SSRG International Journal of Computer Science and Engineering
ISSN: 2348–8387 / https://doi.org/10.14445/23488387/IJCSE-V10I5P103

Volume 10 Issue 5, 17-25, May 2023
© 2023 Seventh Sense Research Group®

Original Article

Machine Learning in Google Cloud Big Query using SQL

Ravi Kashyap

SAP Data Management & Analytics Lead, IBM Corporation, 425 Market St, San Francisco, CA, USA

Received: 01 April 2022 Revised: 06 May 2022 Accepted: 17 May 2023 Published: 31 May 2023

Abstract - *In today's world, data has become a valuable resource for businesses, governments, researchers, and individuals alike. However, to truly extract value from data, it is essential to provide the proper context. Simply collecting and analyzing data without understanding its context can lead to inaccurate conclusions and misguided decision-making. An important factor that drives a successful organization is gathering data that can be analyzed to gain greater insights into the business and enable new opportunities, allowing the business to innovate products/services based on consumer preference. Data is the lifeblood of all businesses, and data-driven decisions can make a significant difference in staying ahead of the competition. Machine learning can be the key to unlocking the value of corporate and customer data, enabling businesses to leverage their data to make more accurate predictions and decisions. It can help businesses to identify patterns and trends in their data that might not be apparent to humans, leading to more accurate predictions and better decisions.*

Challenge: Machine learning (ML) requires trained professionals and data scientists having good knowledge of programming languages like Python or R. One of the big challenges for data analytics or business people with an SQL background is that it does not matter how good their domain knowledge is, they still cannot contribute much technically when it comes to ML initiatives in their organization. On the other hand, data scientists have good knowledge of statistics, algorithms, Python or R, but they may not have specific data or domain knowledge which can lead to inaccurate predictions and flawed decisions.

This paper aims to enable data analysts and data scientists to build and deploy machine learning models on massive datasets using SQL queries without the need for extensive knowledge in programming or data preprocessing.

Keywords - *Data warehousing, BigQuery, Machine learning, Data analysis, Predictive modeling, SQL, Data preprocessing, Model selection, Model evaluation, Cloud computing, Artificial intelligence.*

1. Introduction

Google BigQuery, Google Cloud's enterprise data warehouse designed for business agility, was released to general availability in 2011. The serverless architecture enables scalable and fast operations, enabling fast SQL analytics on large datasets. Since its inception, numerous features and improvements have been made to enhance its performance, security, reliability, and user experience, making it easier for users to discover insights.

BigQuery machine learning is a cloud-based machine learning service offered by Google Cloud Platform. It enables data analysts and data scientists to build and deploy machine learning models on massive datasets using SQL queries without the need for extensive knowledge in statistics, algorithms, Python or R programming, or data preprocessing. BigQuery ML democratizes machine learning by letting data analysts create, train, evaluate and predict with ML models using existing SQL tools and skills. It is a blessing to data analysts with good knowledge of domain data and SQL to build operational production-grade ML models.

Consider BigQuery as the combination of data warehouse and machine learning. It provides the capabilities to create robust and scalable ML models while staying in a single Google Cloud service. The service provides various pre-built machine learning models such as linear regression, logistic regression, k-means clustering, and time-series forecasting, which can be applied to the data with simple SQL statements. BigQuery ML also allows users to create custom models using TensorFlow or Keras and integrate them into their SQL queries.

BigQuery ML provides advanced features like automated data preprocessing, automatic model selection, and hyperparameter tuning. The service also allows users to monitor the performance of their models and get insights into the accuracy of the predictions through the integrated TensorFlow Data Validation. It provides a powerful and user-friendly solution for building and deploying machine learning models on large datasets, making it an excellent choice for businesses looking to make data-driven decisions.

2. Literature Review

BigQuery ML is a machine learning tool that enables users to build and execute machine learning models directly in Google BigQuery, a cloud-based data warehouse. It combines the power of BigQuery's scalable, high-performance SQL engine with Google's machine learning expertise, providing a powerful tool for data analysts and data scientists to analyze and model large datasets.

The advantage of BigQuery ML is its ease of use, as it requires minimal coding and no expertise in machine learning algorithms. Instead, users can use SQL queries to create and train models and evaluate and predict outcomes.

BigQuery ML has been used in various applications, including predictive modeling, classification, and feature engineering. For instance, one study used BigQuery ML to develop a model for predicting the risk of hospital readmissions, achieving high accuracy, and identifying key factors associated with readmissions. Another study used BigQuery ML to classify customer reviews of a product, achieving high accuracy and identifying key features that influenced customer satisfaction.

However, BigQuery ML has some limitations. For example, the platform does not support unsupervised learning algorithms, which limits its applicability for certain use cases. Additionally, the lack of flexibility in model selection and evaluation can make fine-tuning models for specific tasks challenging.

BigQuery ML is a powerful tool for machine learning in the cloud, providing users with a scalable and easy-to-use data analysis and modeling platform. BigQuery data warehouse makes it a valuable tool for data scientists and analysts looking to leverage machine learning for their business applications.

3. An Overview of BigQuery ML (BQ ML)

3.1. What is BigQuery ML

BigQuery ML is a machine learning tool provided by Google Cloud's BigQuery platform. It allows users to build and deploy machine learning models directly within BigQuery using SQL queries without the need for additional programming languages or tools. With BigQuery ML, users can eliminate the need for complex data preprocessing and feature engineering that is typically required in traditional machine learning workflows. The platform supports several machine learning algorithms, including linear regression, logistic regression, k-means clustering, and decision trees. It also supports TensorFlow, a popular machine learning framework, allowing users to build custom models using Python.

BigQuery ML is designed to work seamlessly with the rest of the Google Cloud Platform, enabling easy integration with other cloud services such as Cloud Storage, Cloud Dataproc, and Cloud Dataflow. This makes it a powerful tool for data scientists and analysts looking to leverage machine learning for their business applications. Machine learning is used in a wide range of applications, including natural language processing, image and speech recognition, recommendation systems, and predictive analytics. It is also used in industries such as healthcare, finance, and transportation to improve efficiency, reduce costs, and make more accurate predictions.

3.2. What is Machine Learning

Machine learning is a subfield of artificial intelligence (AI) that focuses on developing algorithms and models that enable computers to learn from data and make predictions and decisions without being explicitly programmed. Instead of following static instructions, machine learning algorithms learn patterns and relationships from input data and use that knowledge to make predictions or take actions on new, unseen data. Machine learning has three main categories: supervised learning, unsupervised learning, and reinforcement learning. Supervised learning involves training a model using labeled data to make predictions on new data. Unsupervised learning involves training a model on unlabeled data to identify patterns and structures in the data. In reinforcement learning, a model is trained to make decisions by receiving rewards for correct actions and punishments for incorrect actions.

Machine learning algorithms can be divided into two categories: traditional machine learning algorithms and deep learning algorithms. Traditional machine learning algorithms include decision trees, support vector machines, and logistic regression, while deep learning algorithms include artificial neural networks and convolutional neural networks.

3.3. BigQuery ML Features

1. Train the models in SQL (R or Python knowledge is not required).
2. In BQ ML, most of the hyperparameter tuning activities that are usually highly time-consuming are automated and give the flexibility of self-customization. It also reduced the training time of the models as it works directly where the data has been stored.
3. BQ ML integrates with Business intelligence (BI) Tools like Data Studio, Looker, and Tableau. Predictions are immediately reflected in the BI tools in the form of reports or charts.

4. Using Looker, developers can create bigquery tables for data exploration, and with LookML (markup language), we can create a model while staying in Looker software and even operationalize ML workflow.
5. BQ ML Supports a wide range of functions like standard SQL, UDFs within ML Queries, Evaluation functions, feature preprocessing functions, model weight inspection functions, model explainability functions, and many more.
6. AutoML regressor or AutoML classifier model can be easily created using BigQuery ML.
7. Provide flexibility to export the BQ ML models for online prediction in the Google Cloud AI platform.
8. BigQuery ML functionality is available by using:
   - Google Cloud console
   - bq command-line tool
   - BigQuery REST API
   - Jupyter Notebook or Business Intelligence Platform.

3.4. Workflow of a BigQuery ML(BQ ML)

1. ETL or ingestion data into BigQuery.
2. Feature Preprocessing: Clean the data and convert the raw data into a usable format that can be fed to train the model. Here SQL skills are required to perform table joins, apply string functions, and SQL transformations to prepare a good dataset for training the model. It provides the flexibility of manual preprocessing with certain ML Analytics Functions.
3. Create Data Model: Creating model queries in BigQuery ML is very easy and self-understandable. Here is an example of Creating a Linear Regression Model.
   Syntax:
   "Create or Replace Model '<Model_Name>'
   TRANSFORM (<select_list>) -- Optional
   Options
   (Model_type = '<Model_Option_List>',
   Input_label_cols = [<Column_name>]) AS
   *Select * from <Source_table_name> "*

   Note: It also gives the flexibility to fine-tune the model using optional parameters specific to algorithms. *TRANSFORM* is optional; any data preprocessing before the model is trained can be done here. When the *TRANSFORM* clause is present, only those columns of the TRANSFORM clause are used in training. Functions that cannot be used inside the TRANSFORM clause are aggregation functions, Non-ML Analytic Functions, UDFs, and subqueries.

4. Model Evaluation: To evaluate the model's performance or losses of the trained model. Here is another example of creating tiny queries to evaluate the model. Like the evaluation function, there are more functions like the ROC curve and confusion matrix.
   Syntax:
   *Select * from ML.EVALUATE (MODEL*
   *'<Model_Name>',(select*from Source_table_name>))*

5. Model Prediction: Here is an example to create tiny queries to predict the model.
   Syntax:
   *"Select * from ML.PREDICT(MODEL*
   *'<Model_Name>', (select * from*
   <Source_table_name>))"

BigQuery can be divided into below parts: Ingestion, Storage, and Preparation.

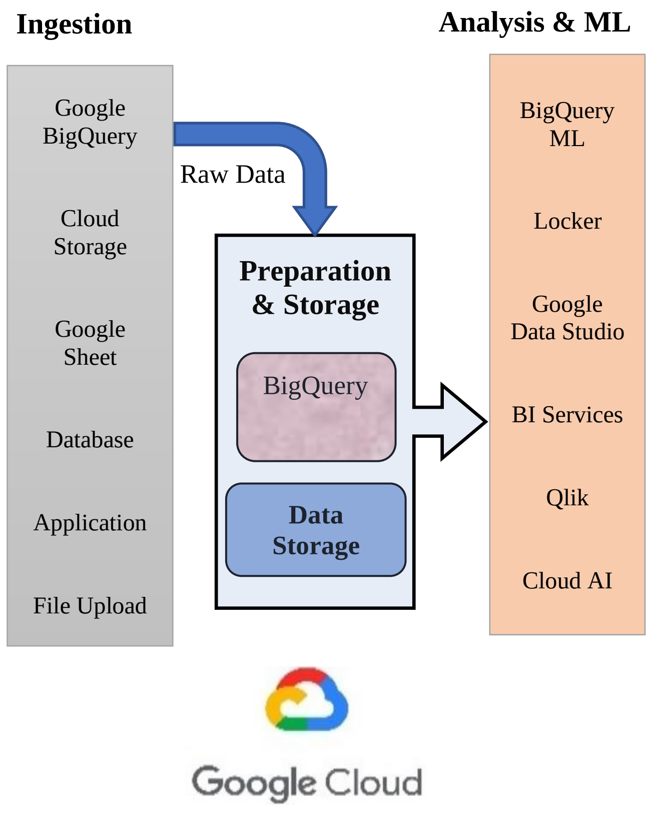

Fig. 1 BigQuery working flow

3.5. BQ ML Supported Model

The following models are built into BigQuery ML:

1. Linear regression: It is the easiest and most popular supervised learning ML Algorithm to forecast numerical values—for example – Forecasting sales of a store. To create a Linear regression model in BigQuery ML, *set MODEL_TYPE = 'LINEAR_REG'.*
2. Logistic regression: It is a supervised machine learning algorithm used for classification problems which means it classifies the incoming data based on historical data. Choosing between multiple options. For example –the probability of a customer purchasing some item can be classified as very high, high, medium or low. To create a Logistic regression model in BigQuery ML, *set MODEL_TYPE = 'LOGISTIC_REG'.*

3. K-means clustering: It comes under the unsupervised category of machine learning algorithms. It groups unlabeled datasets into different clusters. K means to perform the division of objects into clusters that share similarities within the cluster. It is used for data segmentation for similar objects. To create a K-means model in BQ ML, *set MODEL_TYPE = 'KMEANS'.*
4. Matrix factorization: This algorithm is mostly used for building product recommendations based on historical information. Best fit in the media industry for content recommendations. To create a matrix model, *set MODEL_TYPE = 'MATRIX_FACTORIZATION'.*
5. Principal component analysis (PCA): It is an unsupervised dimensionality reduction algorithm that is used to reduce dimension in large datasets by transforming a large set of variables into a smaller one that still contains most of the information of a large set. To create this, *set MODEL_TYPE = 'PCA'.*
6. Time series: Forecast business KPI's by using historical time series data. Time series forecasting involves examining time series data through statistical analysis and modeling to make predictions. It is a technique for predicting events through a sequence of time. These algorithms are mostly used in forecasting the stock market price and website traffic. To create a time series model in BigQuery ML, *set MODEL_TYPE = 'ARIMA_PLUS'.*

The subsequent models are trained in Vertex AI and external to BigQuery ML:

7. Deep neural network (DNN): Creating TensorFlow-based deep neural network for classification and regression model. To create this, *set MODEL_TYPE = {'DNN_CLASSIFIER' | 'DNN_REGRESSOR' }*
8. Wide & Deep: It is a technique that combines the strengths of both linear models and deep neural networks. To create this model in BQML, *set MODEL_TYPE={'DNN_LINEAR_COMBINED_CLASSIFIER'|'DNN_LINEAR_COMBINED_REGRESSOR' }*
9. Autoencoder: It is an unsupervised neural network used for anomaly detection, non-linear dimensionality reduction, feature extraction, and classification. To create this *set MODEL_TYPE = 'AUTOENCODER'*
10. Boosted Tree: This algorithm is used for both classification and regression use cases with the XG Boost variant. To create this model in BQML, *set MODEL_TYPE={ 'BOOSTED_TREE_CLASSIFIER' | 'BOOSTED_TREE_REGRESSOR' }*
11. Random forest: It is a supervised learning algorithm that can be used to build models for predicting the values of a target variable based on one or more input variables. To create this model in BQML, *set MODEL_TYPE={'RANDOM_FOREST_CLASSIFIER' | 'RANDOM_FOREST_REGRESSOR' }*
12. Vertex AI AutoML Tables: It is a supervised ML service that uses tabular data to create ML models on structured data at high speed. To create this model, *set MODEL_TYPE={'AUTOML_REGRESSOR'| 'AUTOML_CLASSIFIER' }*

4. Harnessing the Power of Machine Learning for Predictive Analytics in BigQuery

Here we are going to explain the implementation of the most frequently used ML algorithms in data analytics.

4.1. Implementation of the Linear Regression Algorithm

Linear regression is a type of supervised machine learning algorithm used for regression analysis, where the goal is to predict a continuous output variable based on one or more input variables. In linear regression, a linear relationship is assumed between the input variables and the output variable.

The goal of linear regression is to fit a straight line through the data points that best represents the relationship between the dependent variable and the independent variable(s). This line is called the regression line and is defined by the equation **$y = mx + c + \varepsilon$**
y - Dependent Variable
x - Independent variable
m - Slope of a line
c – Coefficient of the line
ε – Error

This best-fit line should have the least sum of squares of the errors. This is done using a technique called least squares regression.

Note: when a model is trained too well on the training data, to the extent that it starts to fit to noise in the data rather than the underlying patterns. This leads to a model that performs well on the training data but poorly on new data, which means the model is not a generalized model and not good for real data. This problem is called overfitting. To prevent this, various regularization techniques can be applied, such as L1 and L2. These techniques add constraints to the model to prevent it from fitting too closely to the training data.

Here are the high-level steps to implement a linear regression algorithm in BigQuery:

1. Create a dataset and table in BigQuery that contains the data you want to use for the linear regression analysis.
2. Write a SQL query that specifies the linear regression model you want to build. The query should include the following:
   - ➢ The input variables you want to use in the model.
   - ➢ The output variable you want to predict.
   - ➢ The name of the BigQuery ML model.
   - ➢ The type of algorithm you want to use.

Syntax:
"Create or Replace Model '<Model_Name>'
TRANSFORM (<select_list>) -- Optional
Options
(Model_type = 'linear_reg',
Input_label_cols = [<label_column>]) as
Select input_feature1, input_feature2, label_column from <Source_table_name> "

3. Run the query to train the model. BigQuery ML will automatically split the data into training and evaluation sets and train the model using the specified algorithm.
4. Once the model is trained, you can use another SQL query to evaluate the model's performance using a test dataset.
   Syntax:
   *SELECT * FROM ML.EVALUATE(MODEL '< Model_Name >', (Select input_feature1, input_feature2, label_column from <Source_table_name>))*
   Or
   *SELECT * FROM ML.EVALUATE(MODEL '< Model_Name >', table '<Source_table_name>')*

   Note: ML.EVALUATE function does not support the imported TensorFlow model.

   There is another ML.TRAINING_INFO function that allows seeing information about the training iterations of a model. It does not work with Imported TensorFlow models and for Time series models.
5. Optimization technique to tune and resolve the overfitting problem in Create model query by adding the following new feature options.
   - L2_REG, also known as Ridge regression, is a type of linear regression that adds a regularization term to the standard least squares objective function. The regularization term is the L2 norm of the model coefficients, multiplied by a regularization parameter alpha.
   - Early stopping is a regularization technique used to prevent overfitting and improve the generalization performance of a model. If set to true, it stops training after the first iteration in which the relative loss improvement is less than the threshold values.
   - Optimize strategy specifies the strategy to minimize the cost function. It can have the following values.
     AUTO_STRATEGY – This chooses the next 2 options based on the total cardinalities of training.
     BATCH_GRADIENT_DESCENT – Use all the observations of the dataset, calculate the cost of the function, and update the parameters.
     NORMAL_EQUATION – It computes the least square solution of the linear regression problem.

   Note: Avoid selecting the columns from the source table which are not useful for the model will optimize the performance further.

   Syntax:
   "Create or Replace Model '<Model_Name>'
   TRANSFORM (<select_list>) -- Optional
   Options
   (Model_type = 'linear_reg',
   Input_label_cols = [<label_column>],l2_reg = 0.1,early_stop = false, max_iteration = 12,optimize_strategy = 'batch_gradient_descent') AS
   Select input_feature1, input_feature2, label_column from <Source_table_name> "

6. Final step, use the trained model to make predictions on new data by executing the below query.
   Syntax:
   *Select * from ML.PREDICT(MODEL '<Model_Name>' , (Select input_feature1, input_feature2, label_column from <Source_table_name>))*
   Or
   *SELECT * FROM ML.PREDICT (MODEL '< Model_Name >', table '<Source_table_name>')*

In step-5, you have observed that L2_REG values have been passed manually. This can be eliminated by using the Hyperparameter Tuning option. The process of trying multiple values for a hyperparameter in model training and finding the best value for a model is called hyperparameter tuning.

The hyperparameters that can be tuned in BigQuery ML depend on the type of model being trained. For example, in linear regression, the hyperparameters that can be tuned include the learning rate, the regularization parameter, and the early stopping criteria. In logistic regression, the hyperparameters that can be tuned include the learning rate, the regularization parameter, and the class weight. Please refer to the link in the reference section to review the complete list of hyperparameters options.

To let the model train using hyperparameter tuning, the minimum option is to add Num_trails to create the model query.

Syntax:
"Create or Replace Model '<Model_Name>'
TRANSFORM (<select_list>) -- Optional
Options
(Model_type = 'linear_reg',
Input_label_cols = [<label_column>],early_stop = false, max_iteration = 12,optimize_strategy = 'batch_gradient_descent',NUM_TRAILS = 10,l2_reg = HPARAM_RANGE(0.9,2)) AS
Select input_feature1, input_feature2, label_column from <Source_table_name> "

Num_trails specifies the number of times to train the model with different hyperparameter settings. Usually, the thumb rule is to use at least a number of hyperparameters *10 trials to tune a model. For example, we have to tune only 1 (l2_reg hyperparameter), then 1*10. The total is 10 values for NUM_TRAILS, as mentioned in the above syntax. Setting a higher value for NUM_TRAILS increases the chances of finding the best set of hyperparameters for the model, but it also increases the computational cost and training time.

Note: The ML.TRIAL_INFO function is used to display information regarding trials from a hyperparameter tuning model instead of ML.TRAINING_INFO.

Another function is model explainability which is optional and helps users to understand and interpret the behavior and predictions of the machine learning models built in BigQuery ML. For the Linear regression model, we have 3 explainability functions.

- ML.EXPLAIN_PREDICT - This function generate a predicted value and a set of feature attributes per instance of the data. Feature attributes indicate how much each feature contributed to the final prediction in the model. Consider this as an extended version of ML.PREDICT.

Syntax:
*SELECT * FROM ML.EXPLAIN_PREDICT (MODEL '< Model_Name >', table '<Source_table_name>')*

- ML.GLOBAL_EXPLAIN - This function explains the entire model by aggregating the local explanation of the evaluation data. It shows how much each feature contributed to the whole model. To use this option, it is mandatory to set "enable_global_explain = true" in create the model query.

Syntax:
*SELECT * FROM ML.GLOBAL_EXPLAIN (MODEL '< Model_Name >', table '<Source_table_name>')*

- ML.ADVANCE_WEIGHTS – This function is used to see underlying weights used by linear regression during prediction with associated p-values and standard errors for the weight.

Please refer to the link in the reference section to review the list of functions supported by all models.

4.2. Implementation of Logistic Regression Algorithm

Logistic regression is a supervised machine learning algorithm used for classification problems which means it classifies the incoming data based on historical data. It is popular for binary classification, where predictions come in the form of yes or no, 0 or 1.

In logistic regression, the logistic function (also known as the sigmoid function) is used to model the relationship between the input features and the probability of the positive class. The output of the logistic function is a value between 0 and 1, which can be interpreted as the probability of the input belonging to the positive class. The logistic function is defined as:

$$f(x) = 1 / (1 + e^{(-z)})$$

Where z is the linear combination of the input features and weights:

$$z = w0 + w1x1 + w2x2 + ... + wnxn$$

The logistic regression model is trained using maximum likelihood estimation, which finds the set of weights that maximize the likelihood of the observed data given the model. This involves minimizing a loss function, typically the cross-entropy loss, which measures the difference between the predicted probabilities and the true labels.

Type of logistic regression.

1. Binomial: It has 2 possible types of dependent variables such as 0 or 1, Pass or fail, etc.
2. Multinomial: It is used for multi-class classification problems, where the goal is to predict the probability of an input belonging to each of the multiple classes.
3. Ordinal: It is used to model relationships between input features and an ordered categorical response variable.

Now, another important point here is logistic regression evaluation metrics which are based on the combination of True positive, True negative, False positive, and False negative values.

A True positive (TP) is an outcome where the model correctly predicts the positive class.
A True negative (TN) is an outcome where the model correctly predicts the negative class.
A False positive (FP) is an outcome where the model correctly predicts the positive class.
A False negative (FN)is an outcome where the model correctly predicts the negative class.

Based on the above values, we can build a confusion matrix for summarizing the performance of the classification algorithm. Using this confusion matrix, we can create a ROC curve and visualize important predictive analytics like recall, accuracy, and precision.

ROC Curve is based on the TP rate and NP rate.
True positive Rate = TP / (TP+FN)
False positive Rate = FP / (FP+TN)

Precision provides a measure of how precise the model's positive predictions are.
TP / (TP+FP)

Recall provides a measure of how well the model can identify positive instances.
TP / (TP+FN)
Accuracy measures the model's ability to classify instances into their true classes correctly.
TP+TN / (TP+FP+TN+FN)

Here are the high-level steps to implement a logistic regression algorithm in BigQuery:

1. Data needs to be preprocessed with one row per observation and one column per feature. We should also have a label column indicating the binary outcome for each observation stored in the BigQuery table before applying logistic regression.
   BigQuery supports two types of feature preprocessing to prepare data for analysis using machine learning models.

   Automatic Preprocessing – BigQuery ML performs certain types of preprocessing during the training process when using the CREATE MODEL statement. This consists of missing value imputation and feature transformations. It includes handling missing values and performing feature transformations, such as one-hot encoding and normalization.

   Manual Preprocessing – Following are manual preprocessing functions like ML.BUCKETIZE to convert a numerical column to a categorical one. Other functions are ML.QUANTILE_BUCKETIZE
   - ML_FEATURE_CROSS
   - ML.NGRAMS
   - ML.MIN_MAX_SCALER
   - ML.STANDARD_SCALER
   - ML.FEATURE_INFO allows you to see information about the input features used to train a model.
2. Write a SQL query that specifies the logistic regression model you want to build.

Syntax:
"Create or Replace Model '<Model_Name>'
TRANSFORM (<select_list>) -- Optional
Options
(Model_type = 'logistic_reg',
Input_label_cols = [<label_column>],
AUTO_CLASS_WEIGHT = True) as
Select input_feature1, input_feature2, label_column from <Source_table_name> "

3. Other steps are similar, which have been explained in *4.1. (Linear regression implementation).*

4.3. Implementation of K-means Algorithm

Let's understand clustering first before drilling down to the K-means clustering algorithm. Clustering in ML is the process of dividing the datasets into groups based on similar data points. While creating the group, it is important to make sure that the points or entities in the same groups have as similar features as much as possible and that the points in different groups are as much as dissimilar. It is an unsupervised learning technique that does not rely on labeled data to learn patterns. Instead, it uses an algorithm to analyze the data, identify similarities or differences between data points, and then group them based on those similarities or differences. It has many use cases, such as customer segmentation, market segmentation, image segmentation, anomaly detection, social networking analysis, search result grouping, etc.

Types of clustering algorithms are Centroid-based clustering (K-means), hierarchical clustering, density-based clustering, and distribution-based clustering. BigQuery only supports the K-Means algorithm.

K-Means is an unsupervised learning algorithm that attempts to group data points into a predefined number of clusters, with each data point belonging to the cluster with the closest mean. It is an iterative algorithm that divides the unlabeled dataset into 'K' different clusters in such a way that each data point belongs to only one group that has similar properties.

The k-means algorithm works as follows:

1. Choose the number of clusters (k) that you want to create. The elbow method and The Silhouette method can be used to find the K values.
2. Randomly assign each data point to a cluster.

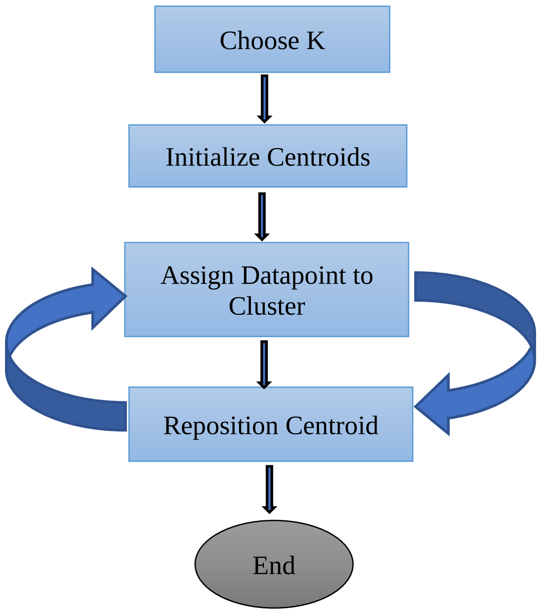

Fig. 2 Flowchart of K means

3. Compute the centroid of each cluster (the mean of all data points in the cluster).
4. Assign each data point to the closest centroid.
5. Compute the new centroid of each cluster based on the updated assignments.
6. Repeat steps 4 and 5 until the assignments no longer change or a maximum number of iterations is reached.

The K-means algorithm does have some limitations as well. For example, it assumes that the clusters are spherical and equally sized, which may not be the case in some datasets. Additionally, the choice of the number of clusters (k) can be challenging and affect the resulting clusters' quality. It does not work very well when there are two highly overlapping data.

The SQL query specifies the K Means model using the hyperparameter parameter tuning option to auto-pick the best value for K means. Keep relevant columns selected from the source table.

Syntax:
"Create or Replace Model '<Model_Name>'
TRANSFORM (<select_list>) -- Optional
Options
(Model_type = 'KMEANS',
NUM_TRAILS = 6,NUM_CLUSTERS = HPARAM_RANGE(2,7), KMEANS_INIT_METHOD = 'KMEANS++') AS
Select input_feature1, input_feature2,……n column from <Source_table_name> "

Note:
1. NUM_CLUSTER<int64>
   - Set the number of clusters to identify in the input data.
   - Allowed range is an integer value ranging from 2 to 100.
   - Default value is log10(n), where n is the number of training examples.
2. KMEANS_INIT_METHOD – It can have the below values.
   - RANDOM- It initializes the centroids by randomly selecting data points from the input data. Default strategy.
   - KMEANS++ - 1st cluster centre is chosen at random; afterwards, each subsequent centroid is chosen with probability proportional to its squared distance from the points closest to the existing cluster.
   - CUSTOM – It uses a customer-provided Boolean column.

4.4. Implementation of Principal Component Analysis (PCA) Algorithm

Principal Component Analysis (PCA) is a well-known unsupervised machine learning technique used for reducing data dimensionality and feature extraction. It works by transforming the original high-dimensional data into a lower-dimensional space while preserving the most important information.

The steps involved in PCA are:
1. Standardize the range of the continuous initial variable so that each one of them contributes equally to the analysis.
2. Calculate the covariance matrix. Covariance is a measure of the relationship between two random variables, and a covariance matrix is a table that summarizes the correlation between all the possible pairs of variables.
3. Calculate Eigenvectors and Eigenvalues. Eigenvectors determine the axis direction with maximum variance, and Eigenvalues determine the amount of variance along the Eigenvector.

PCA is widely used in various applications such as image recognition, signal processing, and finance. It can help to reduce the computational cost of machine learning algorithms and improve their performance by removing irrelevant or redundant features.

Here is the SQL query that specifies the PCA model. Keep relevant columns selected from the source table.

Syntax:
"Create or Replace Model '<Model_Name>'
TRANSFORM (<select_list>) -- Optional
Options
(Model_type = 'PCA',
***NUM_PRINCIPAL_COMPONENTS = <Number>')** AS*
Select input_feature1, input_feature2,…….n column from <Source_table_name> "

Note:
NUM_PRINCIPA_COMPONENTS option is the number of final features to keep. This option's values cannot be larger than the total number of rows or the total feature cardinalities.

5. Advantage of BigQuery ML (BQ ML)

1. Integration: BigQuery ML integrates seamlessly with other Google Cloud Platform services, such as Google Data Studio and Google Cloud Storage, making it easy to perform end-to-end machine learning workflows.
2. Cost-effectiveness: BigQuery ML offers a pay-as-you-go pricing model, allowing users to only pay for the resources they use, making it a cost-effective option for machine learning tasks.
3. Scalability: BigQuery ML can handle massive datasets without requiring users to set up and manage a separate machine learning infrastructure.
4. Automated preprocessing: BigQuery ML automatically handles data preprocessing tasks such as feature transformations and missing value imputation,

simplifying the machine learning workflow.

5. Customizability: BigQuery ML allows users to specify custom models and transformations, providing flexibility and control over the machine learning process.
6. Real-time predictions: BigQuery ML supports batch and online prediction, enabling you to make real-time predictions on new data without exporting the model to a separate prediction service.
7. Automated feature engineering: BigQuery ML automatically performs feature engineering and feature selection to prepare the data for training. This saves data scientists time and effort and helps reduce the risk of overfitting.
8. Integration with SQL: BigQuery ML provides a familiar SQL interface for data scientists and analysts to build and deploy machine learning models without needing to learn a new programming language or platform.
9. Ease of use: BigQuery ML provides a SQL-like interface, making it accessible to data analysts and SQL users without requiring them to learn a new programming language.

6. Conclusion

BigQuery ML is a machine learning platform built on top of Google's BigQuery data warehouse. It allows users to build and train machine learning models directly in BigQuery using standard SQL syntax, eliminating the need for data movement or specialized machine learning tools and allowing users with experience in SQL to quickly get started with machine learning without having to learn a new programming language. With BigQuery ML, users can perform a range of machine learning tasks, including linear regression, logistic regression, k-means clustering, and more. The tool also provides automatic preprocessing and feature selection and the ability to perform custom preprocessing and feature engineering. Additionally, BigQuery ML offers model explainability, hyperparameter tuning capabilities, and integration with other Google Cloud services for data storage, analysis, and visualization. Overall, BigQuery ML provides a fast, scalable, and easy-to-use solution for performing machine learning tasks directly in BigQuery, without the need for additional software or data movement.

References

[1] What is BigQuery ML? [Online]. Available: https://cloud.google.com/bigquery/docs/bqml-introduction

[2] BigQuery Pricing. [Online]. Available: https://cloud.google.com/bigquery/pricing#queries

[3] Hyperparameter Tuning for CREATE MODEL Statement. [Online]. Available: https://cloud.google.com/bigquery/docs/reference/standard-sql/bigqueryml-hyperparameter-tuning#hyperparameters_and_objectives

[4] Sikender Mohsienuddin Mohammad, “Cloud Computing in IT and How It’s Going to Help United States Specifically,” *International Journal of Computer Trends and Technology (IJCTT)*, vol. 67, no. 10, 2019. [Google Scholar] [Publisher Link]

[5] Haleem Khan, and Yu Jiong, “Cloud Computing Effect on Enterprises in Terms of Cost,” *International Journal of Computer Trends and Technology (IJCTT)*, vol. 67, no. 5, pp. 14-19, 2019. [CrossRef] [Publisher Link]

[6] Etene Yonah, and Josphat M. Karani, “Performance in Layered Software Architectures: The Case of Customized Organizational Software,” *International Journal of Computer Trends and Technology (IJCTT)*, vol. 67, no. 12, pp. 15-19, 2019. [CrossRef] [Publisher Link]

[7] G. Anitha et al., “A Survey of Security Issues in IIOT and Fault Identification using Predictive Analysis in Industry 4.0,” *International Journal of Engineering Trends and Technology*, vol. 70, no. 12, pp. 99-108, 2022. [CrossRef] [Publisher Link]

[8] F. Twinkle Graf, and P. Prema, “Secure Collaborative Privacy in Cloud Data with Advanced Symmetric Key Block Algorithm,” *SSRG International Journal of Computer Science and Engineering*, vol. 2, no. 2, pp. 40-44, 2015. [CrossRef] [Publisher Link]

[9] Surendra Kumar Reddy Koduru, “Prediction of Severity of an Accident Based on the Extent of Injury using Machine Learning,” *International Journal of Computer Trends and Technology*, vol. 70, no. 9, pp. 43-49, 2022. [CrossRef] [Publisher Link]

[10] Satyanarayan Raju Vadapalli, “Monitoring the Performance of Machine Learning Models in Production,” *International Journal of Computer Trends and Technology*, vol. 70, no. 9, pp. 38-42, 2022. [CrossRef] [Publisher Link]

[11] Ravi Kashyap, “Data Sharing, Disaster Management, and Security Capabilities of Snowflake a Cloud Datawarehouse,” *International Journal of Computer Trends and Technology*, vol. 71, no. 2, pp. 78-86, 2023. [CrossRef] [Publisher Link]