Vertica ML Python Workshop

Exercise 10: ML Clustering

Ouali Badr

December 6, 2019

Executive Summary

> *"Science knows no country, because knowledge belongs to humanity, and is the torch which illuminates the world."*
>
> **Louis Pasteur**

VERTICA ML PYTHON allows the users to use Vertica advanced analytics and Machine Learning with a Python front-end Interface. In this exercise, you'll learn some basics to begin your fantastic Data Science Journey with the API. As a summary:

- Create a KMeans Model
- Find the K of the KMeans
- Add the prediction in the Virtual Dataframe
- Evaluate the model

Contents

1 Presentation

Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups (clusters). We already did some clustering in exercise 2. It was mainly focused on Outliers and Anomaly Detections rather than doing a complete segmentation. We will use exercise 2 functions to segment flowers in the Iris dataset.

2 Functions used during the Exercise

2.1 KMeans

Create a KMEANS object by using the Vertica Highly Distributed and Scalable KMEANS directly on the data.

initialization

Library: vertica_ml_python.learn.cluster

```
class KMEANS(
  name: str,
  cursor,
  n_cluster: int = 8,
  init = "kmeanspp",
  max_iter: int = 300,
  tol: float = 1e-4)
```

Parameters

- **name:** *<str>*
  Name of the the model. The model is stored in the DB.
- **cursor:** *<object>*
  DB cursor.
- **n_cluster:** *<int>*, optional
  Number of clusters.
- **init:** *<str or list>*, optional
  The method used to find the initial cluster centers. The method can be in {random | kmeanspp}.
  It can be also a list with the initial cluster centers to use.
- **max_iter:** *<int>*, optional
  The maximum number of iterations the algorithm performs.
- **tol:** *<float>*, optional
  Determines whether the algorithm has converged. The algorithm is considered converged after no center has moved more than a distance of 'tol' from the previous iteration.

Methods

The KMeans object has four main methods:

```
# Add the cluster prediction in a vDataframe
def add_to_vdf(self, vdf, name: str = "")

# Drop the model
def drop(self)

# Fit the model with the input columns
def fit(self, input_relation: str, X: list)

# Plot the model (only possible if the number of columns <= 3)
def plot(self)
```

Attributes

The KMEANS object has two main attributes:

```
self.cluster_centers # Position of the cluster centers
self.metrics # Different metrics to evaluate the model
```

Example

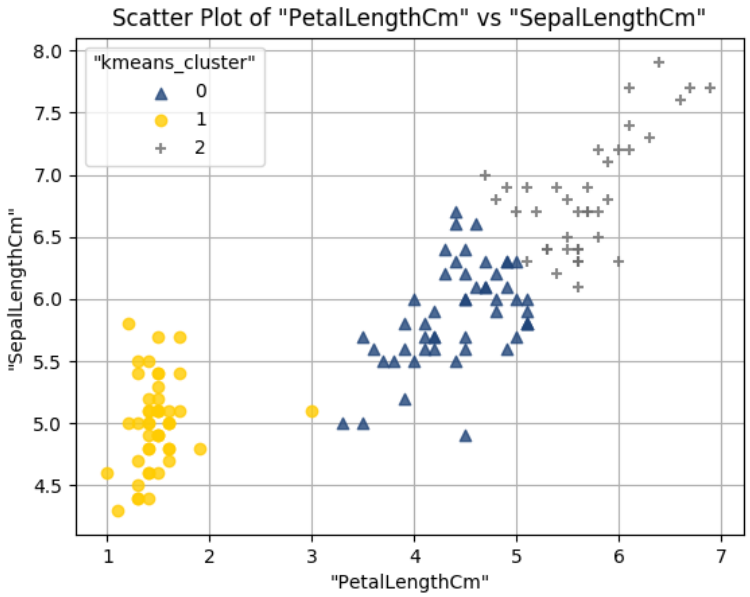
```
from vertica_ml_python.learn.cluster import KMeans

# We can build the model
model = KMeans("kmeans_iris", cur, n_cluster = 3)
model.fit("iris", ["PetalLengthCm", "SepalLengthCm"])

# We can see the different metrics relative to the model
model.metrics

# Output
                                                    value
Between-Cluster Sum of Squares                  512.23072
Total Sum of Squares                            566.03207
Total Within-Cluster Sum of Squares              53.801351
Between-Cluster SS / Total SS           0.9049499969144859
converged                                             True

# And Plot the model
model.plot()
```



2.2 best_k

Library: vertica_ml_python.learn.cluster.model_selection

```
def best_k(
      X: list,
      input_relation: str,
      cursor,
      n_cluster = (1, 100),
      init = "kmeanspp",
      max_iter: int = 50,
      tol: float = 1e-4,
      elbow_score_stop = 0.8)
```

Find the KMeans K by using a score threshold.

Parameters

- **X:** *<list>*
  List of the predictor columns.
- **input_relation:** *<str>*
  Relation used to train the model.
- **cursor:** *<object>*
  DB cursor.
- **n_cluster:** *<tuple>*, optional
  Tuple representing the number of cluster to start with and to end with.

- **init:** *<str or list>*, optional
  The method used to find the initial cluster centers. The method can be in {random | kmeanspp}.
  It can be also a list with the initial cluster centers to use.
- **max_iter:** *<int>*, optional
  The maximum number of iterations the algorithm performs.
- **tol:** *<float>*, optional
  Determines whether the algorithm has converged. The algorithm is considered converged after no center has moved more than a distance of 'tol' from the previous iteration.
- **elbow_score_stop:** *<float>*, optional
  Stop the Parameters Search when this Elbow score is reached.

Example

```
from vertica_ml_python.learn.model_selection import best_k

best_k(["PetalLengthCm", "PetalWidthCm", "SepalLengthCm", "SepalWidthCm"], "
    iris", cur)

# Output
3
```

3 Questions

Turn on Jupyter with the 'jupyter notebook' command. Start the notebook exercise10.ipynb and answer to the following questions.

- **Question 1:** Find the best K which will segment the data with an elbow score greater than 0.8
- **Question 2:** Use this K to create a KMeans model.
- **Question 3:** Add the prediction in the Virtual Dataframe. Draw an histogram of the Species and the predictions. What do you notice ?