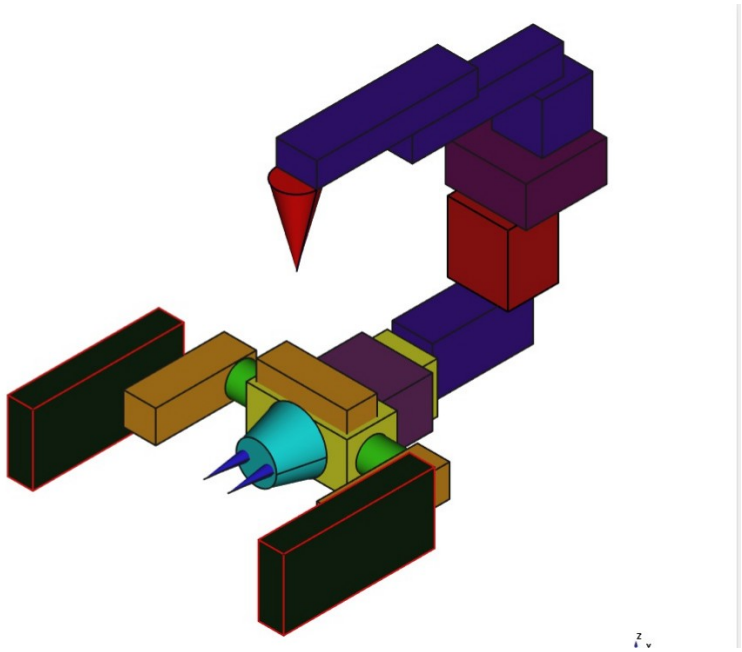
Scorpion 1.0.0 Report



A basic Python package having Machine Learning functionalities.

Calculus Corporation

Introduction

Scorpion in a python package, written entirely in python. All the functionalities included in this package, has been programmed from scratch, apart from the help of some standard packages, like numpy and matplotlib.

The package consists of four full-fledged modules, namely, regression, classification, cluster, and neural.

While regression module deals mainly with traditional regression, the other three modules provide functions for classifying large sets of data.

The detailed descriptions of these modules, alongwith the chief functions in them have been described in the following pages.

regression.py

This module provides all the functionality that someone would need for any type of linear or polynomial regression. Linear regression has been generalised to support any number of features, and the polynomial regression functionality allows users to exactly define the desired feature engineering. Some of the functions in this module are as follows:

- scale_down(x): This function scales down the input array passed into it, to a range of [-1,1]

- J_gen(x_train,y_train,w,b,lamb=0): This function returns the squared error cost function for the training set compared to the prediction of the model.

- R2score(x_train,y_train,w,b): This function evaluates the R2score of the prediction of the model.

- linear_gen(x_train,y_train,alpha=0.01,k=513,lamb=0): This is the core function of this

module, which performs linear regression upon the training set, and returns the resultant array of w-parameter, and b-parameter.

- check_alpha(x_train,y_train,alpha=0.01,k=513,lamb=0): This function checks whether the passed learning rate is appropriate for the regression of the dataset at hand.

- transform(x_train,order,combination=[]): This function transforms the input matrix, such that running linear regression on the transformed matrix is actually equvalent to running polynomial regression upon the input matrix.

- polynomial(x_train,y_train,order=5,alpha=0.01,k=513): This is a simple function, not much necessary, which holds together the functionality of polynomial regression.

classification.py

The classification module has some common functions with regression.py, but this mainly includes logistic and softmax regression. Some of its functions are as follows:

- sigmoid(z): This eveluates the sigmoid function for an array, or a single number z.

- J_softmax(x_train,y_train,theta): This returns the cost function for softmax regression, the cross-entropy cost function.

- linear_gen(x_train,y_train,alpha=0.01,k=513): This is the core function of this module, which performs logistic softmax regression upon the given dataset.

- Domain(y_train): This function returns the domain, or a list containing the number of unique values the passed array contains, in ascending order.

- polynomial(x_train,y_train,order=5,alpha=0.01, k=513): This function enables polynomial logistic regression upon the training dataset.

cluster.py

This module implements two of the famous clustering algorithms, namely, the K-means clustering algorithm, and the K-Nearest-Neighbours algotihm for classifying a new input. Some of its functiuons are:

- classify_kmeans(x_train,k=2): In this function, the second argument, k denotes the number of clusters. The classification is done on the basis of distance of the data-points from the respective centroids of the classes.

- k_nearest_neighbours(x_train,y_train,x_test,k=5): This function takes an already classified dataset as input, and predicts the output of x_test.

neural.py

This module implements neural networks, to solve complex classification problems.

Conclusion

This package was made entirely in python, and the programming was done in Spyder IDE on an Anaconda environment.

Jupyter notebook was also used to test scorpion.