AMOD 5450H: Data Mining

December 16, 2017

Data set: National Collision Database (NCDB)

Data repository: Federal Government of Canada – Open Canada

Division: Transport Canada

Released on: April 13, 2012

Submitted by: Nicholas Hopewell

Submitted to: Dr. Sabine McConnell

I would like to preface this report by mentioning that everything described within the body of this paper is described in much greater detail, and with more careful consideration, in the accompanying Jupyter Notebook.

My approach to this assignment was to put myself in an environment where I knew I would get the most holistic learning experience possible. First and foremost, I needed a data set that was unseen outside of the repository it came from. The dataset had to be one which had never been publicly worked with in the past and had absolutely no previous examples available for reference from any other analysts, scientists, or students. In addition, the data set had to pose many challenges before model induction and analysis could even begin. More specifically, it needed to have mixed data-types (especially categorical data), its dimensionality had to be reasonably high either across columns, attribute levels or both, it had to have missing data, and I wanted it to be large enough in size so that I could continue to apply techniques to it in the future for my own learning purposes (outside of class). Lastly, the data set had to be appropriate from data mining purposes in general with an attribute containing class labels which could be predicted.

While looking for interesting data, which took significant time and consideration, I added one more very important criteria to my search: the dataset had to have a significant class imbalance. If I could find a data set with one or more rare classes, I knew that the learning opportunity presented would be that much better. When considering the breakdown of class labels of the different data sets during my search, I came to the conclusion that many real-world data sets present class-imbalance problems which must be learned from. It was this realization in particular that got me very excited to find a data set which contained this inherent challenge.

After modifying my search needs to include a class imbalance problem, I had a breakthrough of interesting ideas which ultimately led me to the selection of the data set used for this final project.

Sourced from the Open Canada initiative, an initiative which has compelled the Government of Canada to give public access to key data, the dataset I chose was the National Collision Database provided by Transport Canada. This database included all police-reported motor vehicle collisions on public roads across Canada between 1999 to 2014. This data set contains the binary category 'collision severity' which has the levels 'fatal' and 'non-fatal'. These levels describe records in the dataset in which the associated collision either resulted in a fatality or an injury (but no fatality). I knew before downloading the data that I had found the class imbalance that I had been looking for. Thankfully so, the extreme majority of collisions, reported and not reported, do not result in fatality. Collision severity was determined as the target, with fatal and non-fatal determined as the class labels.

Other attributes in the dataset include collision level data, vehicle level data, and person level data. Collision level data included attributes such as the year, month, day of week, and hour of day the collision occurred. These also include the number of vehicles involved in the collision, its configuration, the layout of the roadway, weather conditions and road surface conditions as well as some others. Vehicle level data include attributes such as the type of vehicle involved in the collision, the age of the vehicle, and the sequence number of the vehicle in the collision. Personal level data include attributes such as the age of the person involved in the collision, their gender, their position in the car, medical treatment they required, as well as safety device used, among others. One piece of key information I really would have liked to see in their dataset was

specific location of collision. If this data were provided, even more interesting insights could be drawn.

When first downloading the data, I saw it had almost six million rows. The original breakdown of class labels were 5,761,772 records belonging to the non-fatal class and 98,633 records belonging to the fatal class, a collision severity ratio of over 58:1 in favor of the non-fatal class. Although I knew I could work with this data, I decided that for the sake of time and efficiency I would instead read in the most recent 10% of the data set. This subset included collisions from February 2013 to December 2014 and excluded collisions which occurred during every other year. The new breakdown of class labels was 577,198 records belonging to the non-fatal class and 8,753 records belonging to the fatal class. This presented a ratio of collision severity of just under 66:1 in favor of the non-fatal class. About 98% of the total records are associated with non-fatal incidents which provided the class imbalance I was looking for. Datatypes of the attributes are discrete numeric and categorical which is exactly what I was looking for having previously worked with continuous data many times in the past. In addition, some of the categorical data has a large number of levels.

Another challenge the data presented was that it had a lot of data which were missing for different reasons. Some of the data should have been collected but wasn't (for unknown reasons), while other data was missing because Transport Canada was not required to collect said data. Missing data is outlined in more detail within the Jupyter notebook containing the analysis. No justification to replace the missing data was identified; there could have been many underlying causes for the missing data and so it would not be acceptable to assume it was missing completely at random. Because of this, I decided to drop rows with missing data despite records with partial data still providing useful information. In the future, I plan to compare models

induced from the data after missing examples have been imputed in some form. Because it is categorical data, one option is to take the mode of the attribute while another option would be to cluster the data and take the mode of only its nearest neighbors. After dropping records with missing data, I was left with 326,050 records associated with non-fatal collisions and 4346 records associated with fatal collisions. This produced a final class ratio of over 75:1 in favor of the non-fatal class. I went on to describe in detail within the notebook how I believed this ratio was perhaps more representative of current collision statistics as the trend in fatal collisions, within the data and in general, is going down for various reasons.

Initially, to get attributes into proper data types I had to encode some categories with string values into integers. After that, I checked some summary information on the attributes to see if there were any data which were inconsistent with the accompanying data dictionary which described the levels of the attributes. No data fell outside the minimum and maximum range of values, nor took on an impossible value so I concluded no inconsistent data were present. Next, I printed out the mode of the attributes and interpreted why the associated values were likely the most frequently occurring in the data set. I chose not to look at counts and percentiles because I knew I was limited to visualizing mostly count values during visualization due to having mostly categorical data. In the visualization section I looked at some time-series plots because I presumed that month, time of day and hour of day would be related to collisions both fatal and non-fatal. I also looked at bar plots and factor plots showing counts split by two and even three different categories at once. Insights taken from these plots are included in the notebook.

Following this discussion, I went on to consider more preprocessing requirements. Feature subset selection and feature creation were considered, but were concluded to not be necessary for the analysis and would result in a net loss of information. Next, I discussed

dimensionality reduction with Principal Components Analysis (PCA) which I revisited at a later section of the analysis in great detail. Due to the fact that the dataset contained a large number of categorical attributes, I next had to map the levels of these attributes with 1 to N mapping (also referred to as 'hot encoding'). By mapping in this way, one new column was created for each of the attribute levels and levels which had no data associated with them were not mapped. A discussion about the number of records associated with each level and the density and sparsity of these associated records was included. After mapping all of the categorical data, I deleted the original attributes so not to replicate information and make the performance and results of induced models meaningless.

Mapping the data presented a new problem in that now some of the attributes contained only ones and zeros (explained in the Notebook), while others contained non-binary numbers. These variables included age of the individual involved in the collision as well as age of the vehicle. After determining how I wanted to represent these attributes, I binned them into intervals making the data categorical and then proceeded to map these bins with hot encoding. Before doing this, I justified my reasoning for choosing the bins I chose and included a discussion about whether this could be seen as abstracting from the original data and why I thought that was not only necessary but justified. For instance, the vehicle ages ranged from as early as 1911 to 2015 (because next year's models are always released later in the year), so I binned those into categories representing very new, new, not old but not new, old, very old, and antique cars by choosing reasonable ranges of model years.

To suitably work with the unbalanced classes of the dataset, I researched many methods for resampling the data as well as performance metrics which could more accurately assess the performance of a classification model on unbalanced data beyond simple accuracy and error

rates. I discuss these metrics in detail later. My goal was to evaluate each model's performance on both the unbalanced data as well as balanced data in which the majority class was randomly under-sampled. Because of this, I proceeded to randomly under sample the majority class (non-fatal collisions) so that records associated with this class, as well as fatal collisions, were equally represented in the resampled data set. This resampled data set was much smaller with 4346 records associated with each class. This is because the non-fatal collisions were randomly resampled to achieve a 1:1 ratio of classes and there were 4346 fatal records within the unbalanced data set after missing data were dropped. As result I was left with two data sets, one with more data and the presence of an extremely rare class, and another with less data and balanced class representation.

Now that the data was on a standard scale ranging from 0 to 1, I moved on to a detailed exploration of PCA. First, to produce a 2-dimensional visualization, I reduced the data into 2 components and visualized it as a scatter plot with the data projected along the new axes. Next, I wanted to determine the number of components which adequately described the nature of the data while still significantly reducing its dimensionality. First, I chose to visualize the cumulative explained variance for a large range of components. The first 25 components accounted for approximately 60% of the variance in the unreduced data while 90% of the variance was explained by about 90 components. The number of retained components depends on the dimensionality of the data (and thus the total variance with needs to be accounted for) which in my case was relatively high. Reducing the dimensionality of the data while still retaining as much variance as possible is the key. When projecting the data into lower dimensions I had to consider the maximum number of components for the purposes of the analysis which I chose to be 40 components. Due to the dimensionality of the data I determined that retaining 70% of the

total variance of the original data was adequate. To find the number of components which cumulatively explained approximately 70% of the variance in the data, I printed out the cumulative explained variance of each component as a list. I determined that 34 components accounted for 70% cumulative explained variance.

Next, I explored unsupervised learning with class labels stripped from the data set. First, I looked at the results of k-Means clustering with the data. It was important to reduce the dimensionality of highly dimensional data before clustering because k-Means uses Euclidean distance to assign cluster membership and measures of distance become much less meaningful in high-dimensions. A detailed explanation of the iterative expectation-maximization algorithm as well as how the k-Means variant finds k clusters in data is provided within the notebook. Considering k-means tries to minimize the within-cluster sum of squared errors (looking for tighter clusters), while maximizing the between-cluster sum of squares (clusters which are spaced-away from each other), to determine appropriate number of clusters in the data, I plotted the incremental increase in between-groups sums of squares for each increase in k. While looking for an elbow in this the plot (explained in the notebook), it was apparent that there were no appropriate number of clusters k-Means had determined for the data. This was reconfirmed through examining Silhouette coefficients for each value of k ranging from k = 2 to k = 19: no appropriate value of k was apparent. To *very* briefly summarize the interpretation included in the notebook, the type of clusters k-Means tends to find simply did not match the true nature of the clusters in the data. k-Means finds globular clusters which reside in a certain shaped space. Natural clusters in this data are uniquely-shaped with differing densities and dispersions surrounded by considerable noise; k-Means cannot adequately cluster this type of data. There were some clusters which k-Means seemed to group adequately but overall this was not the case.

Due to the nature of the true clusters in the data, I suggested a density-based algorithm such as DBSCAN would be more appropriate (I later applied this to the data for comparison).

Continuing with clustering and possibly extending to an area where k-Means showed to be inadequate, I applied a Gaussian Mixture algorithm which takes into account the uncertainty of cluster membership to clusters which are close-together or overlapping. Gaussian Mixture incorporates certainty of well-separated clusters and uncertainty of not well-separated clusters by assigning probabilities to each data point based on their distance to all clusters rather than only focusing on the nearest cluster (which is the case with k-Means) during the expectation step of the algorithm. During the maximization step, not only can the location of the cluster center move, the shape of the cluster can also be updated. This algorithm is good at finding crescent-shaped clusters which are not present in this data and thus the clusters did not differ much from those found by k-Means. This results of applying the Gaussian Mixture were not an improvement over those of k-Means.

Finally, I moved on to apply the DBSCAN algorithm which I initially believed would be best suited to the data after seeing the results of k-Means and its associated between-cluster SSE and Silhouette coefficients. After describing how DBSCAN differs from k-Means and Gaussian Mixture, as well as how DBSCN assigns core, border, and noise points, I applied it to the data set four different times. DBSCAN has two important parameters: epsilon (Eps) and MinPts. Epsilon defines the maximum distance which can be between two points from them to be considered in the same neighborhood. Lower values of epsilon results in DBSCAN finding only very dense clusters and labelling many data points in sparser regions as noise points. Higher values of epsilon allow DBSCAN to find sparser clusters, resulting in the labelling of fewer noise points. The MinPts parameter defines the minimum number of points which need to fall within any one

data point's surrounding neighborhood for it to be considered a noise point. Higher values of MinPts results in DBSCAN labelling less points as core points, thus, reducing the size of clusters overall. Because DBSCAN is capable of finding uniquely-shaped clusters based on the tuning of its parameters, I altered the value of epsilon to visualize the results of DBSCAN for Eps = 0.02, 0.05, 0.07, and 0.1. I chose not to alter the MinPts parameters as it is set to 5 points to a cluster by default which I believe is appropriate.

A discussion about the results of DBSCAN, in terms of the nature of the clusters and how noise points were labelled for each value of Eps, is included in the notebook. Overall, DBSCAN was much more capable of discerning between the natural clusters in this data. DBSCAN's method of labelling core, border, and noise points allowed the natural clusters to be labelled correctly in a way which k-Means and Gaussian Mixture could not. DSCAN proved to be robust to noise in the data but had trouble finding clusters of differing densities for any one run through. To briefly summarize why this is the case, Eps can be set to only one value per run and so at low values of Eps many sparser clusters are mislabeled as noise simply because they surround denser regions. This discussion concluded the unsupervised learning section of my project.

Moving on to supervised learning, I trained all models on both the balanced and unbalanced datasets using both the holdout method and 10-fold cross-validation (detailed in the notebook). At this point the mapped data contained only values of 0 and 1 and thus was perfect for training an artificial neural network. I first fed the unbalanced data through a multilayered perceptron specifying 178 nodes in the hidden layer, one for each attribute of the mapped data. This classifier trains using backpropagation to adjust weights between layers and minimize classification error rates by implementing gradient descent (see notebook). First, using holdout, I split the data into test and training sets and trained the model with the unbalanced training data.

Next, I tested the classifier on the testing set and then printed out its confusion matrix and classification report. This report contained multiple performance metrics, but I was particularly interested in precision and recall (with most emphasis given to recall). Accuracy was not of interest to me with the imbalanced data because the class labels fell into the non-fatal class 98% of the time and thus a classifier could simply predict all labels to be non-fatal and be correct 98% of the time without really learning anything useful. Within the notebook, I go into detail about precision, recall, the confusion matrix, false positives, false negatives, true positives, true negatives, sensitivity, and specify.

Importantly, I determined that the number of false negatives in this case were far more important the number of false positives. False negatives are very important in this circumstance because it is for more important to not misclassify a fatal collision as non-fatal than it is to misclassify a non-fatal collision as fatal. In general, predicting the rare class correctly is very important in this case so for this reason I was most interested in recall. Overall, the model must have high sensitivity, meaning its true positive rate must be very good. The true positive rate cannot be very high if the number of false negatives made by the model are also high; the model really ought to predict fatalities correctly to be adequate. Because the number of false positives is not much of a concern, the true negative rate is not a concern and thus the specificity of the model is not so important. Because I was more interested in recall than precision, I did not give importance to the F1 metric either.

The recall of the neural net with the unbalanced data was very low at 52% for the fatal class; many fatal collisions were in fact predicted to be non-fatal and thus the false negative rate was high. This was not surprising because fatal collisions were hugely outnumbered in terms of representation in the training data and so the neural net was built as result of learning from far

more non-fatal records. The precision of the model was far better at 78%, meaning the model did not predict many false positives. Although this is nice to see, I have already mentioned that I am not as interested in the precision of the classifier. Using 10-fold cross-validation, I was particularly interested in verifying whether performance metrics would drop compared to those associated with the holdout method. If the performance of the model significantly drops after k-fold cross-validation, then it is reasonable to believe that a considerable amount of the classifier's performance can be attributed to overfitting the training data. This can occur due to a lack of representative samples in the training set. In this data, the fatal class would not be well-represented in the training set. The precision and recall were quite stable as result of cross-validation and so overfitting was likely not a concern. Overall, the model did not perform optimally in the way which is important to the current task of accurately predicting the fatal class.

This same classifier was then built using the balanced data set in which the non-fatal class had been randomly under-sampled. Considering the classes were equally represented in the balanced data, I was interested in overall accuracy rather than precision and recall. The confusion matrix model showed that out of 1091 records predicted to be the fatal class, 926 of those predictions were correct. Of the 1082 predictions made for the non-fatal class, 881 of those predictions were correct. Thus, the overall accuracy of the model was 83.16%, which is strong. Applying this classifier to the balanced data has great benefits: using such data allows the model to learn an equal amount of information from records associated with both classes. Randomly under-sampling the dominant class has created an environment where the classifier can learn the characteristics of collisions which resulted in a fatality without being preoccupied with learning the characteristics of non-fatal collisions. This is very important if we want to correctly predict

non-fatal and fatal collisions. Following 10-fold cross-validation, the accuracy of the model was verified at 76% which is also acceptable. It appears that the model did overfit the training data slightly while using the holdout method. Although these accuracies are acceptable and even quite good, in the real world I would want to be far more confident in a classifier which predicted something as sensitive as human fatality. An accuracy of 83% or 76% is completely relative to the problem at hand. An accuracy of 83% when predicting a less sensitive outcome is far more acceptable than the same accuracy when predicting a sensitive one.

Moving to the next supervised model, as a primer for a random forest ensemble (which I was interested in testing), I began with the induction of a decision tree. It made sense for me to first look at a decision tree so that I could compare its performance on the balanced and unbalanced data with its ensemble method. Furthermore, I wanted to look at the performance of a decision tree because the splits it would make with all binary attributes would be quite simple. After growing the tree from the training set, the precision and recall of the classifier regarding the fatal class were both very bad (hovering around 54-57%). These metrics for the non-fatal class were nearly 100%. This is understandable because as the decision tree tried to appropriately split the data, it built these splits in a way which was biased to the majority class. It is likely that the decision tree is uncertain how to split the data to accurately predict the rare class because not enough records associated with the rare class were represented in the training data.

One issue with decision trees is that they can continue to grow to the point where the data has been so refined from the number of splits that noise in the training data is modelled. Modelling noise is never desirable; modelling noise rather than general relationships results in a model which might fit the present data better but is less generalizable. For this reason, it is important to validate the model's performance with cross-validation. Precision and recall were

similarly weak after cross-validation. When a decision tree was induced from the balanced data, it had an accuracy of about 79%, which dropped to 73% after cross-validation. As the data did not have noise in the form of redundant attributes which the tree would be robust to, I expected similar results to those attained. Because the required splits for this data would all be binary, and the decision tree would not be limited in modelling complex relationships amongst continuous attributes, I expected it to perform slightly better overall.

Extending beyond a single decision tree, I was most interested in comparing the decision tree to a random forest. Random forests are an ensemble method based on decision trees. The process of growing a random forest as well as how the ensemble classifies based on a majority voting scheme rather than a single classifier is described in more detail within the notebook. When trained on the unbalanced dataset, the confusion matrix of the random forest was very interesting. Compared to the decision tree, the ensemble had almost zero false positive predictions; the precision of the ensemble was 99%. Furthermore, the ensemble wrongly predicted more fatal collisions as non-fatal then it correctly predicted fatal collisions. As result, the recall of the ensemble was very low (45%) since the number of false negatives outnumbered the number of true positives. If this were a different scenario in which false positives were far more important to reduce than false negatives, this ensemble would have performed incredibly well. Because wrongly predicting fatal accidents as non-fatal (false negative) is much more important to this analysis than false positives, the poor recall is a concern.

More carefully considering the performance of the random forest, it appears that the ensemble was able to arrive at a vote which could predict, with incredible accuracy, which records in the data were *not* fatal collisions, but it had a very hard time determining which collisions *were* fatal. Overall it seems that the random forest knew which characteristics *excluded*

collisions from being classified as fatal but was not nearly as good at determining characteristics which *included* collisions into the fatal class and not the non-fatal class. These inclusionary criteria developed by the ensemble seem to be very vague. Because of this, many fatal collisions were predicted to be non-fatal but almost no non-fatal collisions were predicted to be fatal. Perhaps the ensemble could collectively decide on exclusionary criteria but not inclusionary criteria because the characteristics which include a record as a fatal collision while also excluding it from being a non-fatal collision are much subtler.

Precision and recall did not noticeably change with cross-validation. When the random forest was grown using the balanced data, ensemble voted on class labels with 79% overall accuracy (the mean of the accuracy of all 10-folds of cross-validation). Not surprisingly, this was an improvement over the single decision tree because the ensemble only makes wrong predictions if more than half of the individual decision tress within the random forest made wrong predictions. Because of this, the error rate of the ensemble is lower than that of the individual classifiers. Assuming the predictions made by the individual classifiers are better than random guesses, ensembles perform better than any one individual classifier. In other words, in the case of ensemble methods, the sum is greater than that of its parts.

At this point I ended my exploration of the collision data base for the time being. My goal was to end with a very in-depth discussion and application of support vector machines, but the run time was incredible, so I will trouble shoot this and explore it further in the future. Towards the end of the notebook you can see that I have a detailed exploration of support vector machines all set up; I have all the code and organization to apply linear and non-linear SVM to the balanced and unbalanced data sets. An interesting thing I wanted to try was tune the hyper-parameter which hardens and softens the margins, allowing for classes to wander inside the

margins if the overall fit of the model improves as result. The most interesting thing I wanted to explore, though, was using weighted margins to predict unbalanced classes. The parameters of SVM can be altered such that the penalty for misclassifying the rare class is increased; this would allow the model to predict fatal collisions with greater quality. By informing SVM that you want the class weights to be balanced, the class weights are adjusted in proportion to their representation in the data such that the rare class does not get overwhelmed by the dominant class when learning a model. Using class weights, SVM implicitly balances classes without balancing the physical representation of the classes within the training data. This is important to the current problem because, as previously seen and discussed, records associated with fatal collisions effectively are dominated by those associated with non-fatal collisions. An explanation of linear SVM, non-linear SVM (including the kernel trick) and hard and soft margins, as well as class weighting is included in the notebook. All associated code in this section is commented out.

Reference:

Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. 2005. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.