Vertica ML Python Workshop

Exercise 12: ML Regression

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:

- Split the data
- Build a Linear Regression model
- Evaluate the model with different metrics
- Compute the model features importance

Contents

1 Presentation

Regression is a set of statistical processes for estimating the relationships between a dependent variable (often called the 'outcome variable') and one or more independent variables (often called 'predictors', 'covariates', or 'features?). Vertica ML Python has many regression algorithms already implemented (Linear Regression, Random Forest, SVM...). We will use a Linear Regression to estimate the wine quality of specific wines.

2 Functions used during the Exercise

2.1 ElasticNet

Create a ElasticNet object by using the Vertica Highly Distributed and Scalable Linear Regression on the data.

initialization

Library: vertica_ml_python.learn.linear_model

```
class ElasticNet(
    name: str,
    cursor,
    penalty: str = 'ENet',
    tol: float = 1e-4,
    C: float = 1.0,
    max_iter: int = 100,
    solver: str = 'CGD',
    l1_ratio: float = 0.5)
```

Parameters

- **name:** *<str>*
  Name of the model.
- **cursor:** *<object>*
  DB cursor.
- **penalty:** *<str>*, optional
  Determines the method of regularization: {None | L1 | L2 | ENet}
- **tol:** *<float>*, optional
  Determines whether the algorithm has reached the specified accuracy result.
- **C:** *<float>*, optional
  The regularization parameter value. The value must be zero or non-negative.
- **max_iter:** *<int>*, optional
  Determines the maximum number of iterations the algorithm performs before achieving the specified accuracy result.
- **solver:** *<int>*, optional
  The optimizer method used to train the model: {Newton | BFGS | CGD}
- **l1_ratio:** *<float>*, optional
  ENet mixture parameter that defines how much L1 versus L2 regularization to provide.

Methods

The ElasticNet object has many methods:

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

# Save a table or a view in the DB corresponding to the model predictions for
    all the classes
def deploy_to_DB(self, name: str, view: bool = True, cutoff = -1)

# Drop the model from the DB
def drop(self)

# Compute the importance of each feature
def features_importance(self)

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

# Plot the SVM if it is possible (The length of X must be lesser of equal to
    3)
def plot(self)

# Compute different metrics to evaluate the model
def regression_report(self)

# Compute the selected metric
def score(self, method: str = "r2")
```

Attributes

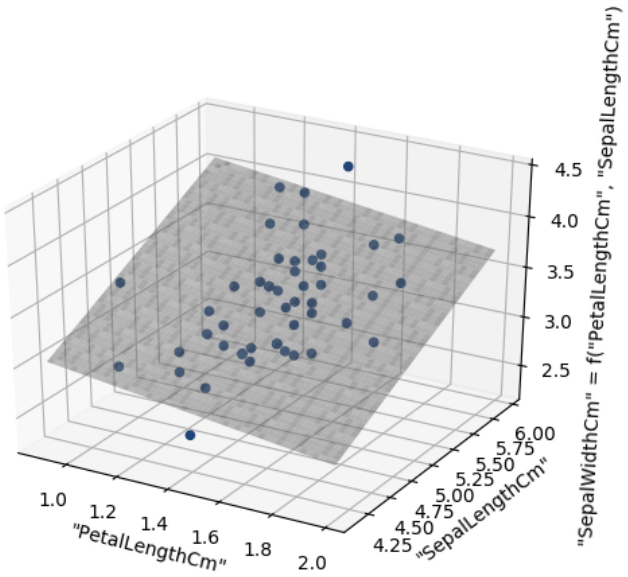
The ElasticNet object has only one attribute:

```
self.coef # Informations about the model coefficients
```

Example

```
from vertica_ml_python.learn.linear_model import ElasticNet

# We can build the model
model = ElasticNet("enet_iris", cur, penalty = "None", tol = 1e-8)
model.fit("iris", ["PetalLengthCm", "SepalLengthCm"], "SepalWidthCm")

# We can evaluate the model
model.regression_report()

# Output
                                    value
explained_variance         0.452452439026576
```

```
max_error                  0.861565208032992
median_absolute_error      0.203743266177453
mean_absolute_error        0.251888470089032
mean_squared_error         0.102254872043582
r2                         0.452452439026074

# We can also draw the model
model.plot()
```



3 Questions

Turn on Jupyter with the 'jupyter notebook' command. Start the notebook exercise12.ipynb and answer to the following questions.

- **Question 1:** Split the dataset into a training and a testing.
- **Question 2:** Create a Linear Regression model to rate the wines.
- **Question 3:** Look at the model coef attribute and see what features you should eliminate if you decide to build another Linear Regression model.
- **Question 4:** Look at the features importance and confirm the hypothesis.
- **Question 5:** Compute a regression report. What can you say about your model ? Which type of model you should probably consider to rate wines ?