Vertica Machine Learning V8.1.1 Cheat Sheet

Vertica Machine Learning supports the whole workflow of machine learning via a SQL interface. To learn the full capability of Vertica ML, go to my.vertica.com/documentation. Example data sets used in the cheat sheet are available on github.com/vertica/Machine-Learning-Examples.

A basic example

```
=> CREATE TABLE iris (id int, sepal_length float, sepal_width float, petal_length float, petal_width float, species varchar(10));
=> COPY iris FROM LOCAL 'iris.csv' DELIMITER ',' ENCLOSED BY '"' SKIP 1;
=> CREATE TABLE iris_test AS SELECT * FROM iris TABLESAMPLE(25);
=> CREATE TABLE iris_train AS (SELECT * FROM iris EXCEPT SELECT * FROM iris_test);
=> SELECT rf_classifier('myRFModel', 'iris_train', 'species', 'sepal_length, sepal_width, petal_length, petal_width' USING PARAMETERS ntree=100, sampling_size=0.3); --train a model to predict species
=> CREATE TABLE iris_prediction AS SELECT species, predict_rf_classifier(sepal_length, sepal_width, petal_length, petal_width USING PARAMETERS model_name='myRFModel') AS predicted FROM iris_test;
=> SELECT confusion_matrix(CASE species WHEN 'setosa' THEN 2 WHEN 'versicolor' THEN 1 ELSE 0 END, CASE predicted WHEN 'setosa' THEN 2 WHEN 'versicolor' THEN 1 ELSE 0 END USING PARAMETERS
   num_classes=3) OVER() FROM iris_prediction;
```

Preprocessing the data

Detect outliers

```
=> SELECT detect_outliers('baseball_outliers', 'baseball_roster', '*', 'robust_zscore' USING PARAMETERS outlier_threshold=3.0, exclude_columns='id, last_name');
```

Normalize

```
=> SELECT normalize('mtcars_normz', 'mtcars', 'wt, hp', 'zscore'); --output a view ‘mtcars_normz’
=> SELECT normalize_fit('mtcars_normfitrz', 'mtcars', 'wt,hp', 'robust_zscore'); --store normalization parameters in a model 'mtcars_normfitrz'
=> SELECT apply_normalize(wt, hp USING PARAMETERS model_name = 'mtcars_normfitrz') FROM mtcars; --apply the normalization parameters to ‘mtcars’
=> SELECT reverse_normalize(wt, hp USING PARAMETERS model_name = 'mtcars_normfitrz') FROM mtcars; --reverse the normalization in ‘mtcars’
```

Impute missing values

```
=> SELECT impute ('myImputedView', 'small_input_impute', 'x1,x2, x3', 'mean' USING PARAMETERS partition_columns='pclass,gender'); --impute the missing value for each cluster independently
```

Process imbalance data

```
=> SELECT balance ('myOutputView', 'small_input_impute', 'gender', 'over_sampling' USING PARAMETERS sampling_ratio=1.0); --make the sample size even between ‘male’ and ‘female’ samples
```

Sample

```
=> CREATE TABLE baseball_sample AS SELECT * FROM baseball TABLESAMPLE(25); --generate a 25% sample set randomly
```

Training and predicting

Regression

Linear Regression

```
=> SELECT linear_reg('myLinearRegModel', 'faithful_train', 'eruptions', 'waiting' USING PARAMETERS optimizer='BFGS', regularization='L2'); --use ‘waiting’ time to predict the duration of the next ‘eruption’
=> SELECT id, predict_linear_reg(waiting USING PARAMETERS model_name='myLinearRegModel') FROM faithful_test;
```

Support Vector Machines (SVM)

```
=> SELECT svm_regressor('mySvmRegModel', 'faithful_train', 'eruptions', 'waiting' USING PARAMETERS error_tolerance=0.1, max_iterations=100);
=> SELECT id, predict_svm_regressor(waiting USING PARAMETERS model_name='mySvmRegModel') FROM faithful_test;
```

Classification

Logistic Regression

```
=> SELECT logistic_reg('myLogisticRegModel', 'mtcars_train', 'am','mpg, cyl, disp, hp, drat, wt, qsec, vs, gear, carb' USING PARAMETERS exclude_columns='hp', optimizer='BFGS', regularization='L2');
    --train a model to predict if a car has automatic or manual transmission
=> SELECT car_model, predict_logistic_reg(mpg, cyl, disp, drat, wt, qsec, vs, gear, carb USING PARAMETERS model_name='myLogisticRegModel') FROM mtcars_test;
```

Support Vector Machines (SVM)

```
=> SELECT svm_classifier('mySvmClassModel', 'mtcars_train', 'am', 'mpg,cyl,disp,hp,drat,wt,qsec,vs,gear,carb' USING PARAMETERS exclude_columns='hp,drat');
=> SELECT car_model, predict_svm_classifier(mpg,cyl,disp,wt,qsec,vs,gear,carb USING PARAMETERS model_name='mySvmClassModel') FROM mtcars_test;
```

Naive Bayes

```
=> SELECT naive_bayes('naive_house84_model', 'house84_train', 'party', '*' USING PARAMETERS exclude_columns='party, id'); --train a model to predict a person's party association
=> SELECT party, predict_naive_bayes(vote1, vote2, vote3 USING PARAMETERS model_name='naive_house84_model', type='response') AS predicted_party FROM house84_test;
=> SELECT predict_naive_bayes_classes(id, vote1, vote2, vote3 USING PARAMETERS model_name='naive_house84_model', key_columns='id', exclude_columns='id', classes='democrat, republican',
   match_by_pos='false') OVER() FROM house84_test; --return the probability of the predicted class and the specified class 'democrat' and 'republican'
```

Random Forest

```
=> SELECT rf_classifier('myRFModel', 'iris_train', 'species', 'sepal_length, sepal_width, petal_length, petal_width' USING PARAMETERS ntree=100, sampling_size=0.3);
=> SELECT id, predict_rf_classifier(sepal_length, sepal_width, petal_length, petal_width USING PARAMETERS model_name='myRFModel') FROM iris_test;
=> SELECT predict_rf_classifier_classes(id, sepal_length, sepal_width, petal_length, petal_width USING PARAMETERS model_name='myRFModel', key_columns ='id', exclude_columns='id') OVER () FROM iris_test;
    --return the probability of the predicted class
```

Clustering

K-means

```
=> SELECT kmeans('myKmeansModel', 'iris', '*', 5  USING PARAMETERS max_iterations=20, key_columns='id', exclude_columns='species, id'); --cluster iris records into groups
=> SELECT id, apply_kmeans(sepal_length, 2.2, 1.3, petal_width USING PARAMETERS model_name='myKmeansModel', match_by_pos='true') FROM iris;
```

Evaluating model performance

Regression metrics

Mean Squared Error

```
=> SELECT mse(obs, pred) OVER() FROM (SELECT eruptions AS obs, PREDICT_LINEAR_REG (waiting USING PARAMETERS model_name='myLinearRegModel') AS pred FROM faithful_testing) AS prediction_output;
```

R Squared

```
=>SELECT rsquared(obs, pred) OVER() FROM (SELECT eruptions AS obs, PREDICT_LINEAR_REG (waiting USING PARAMETERS model_name='myLinearRegModel') AS pred FROM faithful_testing) AS prediction_output;
```

Classification metrics

Confusion Matrix

```
=> SELECT confusion_matrix(obs::int, pred::int USING PARAMETERS num_classes=2) OVER() FROM (SELECT am AS obs, predict_logistic_reg(mpg, cyl, disp, drat, wt, qsec, vs, gear, carb
   USING PARAMETERS model_name='myLogisticRegModel')::INT AS pred FROM mtcars) AS prediction_output;
```

Error Rate

```
=> SELECT error_rate(obs::int, pred::int USING PARAMETERS num_classes=2) OVER() FROM (SELECT am AS obs, predict_logistic_reg(mpg, cyl, disp, drat, wt, qsec, vs, gear, carb
   USING PARAMETERS model_name='myLogisticRegModel', type='response') AS pred FROM mtcars) AS prediction_output;
```

Lift Table

```
=> SELECT lift_table(obs::int, prob USING PARAMETERS num_bins=2) OVER() FROM (SELECT am AS obs, predict_logistic_reg(mpg, cyl, disp, drat, wt, qsec, vs, gear, carb
   USING PARAMETERS model_name='myLogisticRegModel', type='probability') AS prob FROM mtcars) AS prediction_output;
```

ROC

```
=> SELECT roc(obs::int, prob USING PARAMETERS num_bins=2) OVER() FROM (SELECT am AS obs, predict_logistic_reg(mpg, cyl, disp, drat, wt, qsec, vs, gear, carb
   USING PARAMETERS model_name='myLogisticRegModel', type='probability') AS prob FROM mtcars) AS prediction_output;
```

Managing models

List models

```
=> SELECT * FROM models;
```

Delete a model

```
=> DROP MODEL myLinearRegModel;
```

Rename, change owner and change schema of a model

```
=> ALTER MODEL myKmeansModel OWNER TO user1;
=> ALTER MODEL myKmeansModel SET SCHEMA public;
=> ALTER MODEL myKmeansModel RENAME to myKmeans;
```

Summarize a model

```
=> SELECT summarize_model('myLinearRegModel');
```

Read model attributes

```
=> SELECT get_model_attribute(USING PARAMETERS model_name='myLinearRegModel'); --list all attributes in the model
=> SELECT get_model_attribute(USING PARAMETERS model_name='myLinearRegModel', attr_name='data'); --return the value for attribute 'data'
```