David Boateng

12/17/2015

**REGIONAL COST RATIO RECOMMENDER ENGINE SOFTWARE APPLICATION**

**1.0 DEFINE PROBLEM**

This report will serve as my term software project proposal for predicting regional cost for vehicle parts. The rational for this project is based on a business objective from my work, which is to minimize total enterprise cost across the global vehicle development process. The original goal of this project is to evaluate whether regional cost indexes for globally sold commodities can be used to approximate regional prices for vehicle parts sourced in different regions. The new goal of this project is to evaluate whether historical regional part attributes and cost indexes can be used to approximate regional cost for vehicle parts sourced in different countries using user based collaborative filtering methodology.

1.1 Background

We have developed several tools that analyze cost factors that affect total enterprise cost. An additional capability that is needed is approximating sourcing cost for a part across various regions. The traditional business process is to undergo a labor and time intensive cost engineering study for each sourced part. One can see that this becomes an expensive but necessary exercise if you consider the 100s of sourced parts. If we can approximate the supplier sourcing cost in different regions we could use these predictions as a directional indicator for supplier sourcing opportunities. Cost engineers will be able to focus on a smaller set of parts to study thereby increasing the chances of reducing total enterprise cost. Please note that there are proprietary issues surrounding this project therefore any confidential information will be removed from all datasets and the lessons learned from this project will be used at my work and should remain confidential.

1.2 Project Tasks

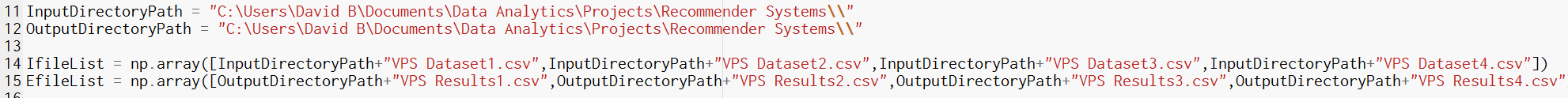
The first task will be to attempt to collect pricing data for various commodities across different regions. I was unable to find consistent pricing for automobiles commodities across different regions. I studied several datasets like Bureau of Labor Statistics Consumer Price Indexing, Burger King Index/Purchasing Price Parity, and the World Bank International Price Indexes for attributes related to energy, raw material, and new vehicle sales. I decided to course correct and use historical supplier cost data for vehicle parts as my primary data source because this data source provides some cost across various countries. I will use the regional part cost data as an input to provide recommendations for approximating the supplier cost in different regions. I will use the user based collaborative filtering technique to approximate pricing in unknown regions. In this case the user will be vehicle parts and the items will be country pricing.

I will test the algorithm with four datasets. The first dataset will be identical to the example used by Dr. Fox’s user based collaborative filtering to prove that the code and algorithm is programmatically correct. The second dataset will be more complex with more countries and missing regions to prove that code and algorithm can properly scale to handle larger datasets. The third dataset will be as complex as the second dataset in size with some random pricing added to prove that the algorithm can adjust to provide different recommendations when the dataset is not as sparse. The fourth dataset will be large and complex to prove that the code and algorithm can properly process large datasets. Lastly, I will provide the knowledge and wisdom learned from each dataset.

**2.0 DEFINE SOFTWARE**

My software project will be developed in python code. It will use the following standard libraries numpy, copy, and math. It will also use scipy.cluster.vq and scipy.stats for kmeans clustering and pearson correlation functions. The software is a regional cost recommender engine. The recommender engine uses the inputted csv files that contain cost for various parts across different countries to predict/recommend cost for countries that don’t have cost.

The software starts by taking an array of input and output directory destinations.



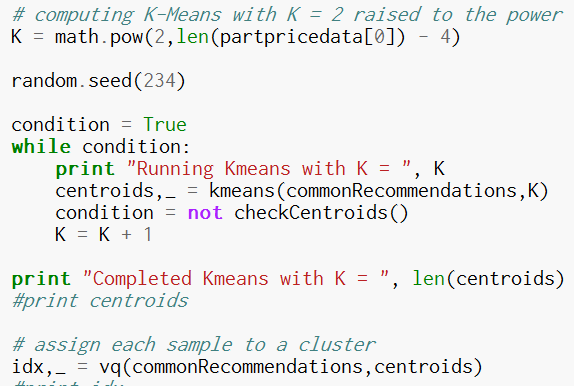
The software loads each csv file into a 2-dimensional array and prepares the matrix for clustering.

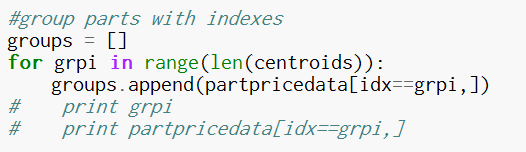


The preparation involves transforming the matrix into sections where regional pricing is known and now known, as well as part attributes that describe the part. The resulting matrix will consist of 0s – indicating part attributes, 1s indicating known pricing, and -1s indicating missing/unknown pricing.

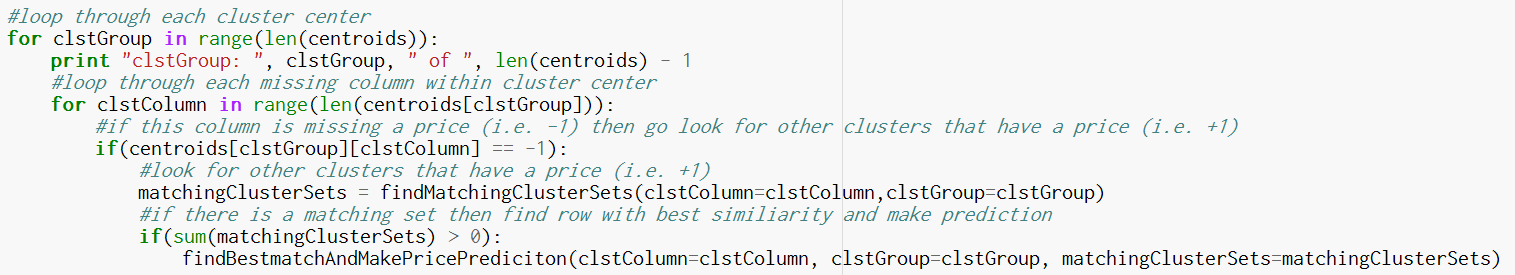


The software takes the transformed matrix and runs a Kmeans clustering algorithm to split the matrix into groups of known/unknown pricing groups. The software will only allow centroids that have elements that consist of 0,-1, and 1. The software will run K-means with different centroid targets until the proper conditions are met. The clustering will be based on regional pricing. The software needs distinct grouping of clusters in order to properly align missing pricing data.



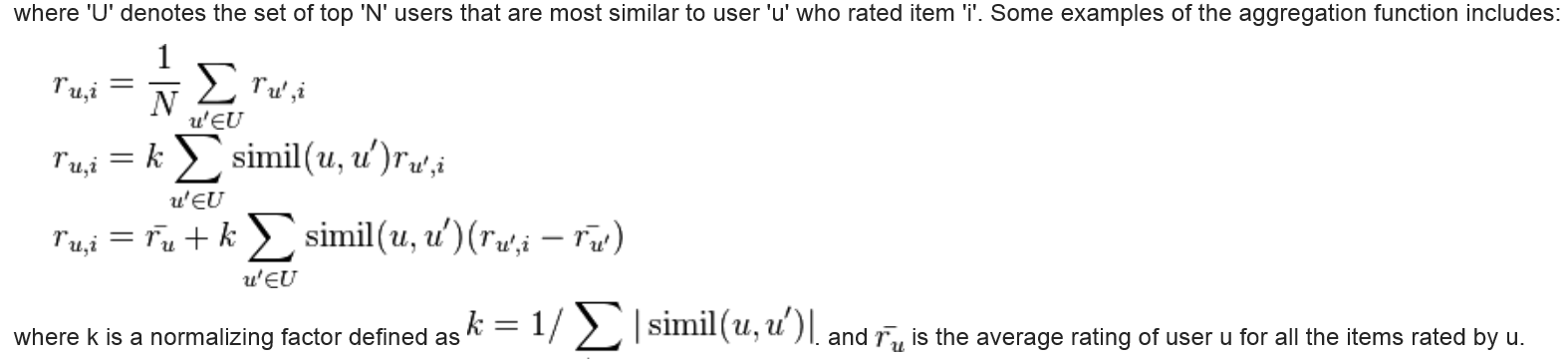
The software then groups the parts into groups of that common missing pricing information according to the centroids. 

The software then loops through each group of parts that have common missing pricing information and determines which other group of parts can provide a recommendation to the missing pricing information.



The recommender engine uses a user based collaboritve filtering technique to provide regional pricing for missing/unknown regions. The core functions for the recommender engine are findMatchingClusterSets and findBestmatchAndMakePricePrediction. The findMatchingClusterSets function determines which group of parts can be paired to provide recommendations for missing regions. For example suppose we have two groups of parts that can be describe as centroids [0,1,1,-1,1] and [0,-1,1,1,1]. The algorithm would pair these two part group centroids together because they both have known pricing for index 2 and 4.

The second function findBestmatchAndMakePricePrediction uses the pairing from findMatchingClusterSets to identify the parts that have the best similarity that can be used to make pricing recommendation/prediction for missing/unknown regional pricing. The similarity will be measured using the Pearson correlation coefficient and based on regional pricing and part characteristics like part cubic feet, weight, and part classification. The recommendation formula will be based on the prediction formula provided by Jannach, Zanker, Friedrich.

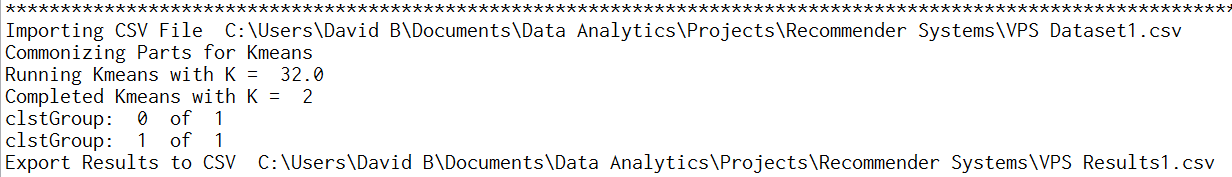


Finally the results of predictions made by the recommender engine are exported to a CSV file.



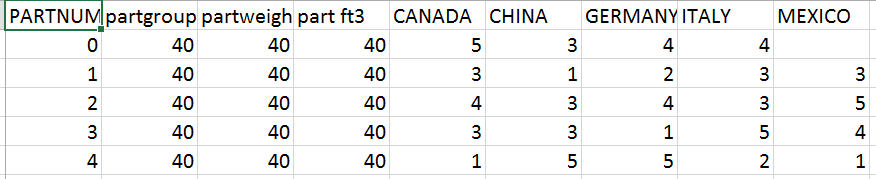
**3.0 PRODUCE RESULTS**

The software will provide status updates on the progress of the recommender engine. The recommender engine quickly processes the first three csv files, and takes much longer to process the larger fourth csv file. The message below are the status updates while running the first csv file.

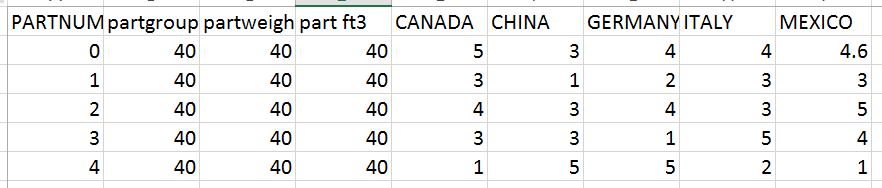


The software first indicates that it is importing a csv file labeled VPS Dataset1.csv. Next the software commonizes the part for kmeans by transforming the part matrix into 0,1, and -1. Next the kmeans clustering routine is initiated with varying K centroids until it arrives at a centroid selection that only consists of 0, 1, and -1. Next the recommender system runs through each centroid group 1 of 8, 2 of 8,… n of 8 and recommends/predicts pricing for missing/unknown regional areas. Lastly the results are exported to a CSV file labeled VPS Results1.csv.

The results from VPS Dataset1.csv are provided in the VPS Results1.csv are below.

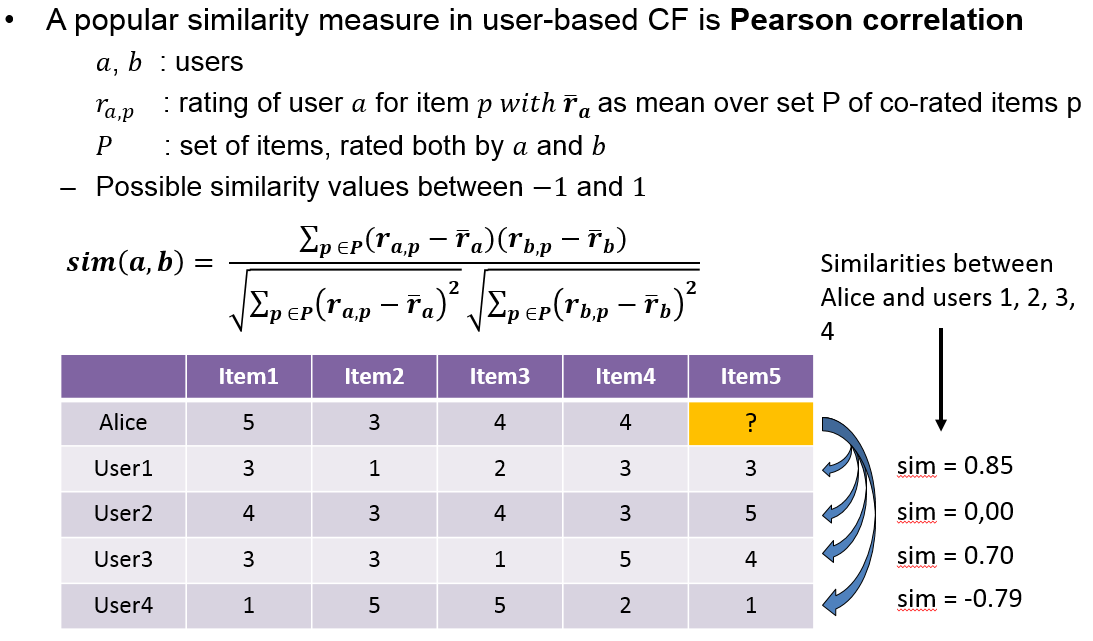
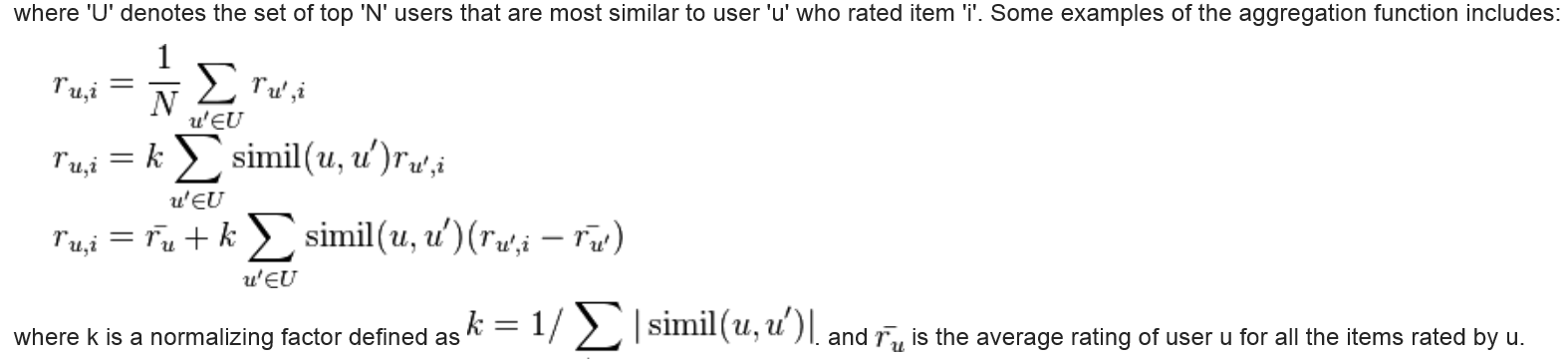


VPS Dataset1.csv



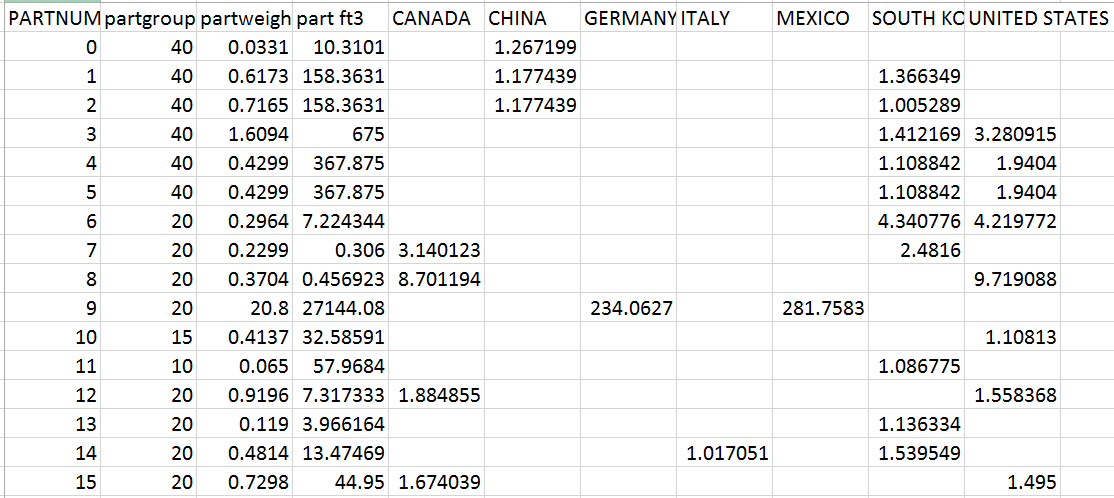
VPS Results1

In this test all parts have a price except for one part in Mexico. The recommender engine should only make one prediction for PartNum = 0 in Mexico based on the most similar PartNum = 1 which should result in a predicted price of 4.6. The prediction and similarity calculations are as follows:

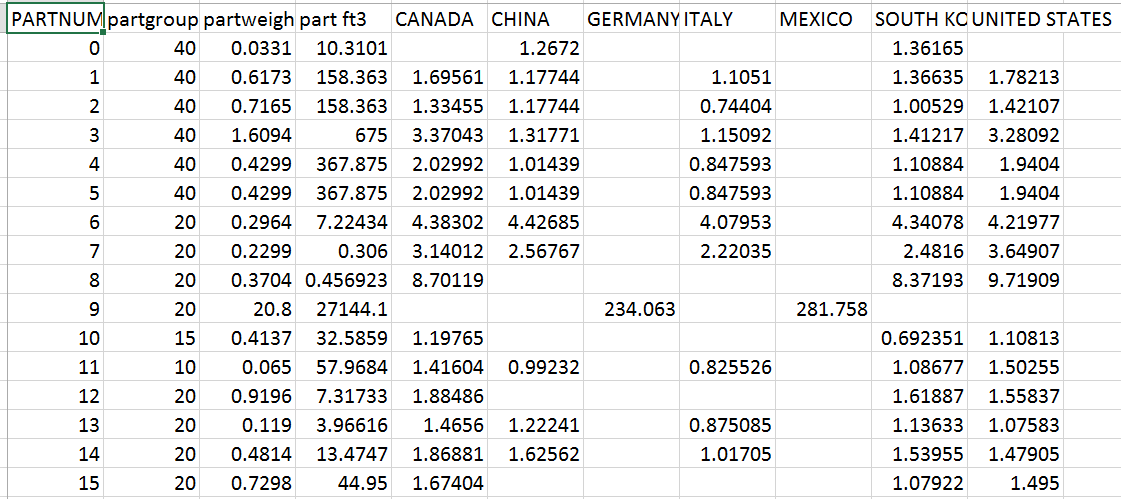
 

pred(PartNum= 0, Mexico) = 4 + 0.6 = 4.6 (*similarity is based on the single most similar part instead of the top N most similar parts)*

The results from VPS Dataset2.csv are provided in the VPS Results2.csv are below.



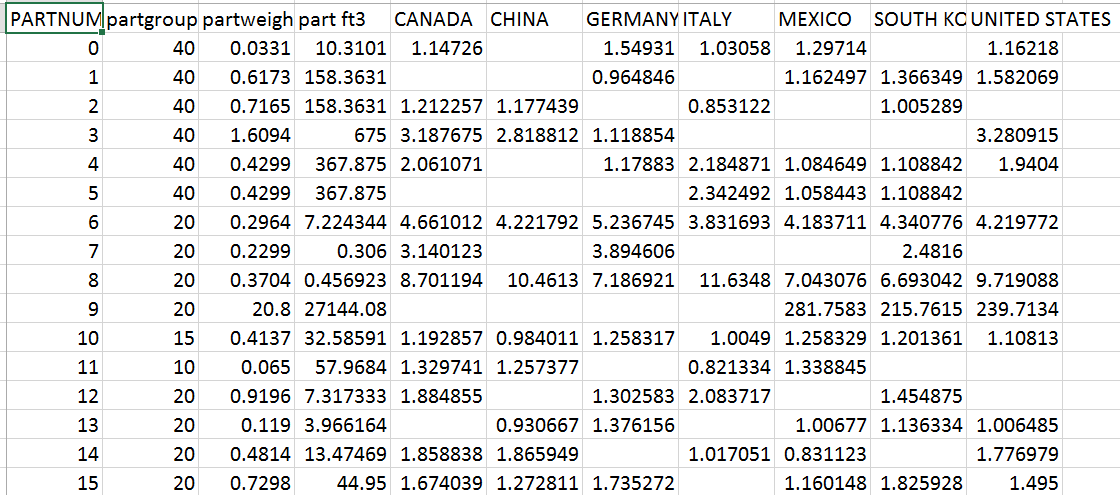
VPS Dataset2.csv



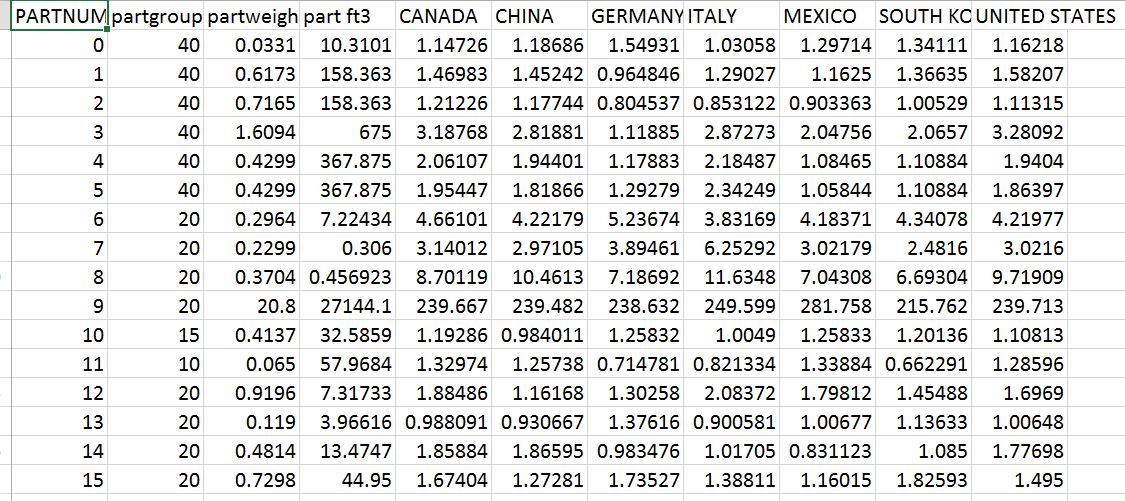
VPS Results2.csv

In this test all of the parts have missing pricing in multiple regions. The recommender engine was able to provide prices for most regions except Germany and Mexico because no other part had similar pricing and part attributes in those regions. The recommended prices for all of the parts were reasonable compared to the originally known regional prices.

The results from VPS Dataset3.csv are provided in the VPS Results3.csv are below.



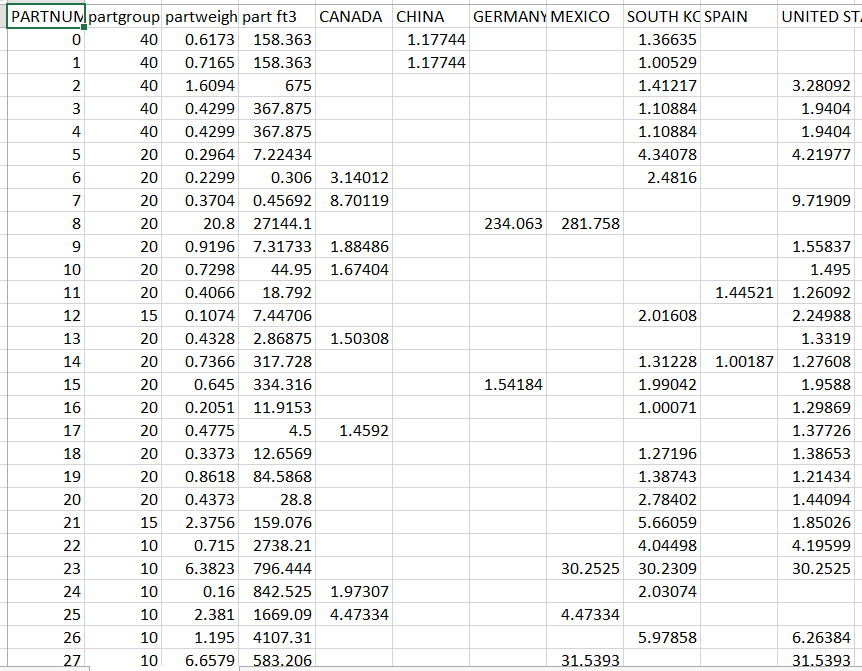
VPS Dataset3.csv



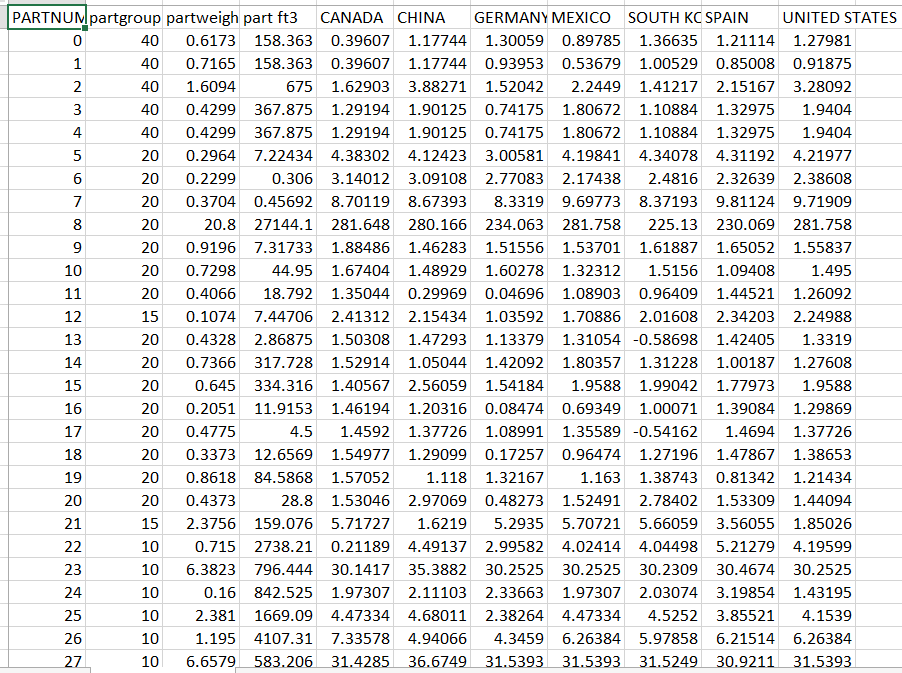
VPS Results3.csv

In this test all of the parts have missing pricing in multiple regions but the input is not as sparse as VPS Dataset2. The recommender engine was able to provide prices for all regions. This test proves that with less a sparse input file the recommender engine can provide more price recommendations for unknown regions.

The results from VPS Dataset4.csv are provided in the VPS Results4.csv are below.



VPS Dataset4.csv *(Screenshot of 27 of the total 498 parts)*



VPS Results4.csv *(Screenshot of 27 of the total 498 parts)*

In this test there are 498 parts with missing pricing across multiple regions. The recommender engine was able to provide prices for all regions. This test proves that the recommender engine can scale up to larger datasets and still provide pricing recommendations.

My recommender engine python software has room for many improvements. First, the recommender engines needs a validation routine to improve its ability to handle loosely constructed datasets i.e. the recommender engine doesn’t work if there are rows in the input file that don’t have at least 2 regional prices because the Pearson correlation routine needs at least 2 prices. Second, if I had more time I would improve the findMatchingClusterSets routine to not be dependent on only handling 0, 1, and -1 in order to find matching centroid sets. If I made this improvement I would not need Kmeans to keep running increasing K centroids until it finds centroids with only 0,1, and -1 (as is the case in VPS Dataset4.csv). Third, I need to find a solution to deal with the data sparsity problems. One thought I had was to seed the initial phase of the recommender engine with average prices from groups of common parts using a Gaussian mix cluster routine to group parts based on part attributes. Next, I would adjust the prediction function to be based on a set of the most similar parts instead of the most single similar part. Next I would add a recommendation score metric that provides some measure of how many predictions the recommender engine was able to produce. Lastly, I would improve my code to run faster and more efficiently to take advantage of python’s quick vector math and use a map/reduce algorithm to process larger datasets like VPS Dataset4.csv.

3.1 Wisdom

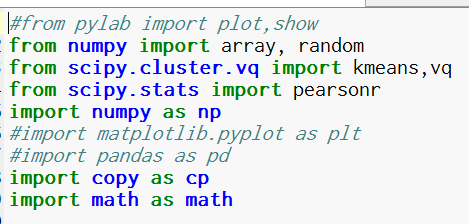
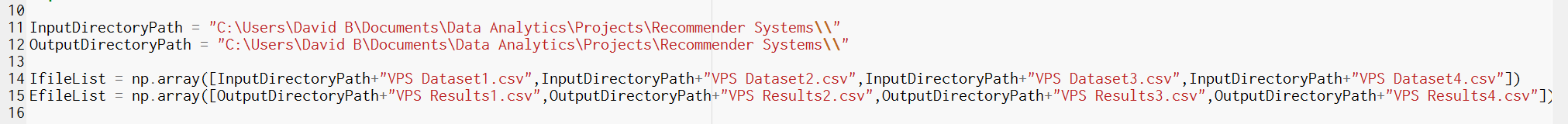
I was able to produce a regional cost recommendation engine for parts based on historical part cost data. It appears that the recommendation engine works really well when it uses datasets that are densely populated and has 100s of part rows. I believe this algorithm is a good start to a productionized recommender engine that takes into account the improvements mentioned above. I would also need to work with a cost engineer to see if the recommended pricing is reasonable.

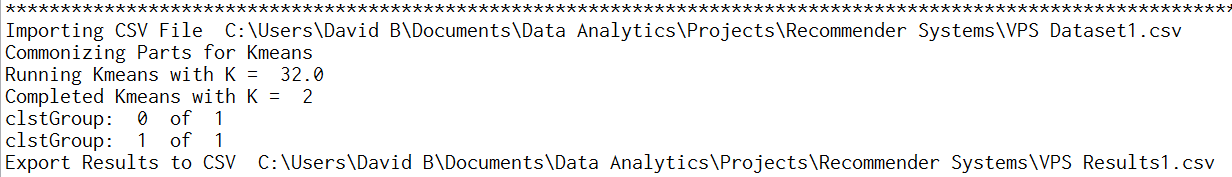
**4.0 REPRODUCIBILITY**

The python code was designed to work on a Windows 8 environment using the Canopy IDE. The project submission will come with the following files:

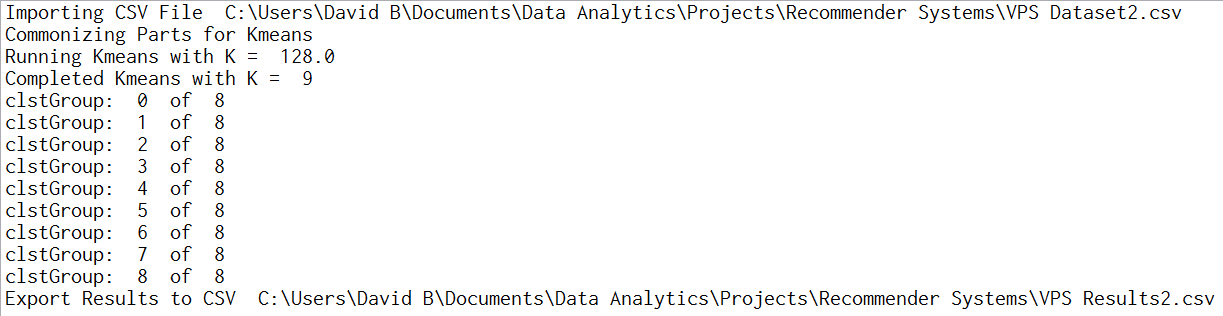
* python code labeled “VehiclePartData Recommender Algorithm v3.py”
* four csv input files labeled “VPS Dataset1.csv”, “VPS Dataset2.csv”, “VPS Dataset3.csv”, and “VPS Dataset4.csv”
* four csv output files labeled “VPS Results1.csv”, “VPS Results2.csv”, “VPS Results3.csv”, and “VPS Results4.csv”.

In order to run the python code you will need to do the following:

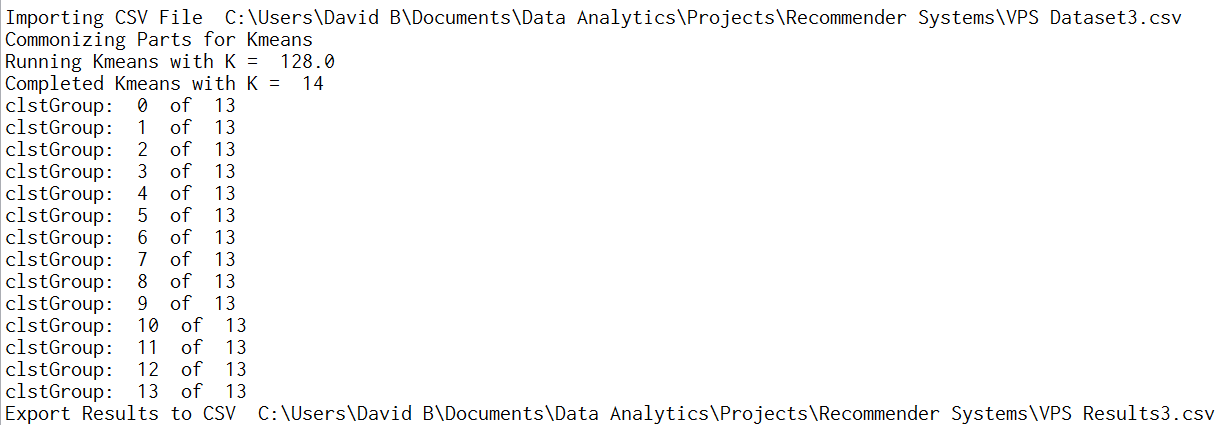
1. Include the numpy, scipy.cluster.vq, scipy.stats, copy, and math libraries to Canopy Package library. The python code requires that your Canopy IDE have the following libraries installed: numpy, scipy.cluster.vq, scipy.stats, copy, and math.
   1. 
2. Save the python code and four csv input files labeled “VPS Dataset1.csv”, “VPS Dataset2.csv”, “VPS Dataset3.csv”, and “VPS Dataset4.csv” to a Windows directory.
3. Open up the python code and update two variables called:
   1. InputDirectoryPath (line 11) to be destination directory for where you saved the four input csv files (“VPS Dataset1.csv”, “VPS Dataset2.csv”, “VPS Dataset3.csv”, and “VPS Dataset4.csv”). Please make sure you end the string with the “\” character.
   2. OutputDirectoryPath (line 12) to be destination directory for where the output csv files generated by the python code will be saved. Please make sure you end the string with the “\” character.
   3. 
4. Click Run.
5. Monitor the python terminal for the following status updates:
   1. The status updates for the first input csv file VPS Dataset1.csv



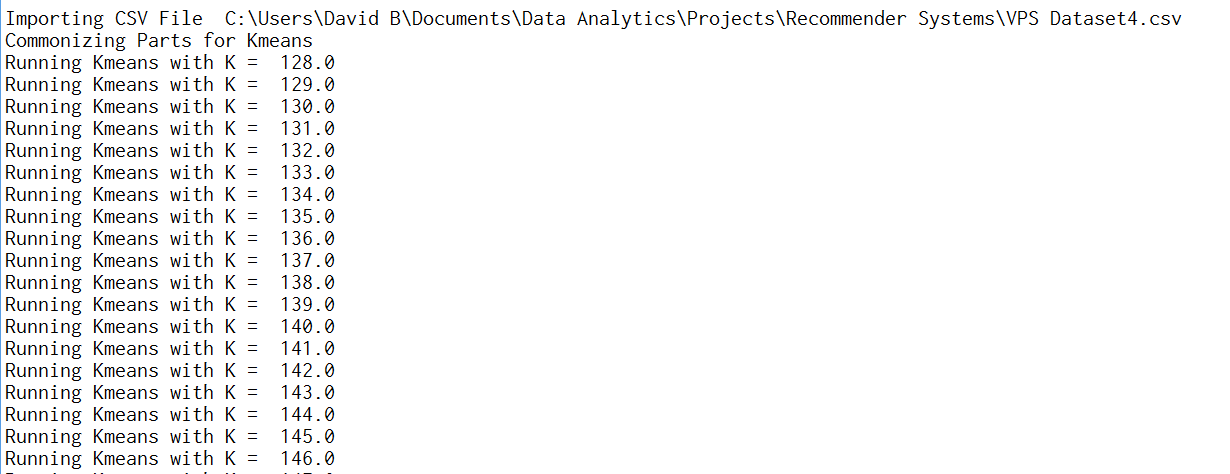
* 1. The status updates for the first input csv file VPS Dataset2.csv



* 1. The status updates for the first input csv file VPS Dataset3.csv



* 1. The status updates for the fourth input csv file VPS Dataset4.csv



…

…



*Snippets of results*

* 1. If for some reason you are not able to successfully run the python code as specified. Please refer to the four csv output files labeled “VPS Results1.csv”, “VPS Results2.csv”, “VPS Results3.csv”, and “VPS Results4.csv” that were generated from my computer.