. Some email users might lose important emails if the precision for spam detection is not high enough.

4.6.2. Recall

Recall is the ratio of correctly predicted outcomes to all predictions. It is also known as sensitivity or specificity.

$$Recall\ = \frac{True\ Positive}{True\ Positive + \ False\ Negative}$$

Recall is just the proportion of positives our model are able to catch through labelling them as positives. When the cost of False Negative is greater than that of False Positive, we should select our best model using Recall.

4.6.3. Accuracy

Accuracy is the ratio of correct predictions out of all predictions made by an algorithm. It can be calculated by dividing precision by recall or as 1 minus false negative rate (FNR) divided by false positive rate (FPR).

$$Accuracy\ = \frac{(TP + \ TN)}{(TP + \ FP + \ TN + \ FN)}$$

4.6.4. F1-score

The F1-score combines these three metrics into one single metric that ranges from 0 to 1 and it takes into account both Precision and Recall.

$$F1\ = 2\ \text{x}\ \frac{Precision * Recall}{Precision + Recall}$$

The F1 score is needed when accuracy and how many of your ads are shown are important to you. We've established that Accuracy means the percentage of positives and negatives identified correctly. Now, we'll investigate "F1 Score,"

which is a way to measure how much accuracy is present in your dataset. we tend not to focus on many different things when making decisions, whereas false positives and false negatives usually have both tangible and intangible costs for the business. The F1 score may be a better measure to use in those cases, as it balances precision and recall.

4.7. Result and Discussion

Table 1. Experiment result

Values	DNN	Logistic Regression	Random Forest
Accuracy	0.32763611302549966	0.7716448726772196	0.7911906400550585
Precision	0.16381805651274983	0.7633439390203195	0.7989208739339401
Recall	0.5	0.7716448726772196	0.7911906400550586
F1-Score	0.24678156146179403	0.7614030648513426	0.7693315556813265

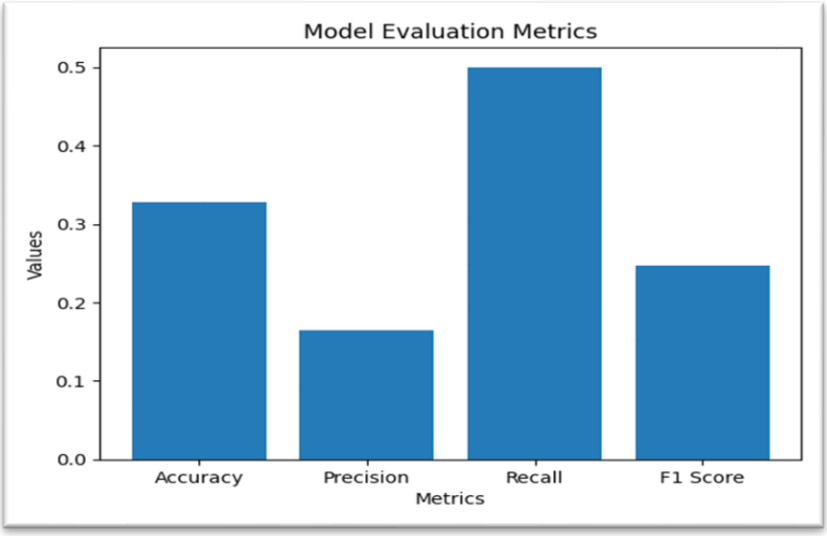

Figure 13. *DNN model performance of evaluation*

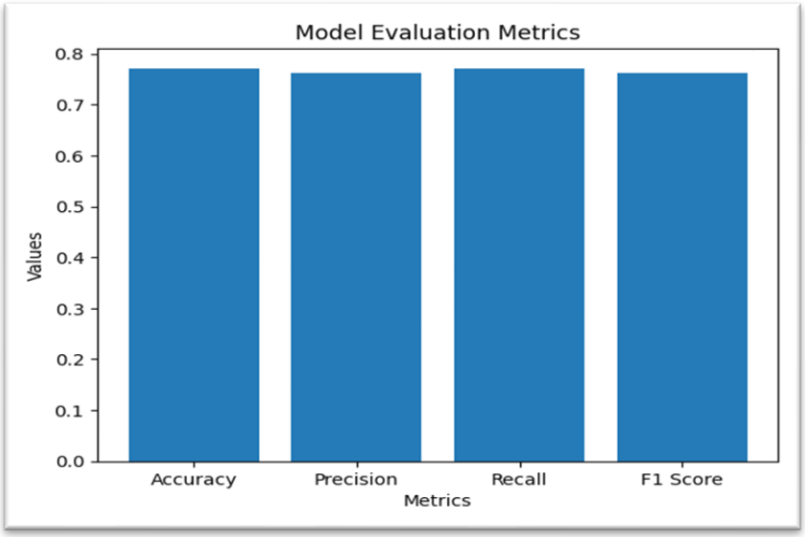

Figure 14. *Logistic Regression model performance of evaluation*

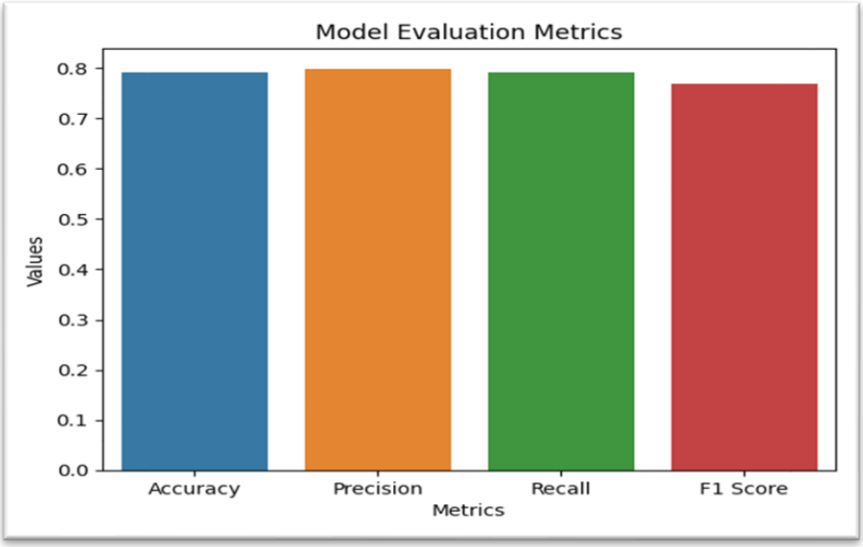

Figure 15. *Random Forest model performance of evaluation*

The overall performance of these models in classifying status of booking hotel is around 30% to 80% and accuracy from 20% to 80% f1-score. In general, Random Forest gives the best performance with 79.11% in accuracy in the dataset. This is easily understood because the algorithm in Random Forest is very effective in improving performance. By contrast, Deep Neural Network (DNN) shows the worst results in dataset.

5. Conclusion

In this study, we demonstrated the application of Deep Learning models using BigDL DLlib on Spark for a large-scale classification task. The chosen Deep Learning model demonstrated promising results on our data. The resrearch highlighted the powerful capabilities of Apache Spark in handling large datasets and performing distributed computing, making it an ideal choice for Big Data tasks. Furthermore, the integration of BigDL DLlib allowed us to leverage state-of-the-art Deep Learning techniques within the Spark ecosystem, presenting a potent combination of tools for advanced analytics and machine learning tasks. Although the metrics are relatively low, if developed correctly, Deep learning models prove their robustness in resolving many problems so apply deep learning models also should be considered to improving the performance in future.

REFERENCES

1. Junlong Xiang , Magnus Westerlund, Dušan Sovilj,Göran Pulkkis (2014) “Using Extreme Learning Machine for Intrusion Detection in a Big Data Environment”IEEE.
2. Samuel Marchal, Xiuyan Jiangz, Radu State, Thomas Engel (2014) “A Big Data Architecture for Large Scale Security Monitoring” , Springer.
3. Sung-Hwan Ahn, Nam-Uk Kim,Tai-Myoung Chung (2014) “Big Data Analysis System Concept for Detecting Unknown Attacks”, IEEE.
4. Kleber M.M. Vieira, Fernando Schubert, Guilherme, A. Geronimo, Rafael de Souza Mendes, Carlos B. Westphall(2014) “Autonomic Intrusion Detection System in Cloud Computing with Big Data”, Conference: The 2014 International Conference on Security and Management.
5. Mohiuddin Solaimani, Mohammed Iftekhar, Latifur Khan, Bhavani,Thuraisingham (2014) “Statistical Technique for Online Anomaly Detection Using Spark Over Heterogeneous Data from Multi-source VMware Performance Data,” IEEE.

6. Mittal, Shweta, and Om Prakash Sangwan. "Implementing machine learning algorithms on spark." *J. Math. Comput. Sci.* 11.5 (2021): 5267-5277.
7. Matei Zaharia et al. “Resilient Distributed Datasets: A Fault-Tolerant Abstraction for in-Memory Cluster Computing”.
8. Matei Zaharia et al. “Spark: Cluster Computing with Working Sets”. In: Pro-ceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing.
9. Michael Armbrust et al. “Spark SQL: Relational Data Processing in Spark”. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data.
10. Xiangrui Meng et al. MLlib: Machine Learning in Apache Spark. 2015. arXiv: 1505.06807 [cs.LG].
11. Elhmadany, Mohammed, Islam Elmadah, and Hossam E. Abdelmunim. "Instance Segmentation on Distributed Deep Learning Big Data Cluster." (2023)
12. Mittal, Sparsh. "A survey on modeling and improving reliability of DNN algorithms and accelerators." *Journal of Systems Architecture* 104 (2020): 101689.
13. Sarica, Alessia, Antonio Cerasa, and Aldo Quattrone. "Random forest algorithm for the classification of neuroimaging data in Alzheimer's disease: a systematic review." *Frontiers in aging neuroscience* 9 (2017): 329.
14. Minka, Tom. "Algorithms for maximum-likelihood logistic regression." *Statistics Tech Report* 758 (2001)
15. https://www.kaggle.com/datasets/ahsan81/hotel-reservations-classification-dataset
16. Xiangrui Meng et al. MLlib: Machine Learning in Apache Spark. 2015. arXiv: 1505.06807 [cs.LG].
17. Feurer M, Hutter F (2019) Hyperparameter optimization. In: Automated machine learning. Springer, Cham, pp 3–33
18. Hossin, Mohammad, and Md Nasir Sulaiman. "A review on evaluation metrics for data classification evaluations." *International journal of data mining & knowledge management process* 5.2 (2015): 1.