Introduction

Problem Description- ABC Insurance Company recently discovered an upward trend in insurance claims in Montgomery County, Maryland over the last few quarters. ABC Insurance Company wants to keep the cost of insurance premiums to customers as low as possible. So when they begin to see a spike in incident claims in a particular area, they decided to look into the data to ensure that their customers were in the correct risk pools..

Motivation- The main motivation of traffic violation data analysis is to build a model that can be used to classify the current customers of the insurance company as well as new customers. This needs to be done in a such a manner that the insurance company can quickly leverage the data model so they can classify new customers and determine what risk pool they should be in. By determining what risk pool the customers should be in, ABC can ultimately avoid the loss of capital due to high payouts on customers with low premium policies.

Report Organization- Our report starts with data exploration. In this section we describe how our data was cleaned and our attributes were selected for our analysis. In methodology, we describe how we processed the data and applied classifiers: Naive Bayes, OneR and Nearest Neighbor. Then we discuss how we evaluated each classifier. In the logic of the problem section, we summarized our checkpoints, pros & cons and suggestions to critical thinking tool based on our experience while doing this whole project. At last, we concluded which classifier predicts best results to categorize drivers and explained how it differed from other classifiers.

Data Exploration

Data Set- The data set collected by Montgomery county sheriff's department contains traffic violation instances from January 01, 2012 to May 16, 2015. There are 740,707 records, and 34 attributes that are grouped into three attribute groups: **incident**, **driver**, and **vehicle**. After removing non-vehicle violations and some records that are outside scope of our effort the size of the data set is 734,546. The attributes of each group are shown in **Appendix A**, and detailed attribute descriptions are shown in **Appendix B**.

Data Cleaning- There were two attributes that required significant amount of cleaning due to spelling errors, typos and junk data. These two attributes were *Make* and *Model*. We applied brute force methods to manually clean the *Make*, however we were not able to do the same with *Model*. To clean this attribute, we utilized the Levenshtein Distance algorithm in a java program to correct the spelling. A master file with about 900 correct spellings of vehicles was generated and compared with the models from our dataset. Correct spellings were generated based on the number of required edits for a given word as we calculated Levenshtein Distance. Generally, but not always the suggestion with the lower number of edits was chosen as the replacement spelling. In the uncleaned dataset, Model contained over 10,000 distinct values. This was reduced to about 1,800 in the clean version. We also merged similar values in the Color column to avoid overfitting. For example, light blue and dark blue were all merged into blue. This was done in Weka filter *MergeTwoValues*.

Missing values and Outliers- In our data set, twelve attributes have missing values. Of these twelve attributes, *Article, Latitude, Longitude, Geolocation, State, Driver city, Driver state* and *DL state* were not used in our analysis, so we did not update them. The remaining attributes with missing values were *Year, Make, Model* and *Color*. We replaced these missing values with the mode attribute value using the Weka filter *ReplaceMissingValues*. The mode values were 2006 for *Year*, TOYOTA for *Make*, ACCORD for *Model* and BLACK for *Color*.

We treated and any records with the Year (car made) attribute value earlier than 1980 or later than 2016 outliers, and we removed these records using weka filter *RemoveFrequentValues*. We did this because we wanted the modes that were generated to be built around later model vehicles and not novelty classic cars other rare vehicles. We used the same method to remove any records with the driver Gender value was unknown. We removed incidents in the dataset that were non-vehicle related offenses as well.

<u>Methodology</u>

Data preprocessing- In order to perform classifications, we added a class label attribute called *RiskPool* with values of Medium, High and Very High. This attribute classified the aggressiveness of the drivers. We used the attributes in incident domain as criteria for assigning the class values. The details can be found in **Appendix E**. The attributes about the driver and vehicle will be used as input to the classification algorithms. These attributes include Commercial_License, Race, Gender, VehicleType, Year, Make, Model and Color. The following attributes from the driver/vehicle groups have been removed: Driver_City, Driver_State, DL_State, and State. These attributes were removed because none of them contribute to a new driver being assigned to a risk pool. Additionally because our dataset is specifically related to Montgomery county Maryland the drivers geographical information presented in these attributes is largely populated with attribute values that are all from the state of Maryland and the cities within the county. For these reasons these attributes were omitted from the analysis. The final sets of the domain-attributes are shown in **Appendix C**.

Since the original data file contained over 700K records, we used Weka filter *Resample* as sampling method to reduce the computation work. This was necessary because of the limitations of Weka when processing a dataset of this size. Details of sampling for each algorithm are discussed in later sections.

<u>Algorithms</u>

Naive Bayes- Naive Bayes is a simple yet powerful and typically accurate classifier. The algorithm is based on Bayes Theorem. According to Ning-Tang, Steinbach and Kumar, Bayes Theorem is a statistical principle for combining prior knowledge of the known classes with new evidence from gathered data to generate a class label. Naive Bayes algorithm has an assumption that attributes are conditionally independent given the class label. This is why Naive Bayes is considered Naive, because this is a naive assumption.

Naive Bayes does not have many parameters to configure, however that does not mean that we simply ran the algorithm without trying some different configurations within our dataset. One of the levers that we played with for this algorithm was sample size. We also reduced the range of the attributes in the dataset. This was accomplished by only looking at instances with the 10 most frequent manufactures, or the most 10 frequent car models. Additionally we looked at using fewer attributes in the classification model. Reducing the scope of the data set did not have a significant impact on the accuracy of the Naive Bayes classifier.

Based on the results of the Naive Bayes classifier experiments we have been able to determine that each attribute in our data set is in fact statistically independent from the class labels (RiskPool). We proved this by removing each attribute one-at-a-time and re-running the classifier and checking the accuracy and ROC value. Each time the value changed, but by an insignificant amount. Naive Bayes behaved as we had expected it to, as the attributes that we used to classify our records were independent from the class labels. The most interesting result from the Naive Bayes classifier was that as the sample size of the data set was reduced down to 2%, the results did not suffer all that much. We had expected that with such a smaller sample size that the results would have decreased by a larger increment.

OneR- The **OneR** rule was one of the Preliminary Learning algorithm used by us for classification because it is very much simple yet highly accurate and these can be explained by the results given below. Also, we thought that it would be a good algorithm as it generates one rule for each predictor and then generates a rule that has the smallest error. This rule is slightly less accurate than many of the algorithms which are state of the art.

We tried changing the seed value and the sample size of the dataset so as to achieve better results but even after that, we didn't find a change in the accuracy of approximately more than 1%. OneR gave us the second best classification accuracy among all the other algorithms that we have tried. We also used the area under the ROC Curve as a performance measure and OneR had the worst among all of the algorithms. The comparisons are shown in the appendix.

The results of OneR indicates that although the accuracy is good for this algorithm, the area under the ROC curve shows how the test used to determine these accuracies is fairly poor.

KNN- Nearest neighbor which is an instance based learning technique allows us to use a training data set with known output to predict output for new data set. In our data set, we have derived customer's classification in medium, high and very high risk drivers. If we create a classification tree with all the attributes, we would have few hundred thousand branches for all attributes. This issue can be fixed by **Nearest Neighbor** Classification or **K-NN** in a very efficient manner as sample size is scalable to whole data set. We have considered nearest neighbor value k = 1, 3 and 5 and observed that as value of K increases, accuracy of classified instances increased marginally for 10 cross fold validation and 66% split estimation methods. We have used performance measure - Area under the ROC curve and KNN had highest average value of ROC curve. We have got maximum accuracy of 75.48% for classified instances and weighted average area under ROC curve of 0.724 with 50% sample size and 66% split estimation which is the highest accuracy among all classifiers. Also we found that when sample size was 50% or more than 50%, 10 cross fold validation test was taking more than 24 hours of time to show results for any value of k though accuracy was increased with large sample size.

Model Evaluation: The strategy we employed for our model evaluation was to utilize a cost matrix for comparing models with a similar sample size, and then using ROC Area for comparing models with different sample sizes and different test set strategies. We choose two methodologies for a couple reasons. First off, it is always helpful to have more than one way to evaluate a model. Think of it as a second opinion. Additionally limitations of the cost model came into play as well. The cost matrix cannot be used to compare models with different characteristics such as testing strategy and sample sizes. This is because the cost matrix takes into consideration the number of instances classified. When the number of instances is different, the evaluation metric is not valid. For example, if we tested a sample size of 50% and used 10-fold Cross-validation we get 367,273 instances classified, however when we use a 66% split we end up with 123,873 instances classified because of how the test set was created. As a result the model that was tested with the 66% split would have a lower cost simply because it had fewer instances classified. This is where ROC Area comes into play. We choose the ROC (Receiver Operating Characteristic) curve because it looks at the classifier's ability to correctly classify the instances of a dataset by plotting the true positive rate against the false positive rate. The size of the dataset is not taken into considerations, only the success and failure rates of the classifiers. By using the ROC curve we are able to compare models with different test sets and different sample sizes. This is because the ROC curve presents the effectivity of the classification model on a scale of 0.0-1.0. The closer to 1.0 your value is the stronger the classification model is.

When we constructed our cost matrix we constructed it such that the True Positives and True Negatives were weighted to reduce the cost of the model. False Positives and False Negatives will increase the cost of the model. Each of the weights can be viewed in **Appendix D.** Generally speaking, the cost model puts more weight on high risk drivers that are not classified correctly and less weight on medium risk drivers correctly classified.

We used a combination of both 10 fold cross validation and percentage split for our test data. The results varied from method to method as to what produced a better output from the classifiers, so we tried to evaluate each model on both the estimation strategies. We found that Naive Bayes gave us slightly better results when using 10 fold cross validation. KNN seemed to have similar results with each methodology. The output we saw for OneR was almost identical with each strategy.

Naive Bayes performed reasonably well in terms of accuracy, however all of models scored fairly low on the ROC Curve. The best performing models were the modes that did not have any sampling performed on them. There was not much difference when comparing the two estimation methodologies we implemented. All of the results were within a few decimal points of each other. For KNN we saw some interesting behavior. We had initially processed the data set with a 10% sample size and 10 fold cross validation. We were not successful at getting any larger samples sizes to process when using KNN and a sample set larger than 10% when using 10-fold cross validation for the test set. Once we switched over to the 66% split we were able to get a 50% sample to run for K=5 and see a very strong accuracy and high ROC curve number. Additionally the speed of the model also increased without the overhead of 10-fold cross validation. The performance for OneR was fairly consistent across the board. Neither Estimation strategy or Sampling size changed the outcome of the Algorithm. All of the results had an accuracy right around 73% and a ROC Curve number of approximately 0.501. A summary of all test results can be found in **Appendix F**.

Logic of the problem

Our experience through Checkpoint 1 to 3- We used the critical thinking tool for two major purposes, to outline the problem we were going to solve, and to help us stay on track along the development of the project. We found that the eight elements of thoughts was a useful tool to train and organize our thoughts. After initially going through the logic of the problem, we were clear about the purpose and the point of view; We were going to think from the perspective of an insurance company to build a more accurate model for evaluating driver's aggressiveness. We raised some basic questions, such as what are the factors we use in the model to evaluate a driver, and we collected information and looked up concepts to answer these questions. We found a data set of traffic violation of Montgomery County Maryland for the last four years, and applied the knowledge of aggressive driving measurements.

As we developing the model, we found more questions unanswered, and we added them in. For example, "how do we determine the impact levels of accident factors" was added when we were assigning the class values. We also realized that we need to make assumptions in order to do analysis. For example, we made a series of assumptions based on our interpretation of the incident attributes to decide the class values. The implications and inferences in the tool helped us to recognize and clarify what results we get from our model and their meanings, and these are what we need to present and clarify with our clients and audiences.

Pros and Cons of this thinking procedure- The critical thinking tool automatically forms a logical relation between the eight elements of thoughts, which is beneficial for team projects. Since each member fills their pieces of thoughts into the eight elements, the tool connects their thoughts through a unified logic that ensures all members think in the same direction to solve the problem. Our team values two elements most important: the assumption and implication. We frequently referred to the assumption when analyzing the problem and building the models, it needs to be documented and understood by all involved. And the implication allows the shareholders to understand the results of our model, so it needs to be clarified. On the other hand, the tool has some limited usability. Users are unable to relate data visualization or arff file with the tool. Also the UI design of the web page is outdated, and there is not formatting option in text fields.

Recommendations and Suggestions- We did not find any step that need to be added or deleted, but we did hope the tool can help demonstrating the associations and dependencies between the eight steps. For example, the information is used to answer the questions.

Conclusion- Our project is intended for ABC insurance company in Montgomery County, Maryland. For this analysis, we used three primary algorithms; Naive Bayes, the OneR and KNN. Each was run with different sample sizes and test options such as 10 folds and 66% split. We found best accuracy in certain cases of KNN followed by OneR and Naive Bayes. Naive Bayes and OneR were almost identical but upon looking at the area under the ROC curve we found that Naive bayes had the edge over OneR. Models with a high sample size and 66% test split of KNN had the highest ROC value overall. The algorithm that works best in this case is KNN with the data sampled to 50% and the k value is kept to 5. The accuracy obtained is approximately 75% and the average area under ROC is 0.724.

We faced some issues during the whole process. Initially our team used four classification methodologies to classify our dataset and build different classification models. Because of poor performance issues with J48 we have eliminated that model from the results. The performance issues were both related to the classification results. When we ran the algorithms on the original data set, the results took a lot time to be generated because of the sheer amount and complexity of the data. Also during the classification, we ran into situations where the algorithm failed to classify high and very high risk from a few of the outputs. This was resolved using Weka preprocessing filters which can be found in **Appendix B.** Some of the filters were for merging similar colors, removing attributes which were used to build the class labels, using only the last 35 years' worth of data, etc.

Future prospects of our project would be to compare our model to the industry standards and look for a chance of improvement if possible. Due to the sheer quantity of data and the number of problems that persists with the spellings and other cleanup that needs to be done, improving the quality of data is of utmost priority so as to achieve even better results. Also we haven't even looked at the plan costs as a part of our model evaluation and that is an area we would like to get into so as to understand how it impacts the overall revenue for ABC Insurance Company.

Appendix A: Attribute summary

Attribute group	**Attribute list**
Incident	Date_Of_Stop, Time_Of_Stop, Agency, SubAgency, Description, Location, Latitude, Longitude, Accident, Belts, Personal_Injury, Property_Damage, Fatal, HAZMAT, Alcohol, Work_Zone, Violation_Type, Charge, Article, Contributed_To_Accident, Arrest_Type, Geolocation
1Driver	Commercial_License, Race, Gender, Driver_City, Driver_State, DL_State
Vehicle	State, VehicleType, Year, Make, Model, Color

Appendix B: Attribute description

Attribute	Description	Group	Data Type	Attribute Type
Date_Of_Stop	Date of the traffic violation.	Incident	Date	Interval
Time_of_stop	Date of the traffic violation.	Incident	Time	Interval
Agency	Agency issuing the traffic violation. (Example: MCP is Montgomery County Police)	Incident	Text	Nominal
SubAgency	Court code representing the district of assignment of the officer.	Incident	Text	Nominal
Description	Text description of the specific charge.	Incident	Text	Nominal
Location	Location of the violation, usually an address or intersection.	Incident	Text	Nominal
Latitude	Latitude location of the traffic violation.	Incident	Text	Interval
Longitude	Longitude location of the traffic violation.	Incident	Text	Interval
Accident	If traffic violation involved an accident.	Incident	Text	Ordinal
Belts	If traffic violation involved a seat belt violation.	Incident	Text	Ordinal
Personal_Injury	If traffic violation involved Personal Injury.	Incident	Text	Ordinal
Property_Damage	If traffic violation involved Property Damage.	Incident	Text	Ordinal
Fatal	If traffic violation involved a fatality.	Incident	Text	Ordinal
Commercial_License	If driver holds a Commercial Driver's License.	Driver	Text	Ordinal
HAZMAT	If the traffic violation involved hazardous materials.	Incident	Text	Ordinal
Commercial Vehicle	If the vehicle committing the traffic violation is a commercial vehicle.	Incident	Text	Ordinal
Alcohol	If the traffic violation included an alcohol related	Incident	Text	Ordinal
Work_Zone	If the traffic violation was in a work zone.	Incident	Text	Ordinal
State	State issuing the vehicle registration.	Vehicle	Text	Nominal

VehicleType	Type of vehicle (Examples: Automobile, Station Wagon, Heavy Duty Truck, etc.)	Vehicle	Text	Nominal
Year	Year vehicle was made.	Vehicle	Number	Interval
Make	Manufacturer of the vehicle (Examples: Ford, Chevy, Honda, Toyota, etc.)	Vehicle	Text	Nominal
Model	Model of the vehicle.	Vehicle	Text	Nominal
Color	Color of the vehicle	Vehicle	Text	Nominal
Violation Type	Violation type. (Examples: Warning, Citation, ESERO)	Incident	Text	Nominal
Charge	Numeric code for the specific charge.	Incident	Text	Nominal
Article	Article of State Law. (TA = Transportation Article, MR = Maryland Rules)	Incident	Text	Nominal
Contributed_To_Accident	If the traffic violation was a contributing factor in an accident.	Incident	Text	Nominal
Race	Race of the driver. (Example: Asian, Black, White, Other, etc.)	Driver	Text	Nominal
Gender	Gender of the driver (F = Female, M = Male)	Driver	Text	Nominal
Driver City	City of the driver's home address.	Driver	Text	Nominal
Driver_State	State of the driver's home address.	Driver	Text	Nominal
DL State	State issuing the Driver's License.	Driver	Text	Nominal
Arrest_Type	Type of Arrest (A = Marked, B = Unmarked, etc.)	Incident	Text	Nominal
Geolocation	Geo-coded location information.	Incident	Number	Interval

Appendix C: Attributes used in analysis

Attribute group	**Attribute list**
Incident	Description, Belts, Personal_Injury, Property_Damage, Fatal, Alcohol, Work_Zone, Contributed_To_Accident
Driver	Commercial_License, Race, Gender
Vehicle	VehicleType, Year, Make, Model, Color

Appendix D: Weka Preprocess filter

Relation: traffData1023

Pre-process filter	Description	Why was it used
-weka.filters.unsupervised.attribute.Reorder-R2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,1		
-weka.filters.unsupervised.attribute.Remove-R1-4,6-8,14	removed attributes that you don't want in the classifier	Remove unneeded attributes
-weka.filters.unsupervised.attribute.NumericToNominal-R3	Unsupervised filter that converts numeric attributes into nominal attributes	Convert the Year attribute from a numeric attribute to a nominal attribute.
-weka.filters.unsupervised.attribute.ReplaceMissingValues	Replaces all missing values with the mean or mode for that attribute	Used to replace all missing values for the dataset.
-weka.filters.unsupervised.instance.RemoveFrequentValues-C8-N2	Keeps records that have the most frequent values	Remove “unknown” gender
-weka.filters.unsupervised.instance.RemoveFrequentValues-C3-N35	Keeps records that have the most frequent values	Preserve only the 35 years with the highest counts
-weka.filters.unsupervised.attribute.MergeTwoValues-C6-F2-S19	Combines similar attributes.	Merge similar colors
-weka.filters.unsupervised.attribute.MergeTwoValues-C6-F10-S16	Combines similar attributes.	Merge similar colors
-weka.filters.unsupervised.attribute.MergeTwoValues-C6-F10-S11	Combines similar attributes.	Merge similar colors
-weka.filters.unsupervised.attribute.MergeTwoValues-C6-F2-S13	Combines similar attributes.	Merge similar colors
-weka.filters.unsupervised.attribute.MergeTwoValues-C6-F11-S13	Combines similar attributes.	Merge similar colors

Appendix E: Class Label Logic

Attribute	Condition	Class Label
Alcohol	Yes	Very High
Fatal	Yes	Very high
Contributed to Accident	Yes	Very High
No Seat Belts	Yes	High
Personal Injury	Yes	High
Description contains “SPEED” + No Seat Belts	Yes	Very High
Description contains “SPEED” + Personal Injury	Yes	Very High
Description contains “SPEED” + Property Damage	Yes	Very High
Description contains “SPEED” + Work Zone	Yes	Very High
Personal Injury + No Seat belts	Yes + Yes	Very High
No seat belts + Property Damage	Yes + Yes	Very High
Property Damage	Yes	High
Speed	Present in description	High
Work Zone	Yes	High
All others		Medium

Appendix E: Cost Matrix

Actual	Predicted		
	Medium	High	Very High
Medium	-5	5	15
High	15	-10	10
Very High	20	10	-15

Appendix F: Testing Output

Test ID	Learning Algorithm	Estimation Strategy	Sample Size	Accuracy	Cost	ROC Curve (Weighted Avg)	K Value
NB1	Naive Bayes	10 Fold Cross Validation	734,546	73.60%	379,720	0.609	N/A
NB2	Naive Bayes	10 Fold Cross Validation	550,910	73.50%	285,335	0.607	N/A
NB3	Naive Bayes	10 Fold Cross Validation	367,273	73.48%	195,370	0.606	N/A
NB4	Naive Bayes	10 Fold Cross Validation	183,637	73.29%	101,870	0.603	N/A
NB5	Naive Bayes	10 Fold Cross Validation	73,455	73.20%	39,135	0.601	N/A
NB6	Naive Bayes	10 Fold Cross Validation	14,691	72.80%	9,140	0.586	N/A
NB7	Naive Bayes	66% Split	734,546	73.58%	124,590	0.607	N/A
NB8	Naive Bayes	66% Split	550,910	73.64%	93,875	0.608	N/A
NB9	Naive Bayes	66% Split	367,273	73.45%	61,910	0.605	N/A
NB10	Naive Bayes	66% Split	183,637	73.45%	30,750	0.604	N/A
NB11	Naive Bayes	66% Split	73,455	73.07%	14,195	0.595	N/A
NB12	Naive Bayes	66% Split	14,691	73.49%	3,235	0.578	N/A
K1	KNN	10 Fold Cross Validation	73,455	70.33%	33,830	0.581	1
K2	KNN	10 Fold Cross Validation	73,455	72.50%	40,000	0.576	1
K3	KNN	10 Fold Cross Validation	73,455	73.17%	40,605	0.578	3
K4	KNN	66% Split	73,455	72.50%	40,000	0.576	5
K5	KNN	66% Split	73,455	69.64%	14,815	0.574	3
K6	KNN	66% Split	73,455	72.90%	15,480	0.572	1
K7	KNN	66% Split	367,273	72.97%	15,480	0.62	5

K8	KNN	66% Split	367,273	75.48%	-91,375	0.724	3
K9	KNN	66% Split	367,273	71.97%	11,515	0.636	5
K10	KNN	10 Fold	183,637	74.14%	-83,015	0.689	1
K11	KNN	10 Fold Cross Validation	14,691	68.82%	10,665	0.553	1
K12	KNN	10 Fold Cross Validation	14,691	72.00%	9,935	0.563	1
K13	KNN	10 Fold Cross Validation	14,691	72.77%	9,580	0.574	3
K14	KNN	10 Fold Cross Validation	73,455	72.02%	2,415	0.619	5
K15	KNN	10 Fold Cross Validation	73,455	72.70%	35,920	0.593	1
K16	KNN	10 Fold Cross Validation	73,455	73.71%	-27,945	0.659	3
K17	KNN	10 Fold Cross Validation	183,637	74.14%	-83,015	0.659	5
OR1	OneR	10 Fold Cross Validation	734,546	73.77%	440,615	0.501	N/A
OR2	OneR	66% Split	734,546	73.88%	144,285	0.501	N/A
OR3	OneR	10 Fold Cross Validation	367,273	73.71%	222,445	0.501	N/A
OR4	OneR	66% split	367,273	73.82%	74,480	0.501	N/A
OR5	OneR	10 Fold Cross Validation	183,637	73.63%	115,650	0.501	N/A
OR6	OneR	66% split	183,637	73.62%	73.62%	0.501	N/A
OR7	OneR	10 Fold Cross Validation	73,455	73.97%	73.97%	0.501	N/A
OR8	OneR	66% split	39,445	74.01%	74.01%	0.501	N/A

Appendix G: Extra Credit Work

Issue	After assigning our class labels we ended up with over 500K medium risk drivers. We wanted to group these instances into groups so that we could apply some more granular class labels to them.
Possible solutions	● Brute force - We could have manually sorted, grouped and classified the incident descriptions and assigned class labels. This approach is not really feasible, simply because there are over 8,300 unique values that span 550,000 records. ● Implement a document clustering algorithm to group similar incident descriptions, then assign class labels to each cluster. There are two algorithms that are applicable here to use, both the K-Means clustering and Hierarchical Agglomerative Clustering.
Attempt 1: K-Means Clustering	Our first attempt at this problem was to write an algorithm that would cluster the document descriptions was done with K-Means clustering. **Pre Processing** Our algorithm read in a TSV input file that contained the incidentID number and the incident description. The algorithm then removed the stop words from the text description field so that we could avoid adding frequent words to our term list. One the stop words had been removed we then passed the descriptions into the Porter Stemming Algorithm. The Porter Stemming algorithm reduces words to their common roots. This helps reduce the number of records added to the term list, and therefore help identify descriptions that share similarities with others. After the stemmer runs we removed the duplicate descriptions from the list of terms we are going to process. (This step was added after we had attempted Hierarchical Clustering and was not part of our initial K-Means testing.) The next step in the process for computing the description clusters was to build the term list. The term list is a complete list of every term that is found in the document space. Once the term list is complete, the vectors for each description are calculated and assigned. We had encountered an issue when performing clustering where we were getting NaN for the results when computing our centroids. This was the result of when we picked the initial centroids in the data space not having any similarities, so when the cosine similarity was 0.00 between the two descriptions. To remedy this we initialized each vector to 1 for each term. **Clustering** The K-Means clustering was performed by picking K cluster centroids at random within our data space. After the centroids were picked we calculated the cosine similarity between each description and the centroids. Centroids are then reassigned based on similarity. This process is completed until the

	centroids have converged (stopped moving). The output from this algorithm is a set of clusters and a listing of the descriptions that are assigned to that cluster.
Issues with K-Means	In addition to the NaN error mentioned above, once we attempted to scale this algorithm out to meet the needs of our data set we ran into issues. Running on the entire data set was out of the question. When attempting to process even 25% of the dataset the JVM on my desktop crashed within minutes of starting the computation of the description vectors. I adjusted the RAM to 12GB, but the result was the same. We attempted to process many other subsets of data, but the results were never fruitful. Each time the JVM crashed, or spent endless hours computing the vectors. Beyond the scaling issues there were going to be some additional challenges for K-Means. We don't have a good indication as to what the correct K value should be, so our choices are arbitrary. With the amount of time it takes to run the algorithm we need to have some good indication as to where to start the experimentation. The initial values we had chosen on our test set of a couple hundred records looked like they were too small so we were getting clusters with a lot of dissimilarity in them. K-Means is also sensitive to where the initial centroids started, so this can also impact the output of the process. As a result of this we attempted Hierarchical clustering.
Attempt 2 - Hierarchical Clustering	**Pre Processing** The pre processing for Hierarchical clustering is identical to that of K-Means. The step to remove the duplicate descriptions from the data set was added to eliminate the overhead of merging identical documents together. We only needed the algorithm to cluster similar documents, not identical ones. **Clustering** We chose to use single link clustering to implement this algorithm. Our implementation was fairly straight forward. First we looked at each description as a cluster, then calculated the cosine similarity of each description vector to the others. The most similar instances were then merged and added as a new cluster. This process was repeated until we had no more clusters to merge.
Issues with Hierarchical Clustering	Similar to K-Means the hierarchical clustering algorithm worked on smaller data sets, but did not scale well. The other issue we had was that the results were difficult to verify when the data set got large.

Program Details	Project Folder: DescriptionClustering External Libraries used: Apache Commons IO and Commons-lang, Porter Stemmer algorithm Test Data Set: data\clusteringIN.txt Full Data Set: clusteringIN3.txt Code: src\DescriptionClustering Cluster.java - cluster object used in Hierarchical clustering Description.java - Description class DescripotinClustering.java - main java class that performs most of the heavy lifing. Stemmer.java - Stemmer algorithim
Arguments	<KMANS|HIERARCHICAL> <Input Text File> <K-Value> (Optional) Ex. KMEANS clusteringIN.txt 3 HIERARCHICALclusteringIN.txt

Sample results for K=3 and 85 unique descriptions:

```
****Description/centroid assignment*****
Doc 0 is assigned to centroid 0
Doc 1 is assigned to centroid 0
Doc 2 is assigned to centroid 0
Doc 3 is assigned to centroid 0
Doc 4 is assigned to centroid 0
Doc 5 is assigned to centroid 0
Doc 6 is assigned to centroid 0
Doc 7 is assigned to centroid 0
Doc 8 is assigned to centroid 2
Doc 9 is assigned to centroid 0
Doc 10 is assigned to centroid 2
Doc 11 is assigned to centroid 2
Doc 12 is assigned to centroid 2
Doc 13 is assigned to centroid 0
Doc 14 is assigned to centroid 0
Doc 15 is assigned to centroid 0
Doc 16 is assigned to centroid 0
Doc 17 is assigned to centroid 2
Doc 18 is assigned to centroid 0
Doc 19 is assigned to centroid 2
Doc 20 is assigned to centroid 1
Doc 21 is assigned to centroid 0
Doc 22 is assigned to centroid 0
Doc 23 is assigned to centroid 2
Doc 24 is assigned to centroid 0
Doc 25 is assigned to centroid 0
Doc 26 is assigned to centroid 0
Doc 27 is assigned to centroid 2
Doc 28 is assigned to centroid 2
Doc 29 is assigned to centroid 2
Doc 30 is assigned to centroid 2
Doc 31 is assigned to centroid 0
Doc 32 is assigned to centroid 0
Doc 33 is assigned to centroid 0
Doc 34 is assigned to centroid 0
Doc 35 is assigned to centroid 0
Doc 36 is assigned to centroid 0
Doc 37 is assigned to centroid 0
Doc 38 is assigned to centroid 0
Doc 39 is assigned to centroid 2
Doc 40 is assigned to centroid 2
Doc 41 is assigned to centroid 0
Doc 42 is assigned to centroid 0
Doc 43 is assigned to centroid 2
Doc 44 is assigned to centroid 0
Doc 45 is assigned to centroid 0
Doc 46 is assigned to centroid 2
Doc 47 is assigned to centroid 2
Doc 48 is assigned to centroid 0
Doc 49 is assigned to centroid 0
Doc 50 is assigned to centroid 0
Doc 51 is assigned to centroid 0
Doc 52 is assigned to centroid 0
Doc 53 is assigned to centroid 0
Doc 54 is assigned to centroid 0
Doc 55 is assigned to centroid 0
Doc 56 is assigned to centroid 2
Doc 57 is assigned to centroid 0
Doc 58 is assigned to centroid 2
Doc 59 is assigned to centroid 2
Doc 60 is assigned to centroid 2
Doc 61 is assigned to centroid 0
Doc 62 is assigned to centroid 0
Doc 63 is assigned to centroid 2
Doc 64 is assigned to centroid 0
Doc 65 is assigned to centroid 0
Doc 66 is assigned to centroid 2
Doc 67 is assigned to centroid 0
Doc 68 is assigned to centroid 0
Doc 69 is assigned to centroid 2
Doc 70 is assigned to centroid 0
Doc 71 is assigned to centroid 0
Doc 72 is assigned to centroid 0
Doc 73 is assigned to centroid 2
Doc 74 is assigned to centroid 0
Doc 75 is assigned to centroid 0
Doc 76 is assigned to centroid 0
Doc 77 is assigned to centroid 0
Doc 78 is assigned to centroid 2
Doc 79 is assigned to centroid 2
Doc 80 is assigned to centroid 2
Doc 81 is assigned to centroid 0
Doc 82 is assigned to centroid 2
Doc 83 is assigned to centroid 2
Doc 84 is assigned to centroid 2
****Description/centroid assignment*****

Old Centroid 0: 0.806 | 0.768 | 0.753 | 0.776 | 0.738 | 0.723 | 0.701 | 0.708 | 0.701 | 0.701 | 0.701 |
0.708 | 0.729 | 0.764 | 0.821 | 0.708 | 0.701 | 0.723 | 0.701 | 0.701 | 0.753 | 0.701 | 0.701 | 0.701 |
0.723 | 0.701 | 0.708 | 0.731 | 0.701 | 0.708 | 0.753 | 0.716 | 0.771 | 0.701 | 0.701 | 0.731 | 0.701 |
0.701 | 0.693 | 0.708 | 0.708 | 0.731 | 0.701 | 0.701 | 0.701 | 0.693 | 0.693 | 0.693 | 0.693 | 0.723 |
0.701 | 0.708 | 0.693 | 0.693 | 0.693 | 0.693 | 0.708 | 0.693 | 0.693 | 0.693 | 0.701 | 0.701 | 0.701 |
0.708 | 0.701 | 0.708 | 0.708 | 0.701 | 0.723 | 0.708 | 0.731 | 0.693 | 0.693 | 0.693 | 0.693 | 0.701 |
0.701 | 0.701 | 0.701 | 0.708 | 0.693 | 0.693 | 0.693 | 0.701 | 0.708 | 0.693 | 0.693 | 0.693 | 0.693 |
0.716 | 0.701 | 0.701 | 0.693 | 0.723 | 0.701 | 0.701 | 0.701 | 0.701 | 0.701 | 0.701 | 0.701 | 0.701 |
0.701 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.716 |
0.701 | 0.716 | 0.701 | 0.701 | 0.701 | 0.701 | 0.701 | 0.708 | 0.701 | 0.693 | 0.693 | 0.693 | 0.693 |
0.693 | 0.693 | 0.693 | 0.693 | 0.708 | 0.701 | 0.701 | 0.708 | 0.701 | 0.701 | 0.701 | 0.693 | 0.693 |
0.693 | 0.708 | 0.708 | 0.701 | 0.701 | 0.701 | 0.701 | 0.701 | 0.708 | 0.708 | 0.708 | 0.701 | 0.701 |
0.701 | 0.701 | 0.701 | 0.701 | 0.701 | 0.701 | 0.701 | 0.701 | 0.701 | 0.701 | 0.693 | 0.701 | 0.701 |
0.716 | 0.716 | 0.693 | 0.693 | 0.701 | 0.708 | 0.701 | 0.701 | 0.693 | 0.693 | 0.693 | 0.701 | 0.701 |
0.701 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.701 | 0.701 | 0.693 |
0.693 | 0.693 | 0.693 | 0.693 | 0.693 |
Old Centroid 1: 1.099 | 1.099 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 |
0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 |
0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 1.099 | 0.693 | 0.693 | 0.693 | 0.693 |
0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 |
0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 |
0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 1.386 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 |
0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 1.099 | 1.099 | 1.099 | 1.099 | 1.099 | 1.099 | 1.099 |
0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 |
0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 |
0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 |
0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 |
0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 |
0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 |
0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 |
0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 |
0.693 | 0.693 | 0.693 | 0.693 | 0.693 |
Old Centroid 2: 0.720 | 0.892 | 0.707 | 0.693 | 0.693 | 0.693 | 0.707 | 0.693 | 0.693 | 0.693 | 0.693 |
0.693 | 0.693 | 0.693 | 0.707 | 0.693 | 0.707 | 0.693 | 0.707 | 0.693 | 1.014 | 0.720 | 0.707 | 0.693 |
0.720 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.707 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 |
0.720 | 0.707 | 0.720 | 0.707 | 0.720 | 0.693 | 0.693 | 0.774 | 0.707 | 0.747 | 0.774 | 0.909 | 0.882 |
0.788 | 0.788 | 0.720 | 0.720 | 0.720 | 0.720 | 0.815 | 0.720 | 0.747 | 0.720 | 0.693 | 0.693 | 0.693 |
0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.707 | 0.720 | 0.720 | 0.720 | 0.693 |
0.693 | 0.707 | 0.693 | 0.707 | 0.707 | 0.707 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 |
0.707 | 0.693 | 0.707 | 0.707 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 |
0.693 | 0.734 | 0.720 | 0.707 | 0.707 | 0.707 | 0.707 | 0.734 | 0.720 | 0.707 | 0.707 | 0.707 | 0.693 |
0.693 | 0.734 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.707 | 0.707 | 0.707 | 0.707 |
0.707 | 0.707 | 0.707 | 0.720 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.707 | 0.707 |
0.707 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 |
0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.707 | 0.693 | 0.693 |
0.693 | 0.693 | 0.707 | 0.707 | 0.693 | 0.693 | 0.693 | 0.693 | 0.720 | 0.707 | 0.707 | 0.693 | 0.693 |
0.693 | 0.707 | 0.707 | 0.707 | 0.707 | 0.707 | 0.707 | 0.707 | 0.707 | 0.707 | 0.693 | 0.693 | 0.707 |
0.707 | 0.707 | 0.707 | 0.707 | 0.707 |

New Centroid 0: 0.806 | 0.768 | 0.753 | 0.776 | 0.738 | 0.723 | 0.701 | 0.708 | 0.701 | 0.701 | 0.701
| 0.708 | 0.729 | 0.764 | 0.821 | 0.708 | 0.701 | 0.723 | 0.701 | 0.701 | 0.753 | 0.701 | 0.701 | 0.701 |
0.723 | 0.701 | 0.708 | 0.731 | 0.701 | 0.708 | 0.753 | 0.716 | 0.771 | 0.701 | 0.701 | 0.731 | 0.701 |
0.701 | 0.693 | 0.708 | 0.708 | 0.731 | 0.701 | 0.701 | 0.701 | 0.693 | 0.693 | 0.693 | 0.693 | 0.723 |
0.701 | 0.708 | 0.693 | 0.693 | 0.693 | 0.693 | 0.708 | 0.693 | 0.693 | 0.693 | 0.701 | 0.701 | 0.701 |
0.708 | 0.701 | 0.708 | 0.708 | 0.701 | 0.723 | 0.708 | 0.731 | 0.693 | 0.693 | 0.693 | 0.693 | 0.701 |
0.701 | 0.701 | 0.701 | 0.708 | 0.693 | 0.693 | 0.693 | 0.701 | 0.708 | 0.693 | 0.693 | 0.693 | 0.693 |
0.716 | 0.701 | 0.701 | 0.693 | 0.723 | 0.701 | 0.701 | 0.701 | 0.701 | 0.701 | 0.701 | 0.701 | 0.701 |
0.701 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.716 |
0.701 | 0.716 | 0.701 | 0.701 | 0.701 | 0.701 | 0.701 | 0.708 | 0.701 | 0.693 | 0.693 | 0.693 | 0.693 |
0.693 | 0.693 | 0.693 | 0.693 | 0.708 | 0.701 | 0.701 | 0.708 | 0.701 | 0.701 | 0.701 | 0.693 | 0.693 |
0.693 | 0.708 | 0.708 | 0.701 | 0.701 | 0.701 | 0.701 | 0.701 | 0.708 | 0.708 | 0.708 | 0.701 | 0.701 |
0.701 | 0.701 | 0.701 | 0.701 | 0.701 | 0.701 | 0.701 | 0.701 | 0.701 | 0.701 | 0.693 | 0.701 | 0.701 |
0.716 | 0.716 | 0.693 | 0.693 | 0.701 | 0.708 | 0.701 | 0.701 | 0.693 | 0.693 | 0.693 | 0.701 | 0.701 |
0.701 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.701 | 0.701 | 0.693 |
0.693 | 0.693 | 0.693 | 0.693 | 0.693 |
```

New Centroid 1: 1.099 | 1.099 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693
| 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 |
0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 1.099 | 0.693 | 0.693 | 0.693 | 0.693 |
0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 |
0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 |
0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 1.386 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 |
0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 1.099 | 1.099 | 1.099 | 1.099 | 1.099 | 1.099 | 1.099 |
0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 |
0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 |
0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 |
0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 |
0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 |
0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 |
0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 |
0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 |
0.693 | 0.693 | 0.693 | 0.693 | 0.693 |
New Centroid 2: 0.720 | 0.892 | 0.707 | 0.693 | 0.693 | 0.693 | 0.707 | 0.693 | 0.693 | 0.693 | 0.693
| 0.693 | 0.693 | 0.693 | 0.707 | 0.693 | 0.707 | 0.693 | 0.707 | 0.693 | 1.014 | 0.720 | 0.707 | 0.693 |
0.720 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.707 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 |
0.720 | 0.707 | 0.720 | 0.707 | 0.720 | 0.693 | 0.693 | 0.774 | 0.707 | 0.747 | 0.774 | 0.909 | 0.882 |
0.788 | 0.788 | 0.720 | 0.720 | 0.720 | 0.720 | 0.815 | 0.720 | 0.747 | 0.720 | 0.693 | 0.693 | 0.693 |
0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.707 | 0.720 | 0.720 | 0.720 | 0.693 |
0.693 | 0.707 | 0.693 | 0.707 | 0.707 | 0.707 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 |
0.707 | 0.693 | 0.707 | 0.707 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 |
0.693 | 0.734 | 0.720 | 0.707 | 0.707 | 0.707 | 0.707 | 0.734 | 0.720 | 0.707 | 0.707 | 0.707 | 0.693 |
0.693 | 0.734 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.707 | 0.707 | 0.707 | 0.707 |
0.707 | 0.707 | 0.707 | 0.720 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.707 | 0.707 |
0.707 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 |
0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.693 | 0.707 | 0.693 | 0.693 |
0.693 | 0.693 | 0.707 | 0.707 | 0.693 | 0.693 | 0.693 | 0.693 | 0.720 | 0.707 | 0.707 | 0.693 | 0.693 |
0.693 | 0.707 | 0.707 | 0.707 | 0.707 | 0.707 | 0.707 | 0.707 | 0.707 | 0.707 | 0.693 | 0.693 | 0.707 |
0.707 | 0.707 | 0.707 | 0.707 | 0.707 |

Appendix H: Model files and Classifier output.

Our model and output files can be found in the *ModelsAndOutput* folder.

IN the folder there are model files for each test we ran as well as textual representations of the classifier output. They are all named according to the test they represent.

For example, for the first test we ran with Naive Bayes the model file is NB1.txt and the output is NB.model

References

Maryland Department of Legislative Services. (2013). *Code of Maryland (Statutes) [Transportation Article]*. Retrieved from http://mgaleg.maryland.gov/2015RS/Statute_Web/gtr/gtr.pdf

Montgomery County of Maryland. (2015). *Traffic Violations [data.montgomerycountymd.gov]*. Retrieved from https://catalog.data.gov/dataset/traffic-violations-56dda

Tan, P. N., Steinbach, M., Kumar, V. (2005). *Introduction to Data Mining.* Boston, MA: Pearson.