**MDS 560 Week 5 Hands-On Accelerator**

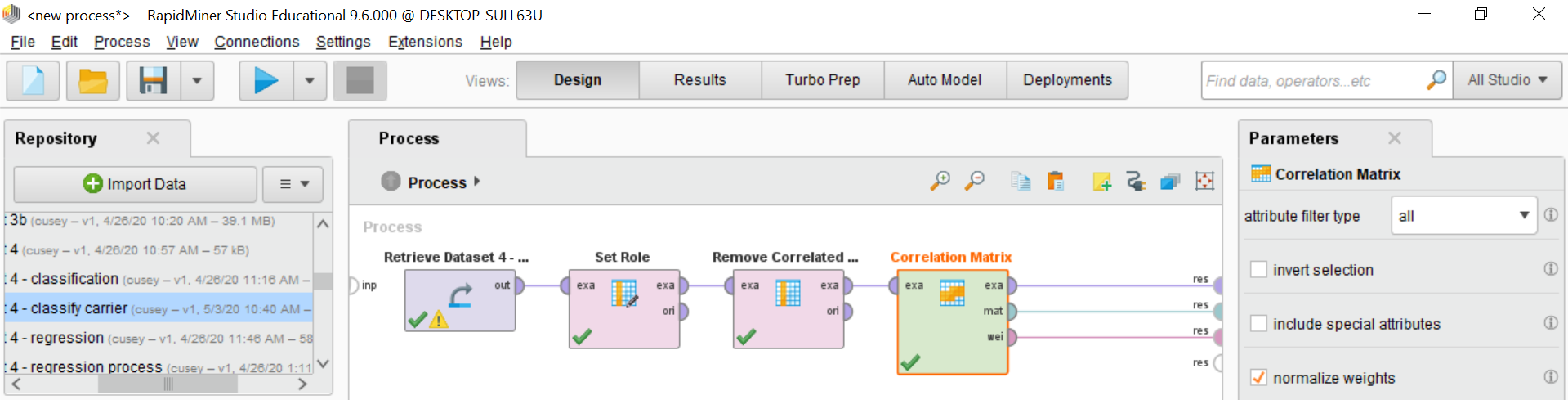
Your deliverables and hands-on activities for this week are:

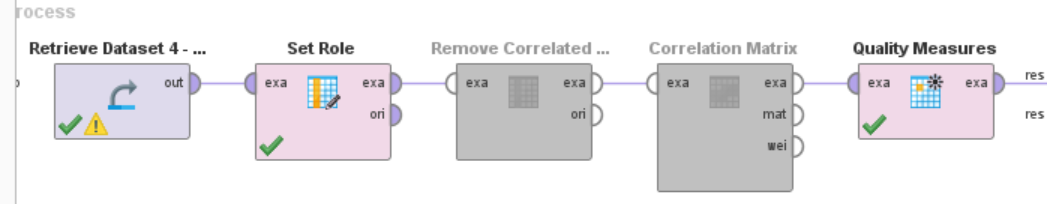
1. Re-Import Dataset 4 and perform predictive analytics for the label “Carrier” using Automodel . Be sure to select only those X variables that are actionable by the business. Compare the performance of the algorithms. Identify the factors that most influence the choice of carrier.

Results: To accomplish this problem, I decided to load in dataset 4 and preprocess it myself, save a new version of the file after reviewing correlations, stability, etc. After iteratively removing variables, running the new dataset through auto modeler, I found that the only variable that really has a big impact on the carrier is the base freight. Honestly, I’m not sure if base freight is talking about the rate to ship the product or if it is the weight of the product. If base freight represents the cost to ship, then it doesn’t tell us too much about what goes into deciding which Carrier to use. Instead, the base freight could represent indirectly the Carrier since Carrier A might be more expensive then Carrier B or C, etc.

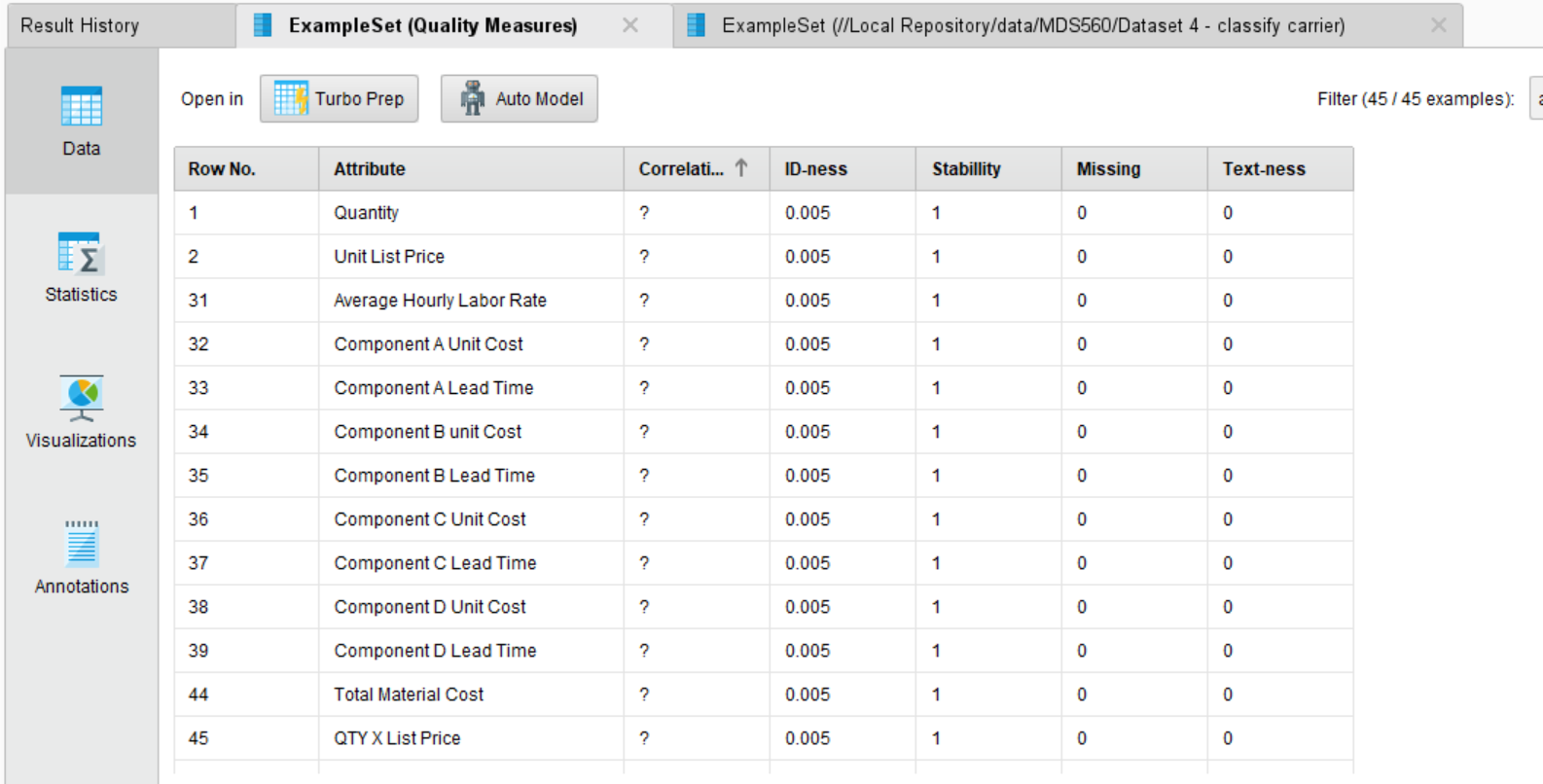
I decided to create an alternative model where I included features with a correlation higher than .2 but excluded the attributes I thought might just be associated to the cost of the carrier. As a result, I didn’t include base freight, total freight cost, and premium. I reviewed the Naïve Bayes Model to interpret the difference between the variables for Carrier A, B, and C. Some things I noticed is that when the ship date is in the 2nd quarter, Carrier B and C are more likely then Carrier A. Alternatively, quarter 1 is more likely to be CarrierA. If the Commit Date is on a Thursday, Carrier C is much more likely than Carrier B and Carrier C. It seems though that the more likely carriers vary frequently through the week for the commit date. If there is more time between the order date and ship date, it appears that CarrierC is more likely to be used.

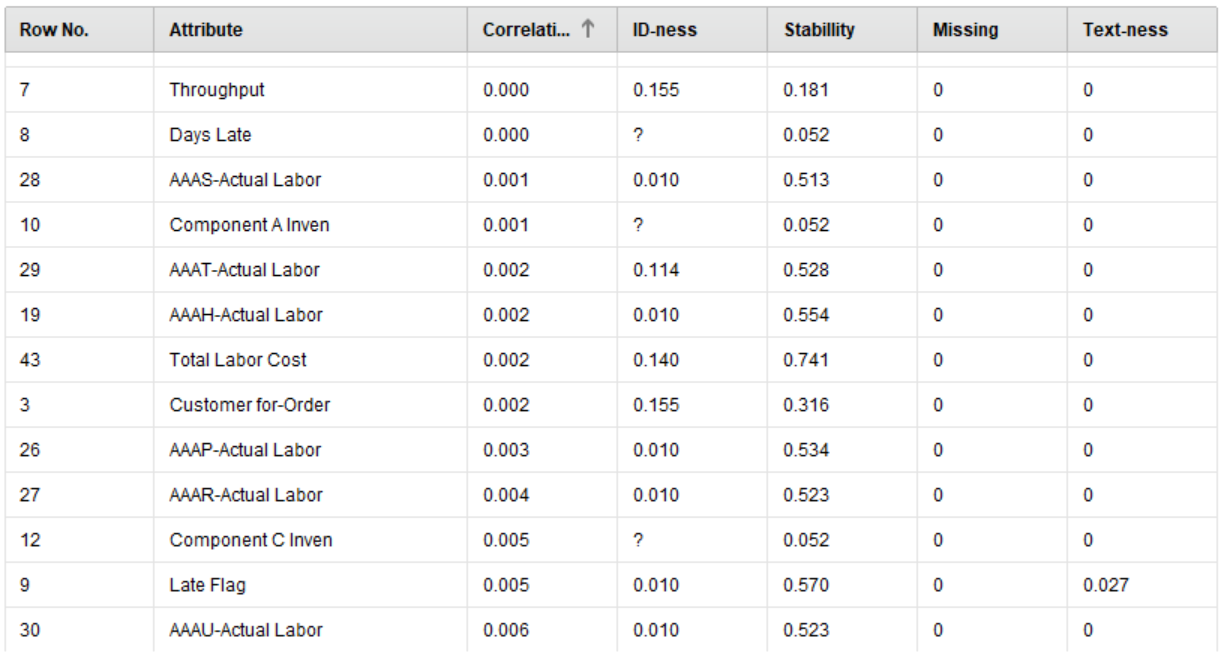
Pre-Processing Steps:

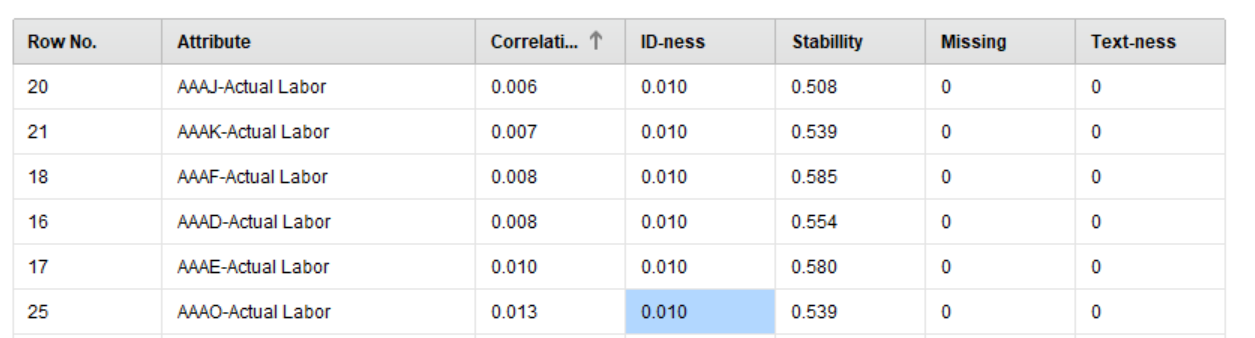




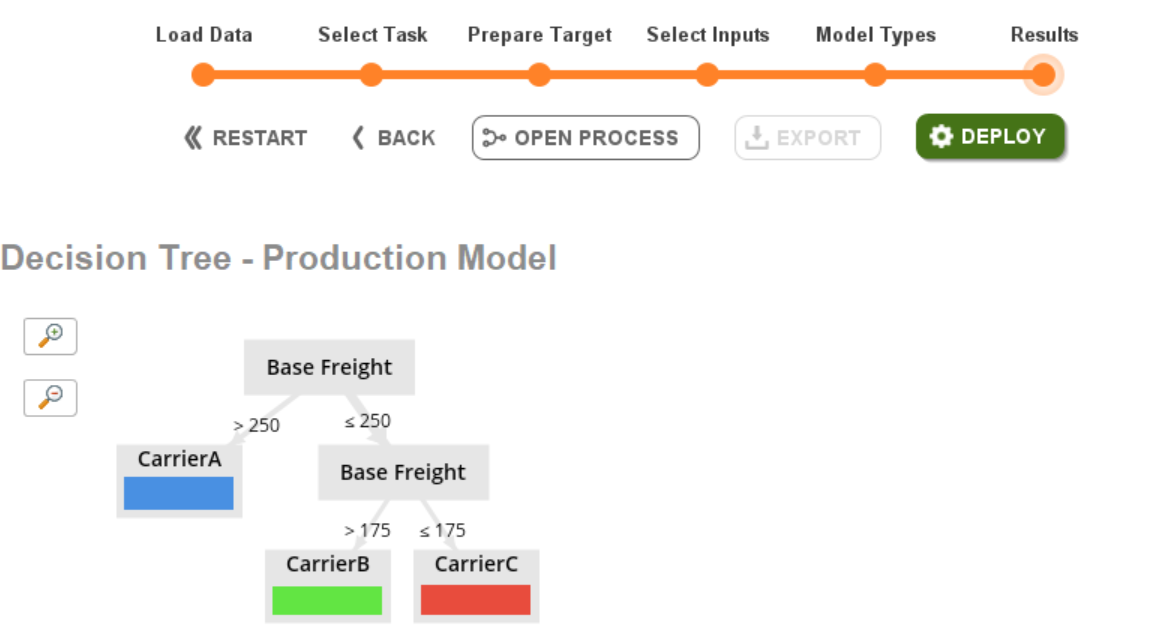
Output of Quality Measures:



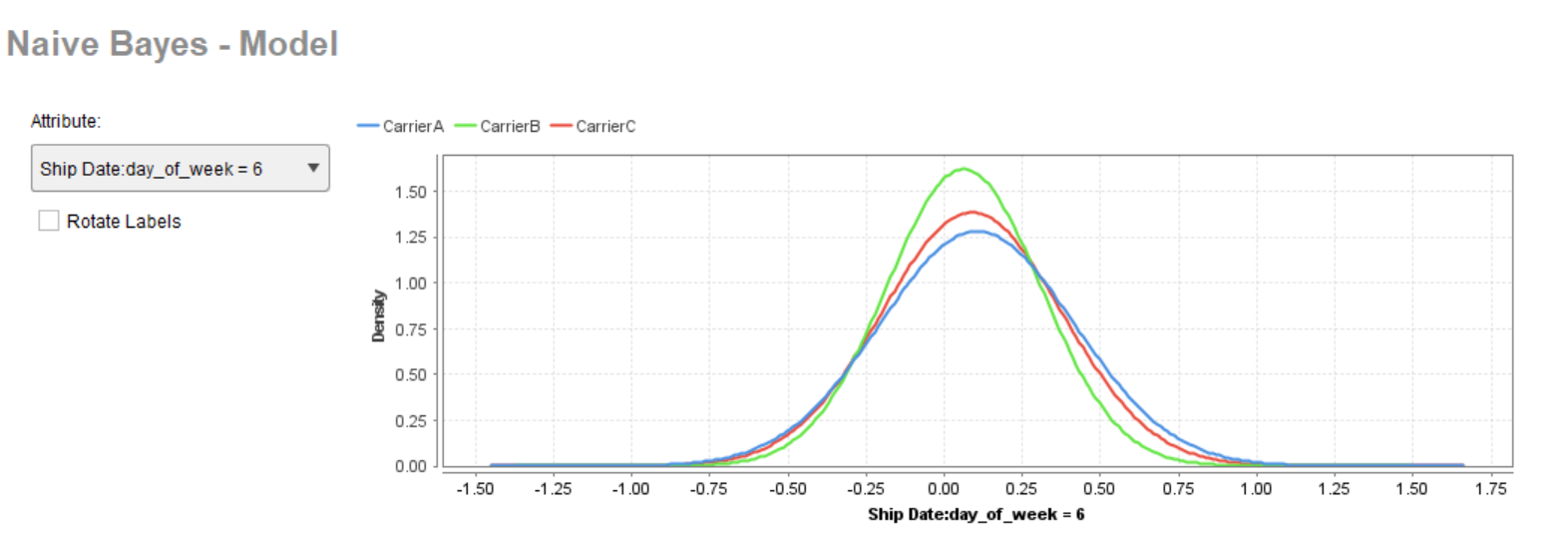




Decision Tree with Base Freight – scored 100% accuracy with only base freight.



Excluding low correlated attributes and base freight, total freight cost, and premium. Used Naïve Bayes Model to page through the different features to see the different between the 3 carriers.

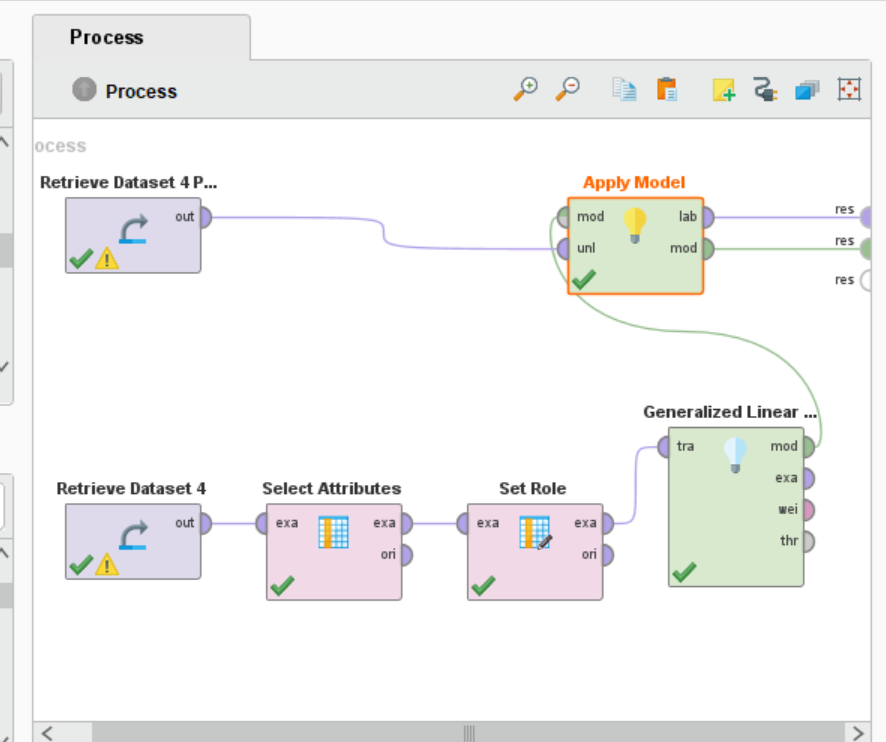


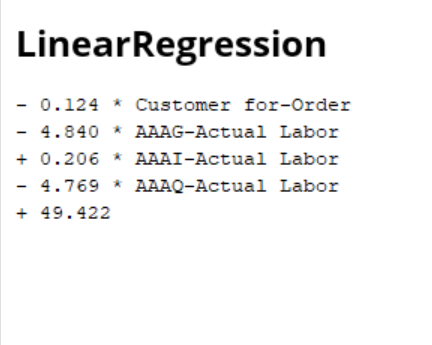
2. Import Dataset 4 and perform regression analysis using the model of your choice for the dependent variable “Throughput”. Using the Unlabeled Production dataset demonstrate how this model could be deployed to production to enable sales order entry staff the capability of quoting more accurate lead times to customers. Communicate your ideas on the BB discussion topic for the week.

Results: In the below screenshots, I created a linear regression model using the training data. Once the model was created, I applied the model to the production data for a label to be assigned.

Once the model is built, we can see the details of the model. Linear Regression is just an equation so we can find the equation built using the most significant variables to simplify the model. This equation can be given to a developer to incorporate into software.

An additional thought I have had is for a more complex model, the lines of code it would take to build the model is significantly less in python. Code can be reused for cleaning/preprocessing from using functions and object-oriented programming. Instead of rebuilding the model when a decision needs to be made on new data, the software could store the model and use it when necessary using the Python code instead of trying to recreate the steps the model would take in whatever middleware code the software is using, I would think there is a way to import a python class to use similar to what I know is possible using Microsoft .Net Framework and .dll libraries.





3. Watch the Rapidminer video regarding Naïve Bayes using the Golf example. <https://www.bing.com/videos/search?q=naive+bayes+rapidminer+video&view=detail&mid=451C577DF1B7C4096589451C577DF1B7C4096589&FORM=VIRE>

Results: Naïve Bayes takes each condition and counts the number of true/false cases. As a result, the probability can be computed of a true/false case when a condition is met.

An example in the video, which applies the method to playing golf or not, examines the outlook of a day which could be sunny, overcast, or rainy. If 2 of the 9 observations include a sunny day when golf was played, the probability of that occurring is 2/9.

The same concept is applied to other conditions and the overall probability of a true case and negative case is calculated.

Multiplying these probabilities together result in the likelihood of the event occurring. The higher the likelihood between positive/negative cases indicate a positive or negative prediction. To convert likelihood to a probability, divide the likelihood by the sum of the likelihood scores. This is the base rule.

Naïve Bayes because the formula assumes the assumption of independent variables which isn't usually the case.

4. Watch the following two videos from 3BlueBrown1 on the topic of Artificial Neural Networks and Deep Learning:

<https://www.youtube.com/watch?v=aircAruvnKk>

<https://www.youtube.com/watch?v=Ilg3gGewQ5U&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi&index=4&t=0s>

Results:

The following are the main concepts I gathered from the above videos:

Convolutional Neural Network = Good for image recognition

Long Short-Term Memory Network = Good for speech recognition

Neural Networks are inspired by the brain.

Neuron = holds a number (between 0-1)

First Layer - Creates a neuron for each pixel. (EX: If 28 X 28 pixels = 784), stores gray scale value which represents "Activation".

Last Layer - Contains the value that represents how much the system thinks the observation belongs to a class/outcome.

Layers in between are "Hidden Layers". Experiment on # of layers and # of neurons for the layers. These hidden layers may be trained to identify loops a number has. Loops can also be divided into a sub problem identifying smaller components of the the loop. These layers can help identify other components of numbers (straight line, etc).

The same concept can be applied on images and audio.

In a layer, weights are assigned to attempt to identify a pattern (edges or loops). A bias can be included to activate when the weighted sum is greater than a number. Weights around the pattern in an area of the image can be negative. The sigmoid function is applied to ensure the neuron's value remains between 0 and 1.

Learning means the computer is trying to find valid settings for weights and biases.

The cost function of neural networks is taking the output of the network, sum the difference the output you wanted it to give, and squaring the results for each observation, average it for the cost of the model.

To train, identify the classification desired. Try to increase the activation for that class and decrease the activation for other classes by 1.) increasing the biases, 2.) increase the weight (in proportion to the activation function), 3.) change the activation function for the previous layer (in proportion to weight).

Back propagation, taking note of adjustments needed for the last layer to the previous later. Identifying the nudges needed for the hidden layers, backwards, to adjust the weights/biases of the layers. The average of these changes needed for each training data is the negative gradient.

5. Go to Provalis Research Site and download QDA Miner and Wordstat. Test the operation to ensure that the applications are installed properly.

Results: QDA Miner and Wordstat are successfully installed on my machine.