André Araújo
E-mail: araujo@pythian.com

2014 CCP: Data Scientist
Medicare Claim Anomaly Detection Challenge

June/2014

This document provide details about my submission to the **2014 CCP: Data Scientist Medicare Claim Anomaly Detection Challenge**, explaining the logic behind my answers and solutions.

Data preparation

Time spent: 4 days to create the Crunch pipeline with the needed input formats

The input data for the challenge was provided in formats (CSV, XML and ADT files) that could not be easily read by mapreduce jobs. I used Apache Crunch to implement a job to read the data in all the input files and save them as Avro files, which would make it easy to process it later in Hive or mapreduce jobs. This job is implemented by the `InputProcessJob` class.

The provided XML, CSV and ADT files were processed by the Crunch job using the following input formats to read the data:

XML files – I used Mahout's `XmlInputFormat` class in the Crunch pipeline to read the patient records from this file.

CSV files – I also found an existing implementation of an input format for CSV files in https://github.com/mvallebr/CSVInputFormat. The implementation of the `CSVNLineInputFormat` and related classes available on GitHub at the time, though, had bugs, so I had to fix those bugs in my project to be able to read the CSV files correctly. The changes are available in the submitted project but haven't been pushed to original repository yet.

ADT files – I couldn't find any available implementation of an input format for ascii-delimited files, so I implemented my own `AsciiDelimitedFileInputFormat` and `AsciiDelimitedLineRecordReader` classes for that.

The data from the files above were rewritten to Avro files, using the Avro schemas defined in the files under the `src/main/avro` directory in this project.

The only input file that didn't require "special" processing was the `REVIEW.TXT` file, containing the claims identified as suspicious. I just loaded it directly into a Hive `TEXTFILE` table.

Part 1

Time spent: 2 days

The questions from this part were objective and straight-forward. Being the first part of the challenge to tackle, I had to spend a bit of time familiarizing with the data set, browsing the CSV files and/or looking at the data already loaded into Hive to understand the significance of the information provided. A bit of research on the Medicare site, Google and Wikipedia also helped to understand the DRG code system and a few other concepts regarding this challenge.

Since I had the data already in Hive tables, I wrote the following queries to answer the 4 questions in the part:

`scripts/part1a.hql` – calculates the relative variance ([stddev/mean]^2) for each procedure (including outpatient and inpatient procedures), sort them by this value and select only the top 3 ones.

`scripts/part2a.hql` – for each procedure calculates the provider that claimed the highest amount and then counts how many times each provider was the highest claimer. Get the top 3 ones.

`scripts/part3a.hql` – first calculates, for each procedure, the average claimed amount per region, by summing the total amount claimed for each provider in that region (average * number of claims) and dividing by the total number of claims in the region. Then it counts how many times each region was the highest claimer for each procedure and gets the top 3.

`scripts/part4a.hql` – for each procedure calculates the provider that had the highest value for the difference between claimed and paid amounts and then counts how many times each provider was the highest claimer. Get the top 3 ones.

All **validation and visualization** for the results of the scripts above was done using R to review and plot the data. The R script used for the review and plotting the data can be found in the script `scripts/part1.r`.

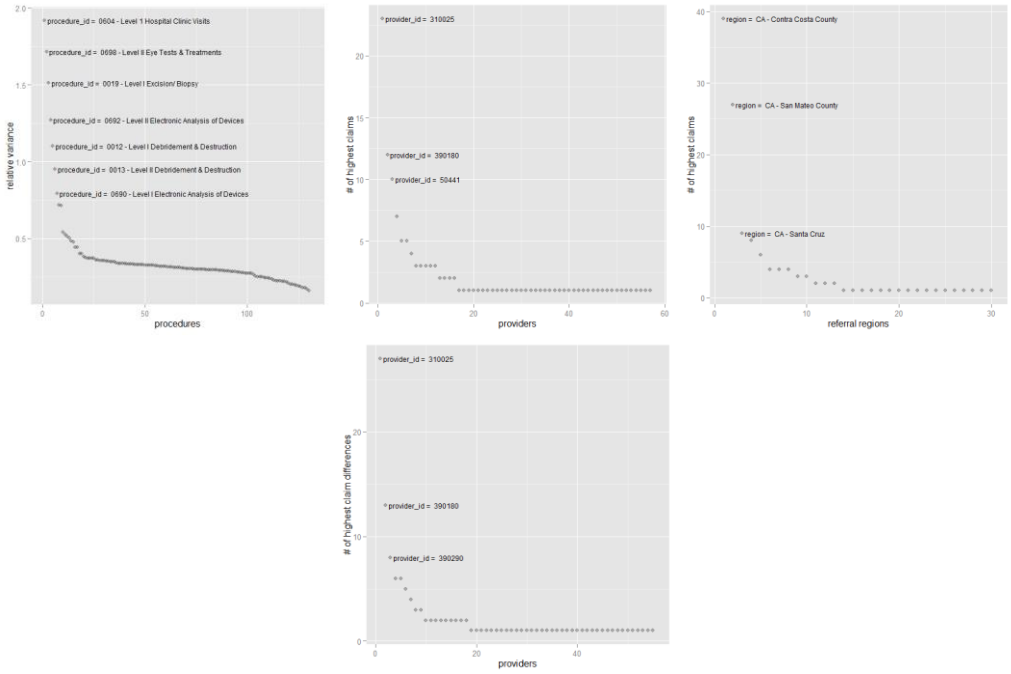

Figure 1 - Plots for questions in part 1

Part 2

Time spent: around 2 days to compare different hypothesis and create the final R and Hive scripts.

A different perspective to take at the claim amounts is in regards to their variability. Variation is expected but excessive variation for the claims of the same type of procedure within the same provider or region could indicate problems and should be reviewed.

Since the price ranges for different kind of procedures are different, making it difficult to do a comparison across all the claims for a given provider or region, I normalized the average charges and used the **standard score** for them to find out which providers and region had the highest variability for their claims. I also noticed that in some regions and providers there was a large different of variation between inpatient and outpatient claims, so I considered claims of different type separately for the ranking.

The variability is calculated as the variance of the standard scores for each procedure, within each provider or region.

For this question, I used R to do some exploratory analysis and compare different hypothesis. The final result file was generated with the Hive queries in file `scripts/part2*`.

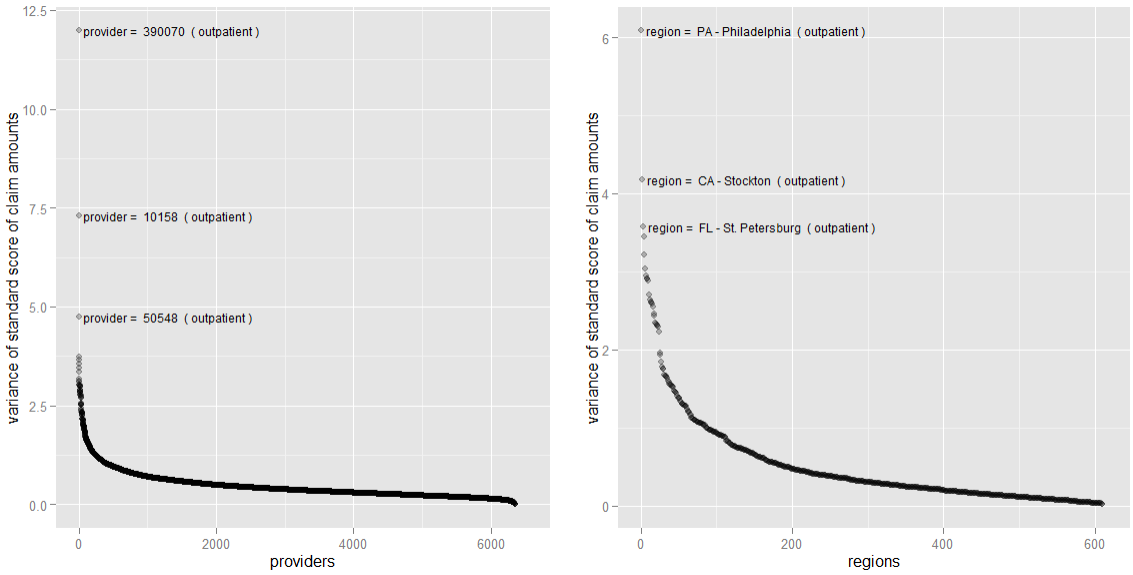

Figure 2 - Plots showing the providers and regions with the highest claim variation

Part 3

Time spent: about 1 week, extracting features, tuning the learning algorithms and trying out different alternatives.

The solution chosen for this problem (finding other 10000 reports that should be reviewed) was to use supervised learning to classify the existing claims as either “ok” or “for review”. The first step was to choose and extract the features that the classifier was to use.

Feature extraction

The data for this problem was provided in the form of patient records, containing demographics data about each patient, and the actual history of claims (date, patient id and the procedure id). The set of features I decided to work with was all the available demographics information and the frequency of the claims per use (both per procedure and per type of claim – inpatient vs. outpatient):

Using Hive to join the patient records with the claim history, I extracted the following features for each patient, storing them in an RCFile table:

- Age (categorical with 5 levels)
- Gender (categorical with 2 levels)
- Income (categorical with 6 levels)
- Number of inpatient procedure claims (numeric)
- Number of outpatient procedure claims (numeric)
- Number of claims per each procedure (numeric, 1 feature per procedure, to a total of 130 features)
- Bias (constant)

Choosing the training set

The data provided with the challenge contained all the data points marked "for review", the positive labels. However, I still needed points that were known, or considered, to be legitimate, as opposed to "for review", data points (negative labels), to be able to train the classifier.

Since we had 50045 data points marked for review, my first thought was to randomly select the same amount of points randomly from the full data set and label them as "legitimate". Since I expect that there are a lot more legitimate reports than suspicious ones, this approach should work with a reasonable accuracy.

However, I thought of the alternative of using unsupervised learning (clustering) to try to separate legitimate claims from suspicious ones. I wasn't sure if the results of the clustering would help much but gave it a got nevertheless. For the clustering, I chose 200,045 data points, including the 50,045 ones marked for review and 150,000 data points chosen randomly from the remaining of the data set.

I implemented the clustering job using the **k-Means algorithm from Mahout 0.9** (`KMeansClaims` class in this project). The clustering converged very quickly for this data set and it turned out that the data clustered in a very specific and interesting way:

	Cluster A	Cluster B
For review	48,992	1,053
Unlabelled	20,766	129,234

Table 1- Clustering results

The vast majority of the suspicious claims (97.9%) ended up in Cluster A. This indicates that if any other data point was a suspicious claim it would also be more likely to be found in Cluster A. The distribution of the unlabelled data points was not as dramatic, but there was visibly a skew towards Cluster B, with 86.2% of the points in that cluster.

With that result, I concluded that if I were to choose one random sample of points from the full data set and another random sample from the data points in Cluster B, it would be much less likely to find suspicious claims in the sample taken from Cluster B. This would help the classifier, since the set of negative labels would have less noise.

The final training set I chose for the supervised learning phase was the union of the 50,045 claims marked for review and 50,000 claims chosen randomly from the ones in Cluster B above.

Classification

For the supervised learning, I used **Online Logistic Regression (OLR) from Mahout 0.9**, implemented in this project in the `ClaimClassifier` class. Even though it's a sequential algorithm it was ok for the size of data set I was using. The training on the 100,045-sample set takes under 1 minute to run and the classification of the remaining ~100 million claims took around 20 minutes to run on a 4-core Intel Xeon 2.6GHz.

The out-of-sample accuracy from OLR was of 98.7%. The classifier implementation prints out, at the end of the training phase, the *beta* coefficients used by the OLR algorithm. This tells us which features had the most weight in the classifier decisions and results.

It indicates that the most determinant features for the classifier were the overall number of **outpatient claims** per patient (negatively correlated) and the overall number of **inpatient claims** per patient (positively correlated). It means that the **more inpatient claims** and the **less outpatient claims** a patient has, the more likely it is to be chosen for review. A few individual procedures also appear to be slightly correlated to the outcome. They are shown in the plot and table below:

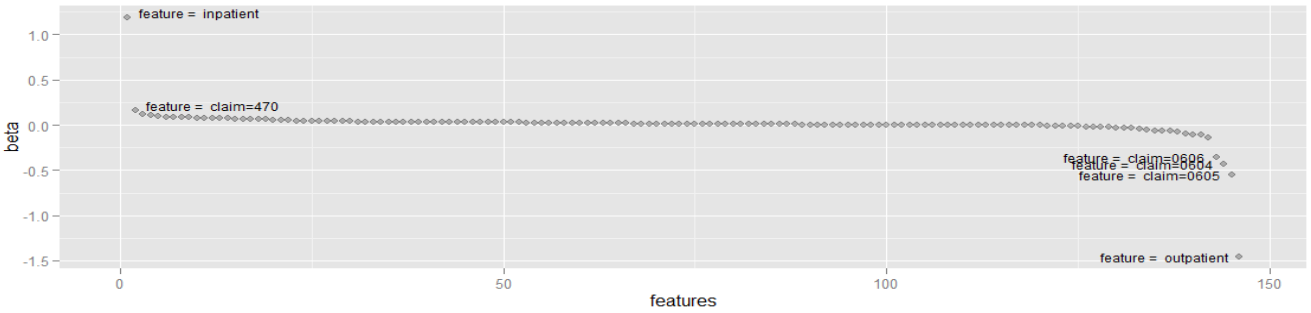

Figure 3 - Beta coefficient for each feature for the OLR classifier

Feature	Beta	Correlation
outpatient	-1.453	Negative
inpatient	1.189	Positive
claim=0605	-0.550	Negative
claim=0604	-0.434	Negative
claim=0606	-0.352	Negative
claim=470	0.165	Positive
claim=0607	-0.139	Negative
claim=871	0.114	Positive
claim=460	0.112	Positive
claim=0690	-0.110	Negative

Figure 4 - Top 10 determinant features

All the scripts used for the part 3 solution, including the R script to plot the graph above and list the top 10 features, can be found under `scripts/part3*`.