ORACLE

Oracle Analytics ML Algorithms – Understanding parameters of Naive Bayes for Classification

10.05.2024, Version 1

Public

Oracle Analytics ML Algorithms – Understanding parameters of Naive Bayes for Classification

Oracle Analytics offers a broad set of native algorithms for training and applying a variety of machine learning models, such as classification, regression and clustering. It supports both Supervised as well as Unsupervised Learning. For an introduction on Machine Learning algorithms available in Oracle Analytics, please refer to the blog - Oracle Analytics ML Algorithms: Make Sense of Your Big Data

In this blog, we will look at one of the algorithm – Naive Bayes and understand the various parameters available when training a model that uses this Algorithm. Naive Bayes for Classification is an example of Supervised Learning and is available for both Binary as well as Multi Classification scenarios.

Classification is a machine learning technique that assigns items in a collection to target categories or classes. The goal of classification is to accurately predict the target class for each case in the data. The simplest type of classification problem is binary classification. In binary classification, the target attribute has only two possible values: for example, high credit rating or low credit rating. Multiclass targets have more than two values: for example, low, medium, high, or unknown credit rating.

Naive Bayes algorithm is based on conditional probabilities. It uses Bayes' theorem, a formula that calculates a probability by counting the frequency of values and combinations of values in the historical data. Bayes' theorem finds the probability of an event occurring given the probability of another event that has already occurred. If B represents the dependent event and A represents the prior event, Bayes' theorem can be stated as follows.

Prob (B given A) = Prob (A and B)/Prob (A)

To calculate the probability of B given A, the algorithm counts the number of cases where A and B occur together and divides it by the number of cases where A occurs alone.

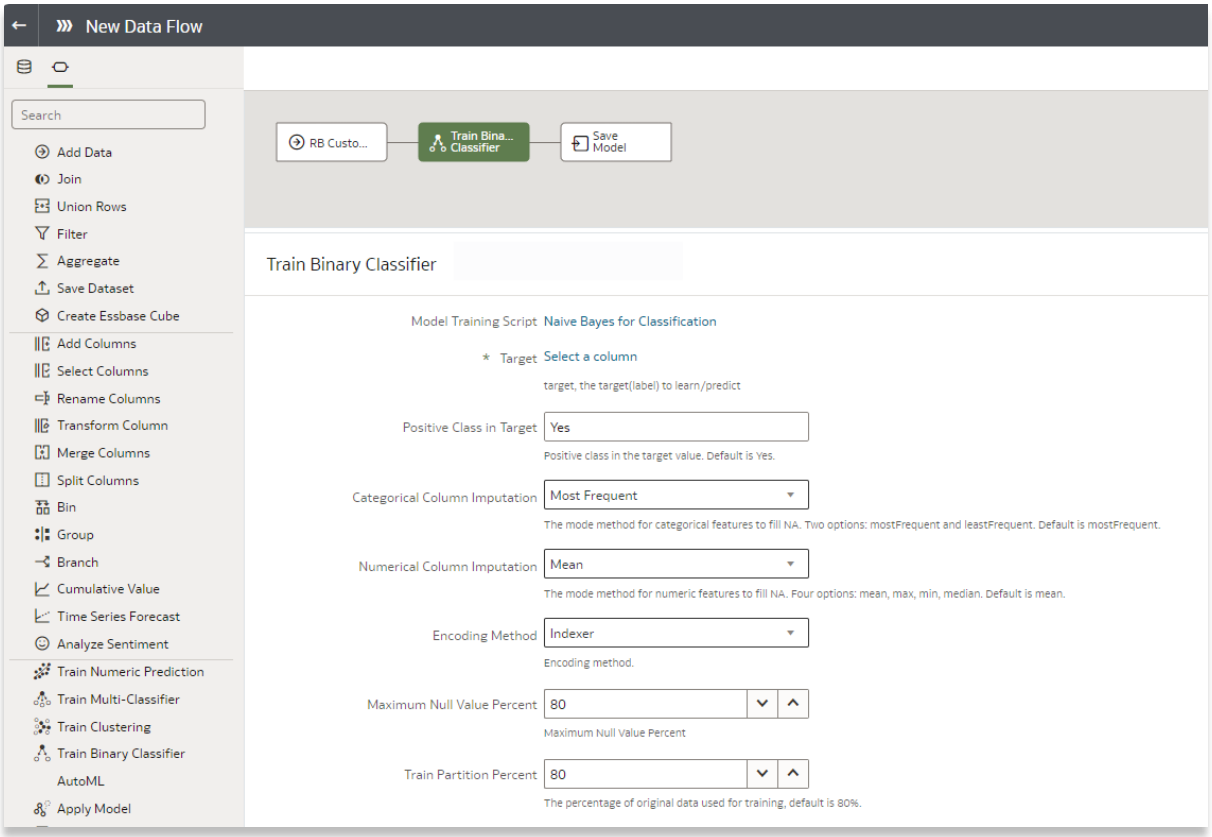
Naive Bayes assumes that each predictor is conditionally independent of the others. For a given target value, the distribution of each predictor is independent of the other predictors.

Advantages of Naive Bayes

1. The Naive Bayes algorithm affords fast, highly scalable model building and scoring. It scales linearly with the number of predictors and rows.
2. The build process for Naive Bayes supports parallel execution. (Scoring supports parallel execution irrespective of the algorithm.)
3. Naive Bayes can be used for both binary and multiclass classification problems.

With a basic introduction of Naive Bayes, let us understand the various parameters used in the creation of a Naive Bayes Classification Model in Oracle Analytics.

Parameters of Naive Bayes Algorithm



Target

The column for which we are predicting, by building the ML Model.

Positive Class in Target

This parameter allows us to specify a value of the Target column that is interpreted as a positive class in the target. These values can vary according to datasets. It could be "Yes" (When values are “Yes” and “No”, "1" (When values are “1” and) "0" Default is “Yes”. In multi classification, say an example of Sentiment column, possible values can be Positive, Negative and Neutral. In this case, “Positive” can be set as the positive class.

Categorical Column Imputation

This parameter allows one to specify how to handle, NA or NULL values in the dataset. When there are columns with NA or NULL values, they may want to impute/replace those columns with valid values. For Categorical Column Imputation, either Most Frequent or Least Frequent value in the column can replace NA or NULL values. Default is “**Most Frequent**”.

Ex: A Categorical column “Fruit” has values – Apple, Banana, Grapes. Apple is the most frequent value and Grapes is the least frequent value in the column. In this case, if we select Most Frequent, then the rows where the value for this column (Fruit) is NA or NULL, it is replaced with Apple.

Numerical Column Imputation

This parameter allows one to specify how to handle, NA or NULL values in the dataset. When there are columns with NA or NULL values, they may want to impute/replace those columns with valid values. For Numerical Column Imputation, either **Mean, Maximum, Minimum** or **Median** Values of that column can replace NA or NULL values. Default is "**Mean**".

Ex: A Numerical column, let's say "Level" has four values – 1,2,3,4 and let us say the Mean is 3, Median is 2, Maximum is 4 and Minimum is 1, for this column. In this case, if we select Mean, then the rows where the value for this column (Fruit) is NA or NULL, it is replaced with 3.

Encoding Method

Two methods are available for Encoding – Indexer and Onehot.

Indexer – In this method, the input values are indexed.

Ex: Let us say a categorical column "Fruit" has values – "Apple", "Banana" and "Grapes". Each of these values will be indexed and coded as integers, Apple – 1, Banana – 2 and Grapes – 3. After this encoding, the processing of the data continues using these encoded values.

Onehot – In this method, each value of the categorical column is converted into a new column with possible values of 0 and 1.

Ex: Let us say a categorical column "Fruit" has values – "Apple", "Banana" and "Grapes". Each of these values will become a column. Therefore, there would be now 3 columns: Fruit_Apple, Fruit_Banana and Fruit_Grapes. If, for a row, say the value of fruit is Apple, then the Fruit_Apple column shall have 1 as the encoded value and the other two columns shall have 0 as the encoded value.

Maximum Null Value Percent

This parameter is configured to check the number of allowed NULL values in the dataset. A value between 1 and 100 can be configured. Default is 80.

The NULL value percent is calculated for each column and is calculated in the following way –

Number of rows with NULL as value for a column / Total number of rows in the dataset

Train Partition Percent

When a model is being created, the input dataset is divided into 2 parts – train and test data. The train data is used to build the ML model and the test data is used to test the accuracy of the model that is built using the train data. To configure the split between this train and test data within the input dataset, this parameter is provided. It determines the percent of input data that is split between train and test data. A value between 10 and 100 can be set. Default is 80.

ML Model Building

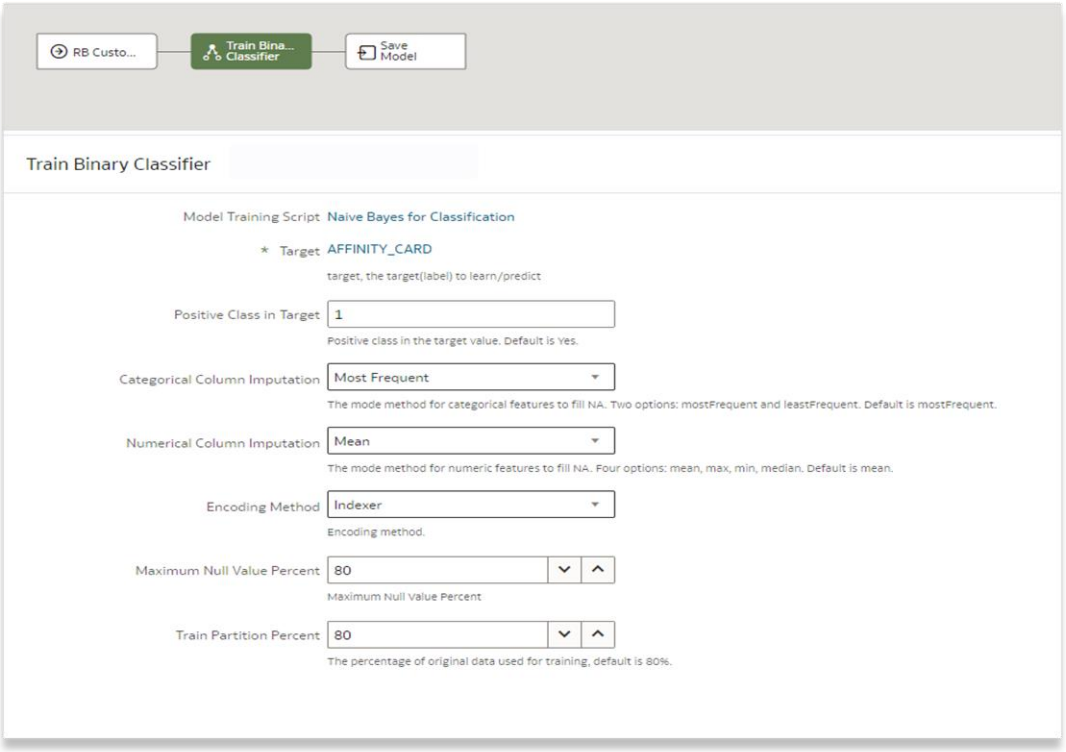
Now that we have the understanding of the parameters for Naïve Bayes for Classification, let's look at an example where we are trying to predict whether a customer would buy an Affinity Card or not. The input dataset has customer demographic data and values for Affinity Card column. The values are 0 and

1. 0 means the customer did not buy the Affinity Card and 1 means the customer did buy the Affinity Card.

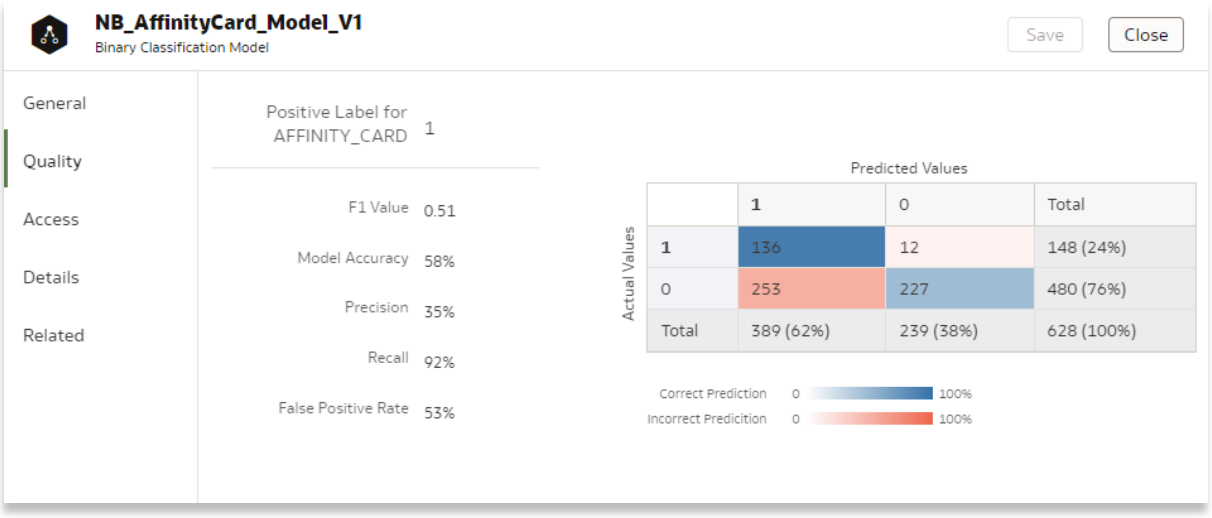
The high-level steps to build and apply a Machine Learning Model in Oracle Analytics are –

1. Identify a sample data (representative data of actual data)
2. Train and build an ML Model using this sample data
3. Inspect the quality of the ML model
4. Tune it, to improve the quality of the ML model, if necessary.
5. Once the ML model is satisfactory, then use it and apply it to actual data (historical).
6. Verify the outcome and if required, tune the ML model further or deploy (schedule) it for new/future data.

For **Iteration 1**, let us specify the Target column as AFFINITY_CARD and the Positive Class in target as 1. Rest of the parameters can be left to default values. Once the dataflow is run, the ML model is built.



Let us check the quality of the ML model built.



These quality metrics are generated based on the testing the ML Model on the test data (Train partition percent) of the input test data.

The first iteration shows a Model Accuracy of 58%, with False Positive Rate of 53%, Precision of 35% and Recall of 92%.

Definitions of parameters –

True Positive	- No. of Outcome(s) where the model correctly predicts the positive class.
True Negative	- No. of Outcome(s) where the model correctly predicts the negative class.
False Positive	- No. of Outcome(s) where the model incorrectly predicts the positive class,
False Negative	- No. of Outcome(s) where the model incorrectly predicts the negative class.
Model accuracy	- (True Positive + True Negative)/Total
Precision	- (True Positive)/ (True Positive + False Positive)
Recall	- (True Positive)/ (True Positive + False Negative)
False Positive Rate	- (False Positive)/ (Actual Number of Negatives)

F1 Score = 2 * (Precision * Recall) / (Precision + Recall)

For the above ML model built, the calculations would be, as follows:

True Positive	136
True Negative	227
False Positive	253
False Negative	12
Total Outcomes	628

Model Accuracy = ((136 + 227) / 628) = 0.578 that is 58% Accuracy

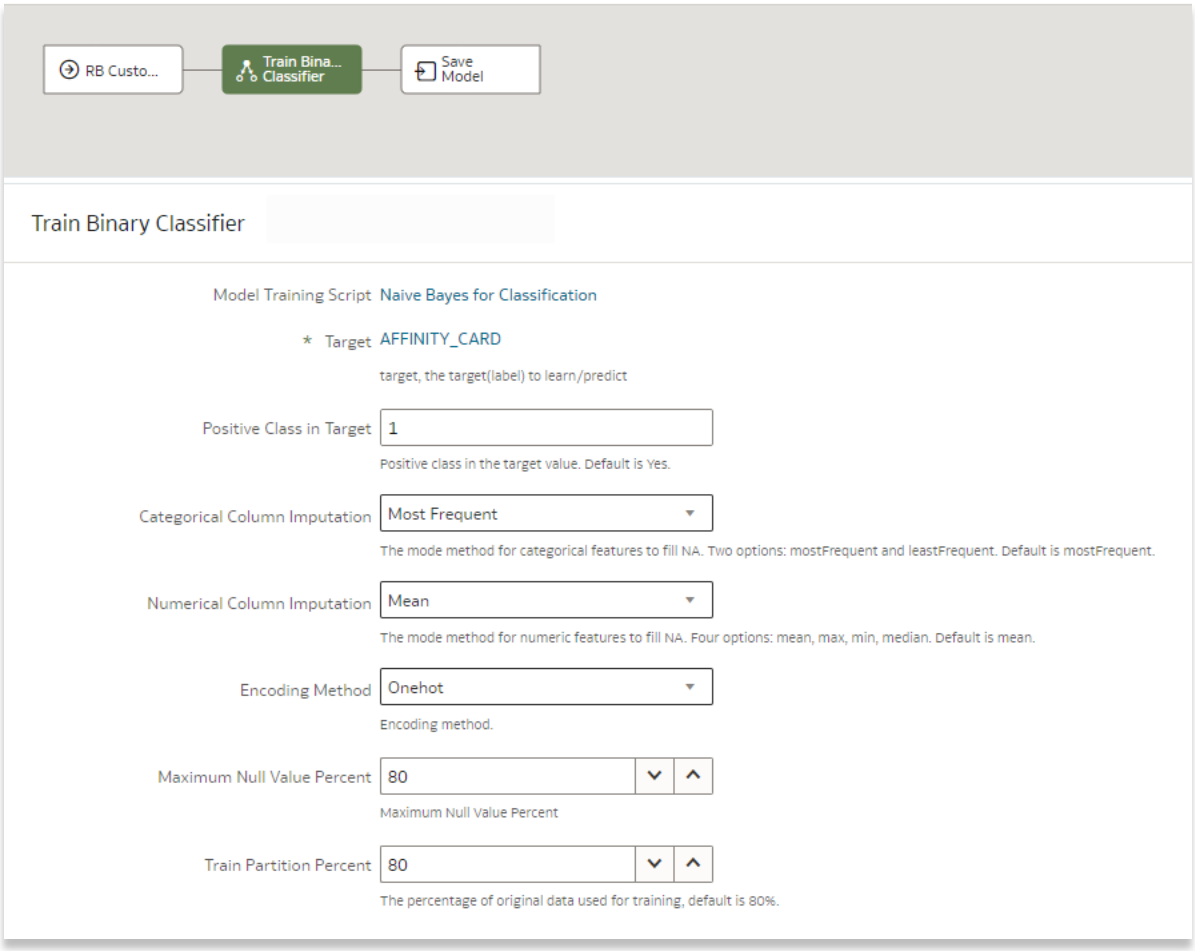
Precision = ((136) / (136 + 253)) = 0.3496 that is 35% Precision

Recall = ((136) / (136 + 12)) = 0.9189 that is 92% Recall

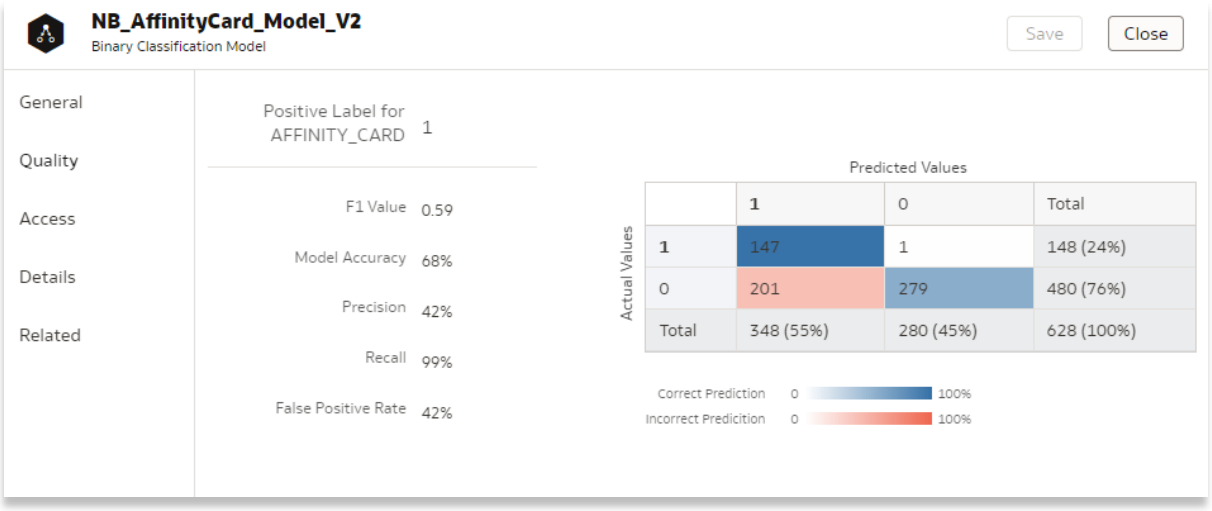
False Positive Rate = ((253) / (480)) = 0.5271 that is 53 % False Positive Rate

F1 Score = 2 * (0.3496 * 0.9189) / (0.3496 + 0.9189) = 0.5064997 that is 0.51 F1 Score.

For **Iteration 2**, let us change the Encoding Method to Onehot.



Let us check the quality of the ML model built.

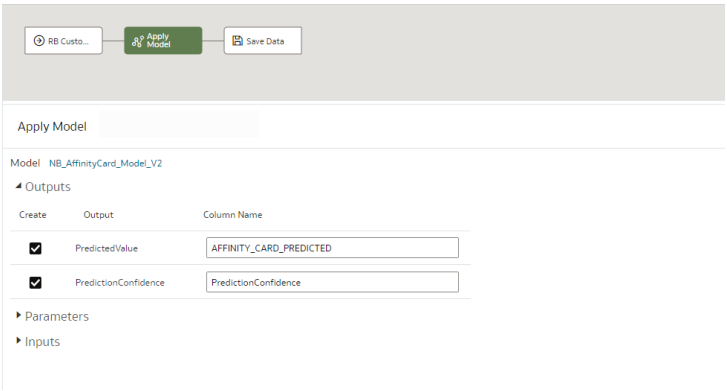


As you can see, the Model Accuracy is increased from 58% to 68% and even the False Positive Rate is reduced from 53% to 42%.

In this way, you can modify the parameters and look at improving the accuracy and other quality metrics of the ML Model.

However, please note that Naive Bayes has a very limited parameter set and so the options for Hyper-parameter tuning are very limited. The ML model accuracy can be boosted by other methods such as data pre-processing and feature selection. Handling Missing values, Encoding Method, Removing Correlated features, are some of the examples.

Once you are satisfied with the ML model quality, you can now apply the model to any dataset by using the apply node in data flows.



Reference – Oracle Machine Learning Basics Documentation (https://docs.oracle.com/en/database/oracle/machine-learning/oml4sql/21/dmapi/naive-bayes.html#GUID-BB77D68D-3E07-4522-ACB6-FD6723BDA92A)

Blog - Oracle Analytics ML Algorithms: Make Sense of Your Big Data