CS-4641-Project

Introduction and Problem Definition

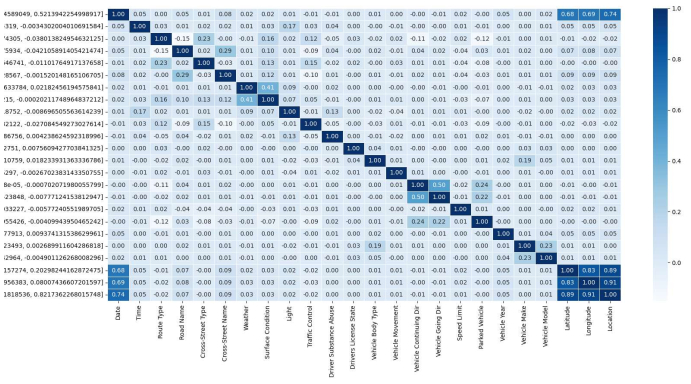
As of 2018, the number of global annual road traffic deaths had reached 1.35 million, with 38,000 fatalities and associated costs of $55 billion in just the United States. In this project, we will develop a car accident risk assessor for Maryland, using 155,108 data points, including situational features like weather, road and traffic conditions, and static features like vehicle condition.

Work already done in this area includes similar data but aims to provide different forms of insight. For example, Wilson (2018) uses a similar dataset for Utah, providing heatmaps of risk per unit time over 7 years. He found that most accidents occurred in high risk areas, but outlined that real time data and more features would result in a better fitted model. Both Wilson (2018) and Cigdem et al. (2018) highlight the problem of Class Imbalance, a problem that has different solutions which include creating non-accident data and filtering data to highlight relevant information. Both Cigdem et al. (2018) and Al-Mistarehi et al. (2022) outline the use of a confusion matrix to measure performance.

Based on corroboration found in the literature on the importance of real time data, we have decided to train and test our data on a pre-existing data, with a goal to develop a model that can provide personalized risk assessments given real-time location, driver, and weather detail.

Methods

In our proposal, we noted that since our data has a large number of dimensions, we would likely need to use an algorithm to reduce the dimensionality. Because our goal for this project is to calculate the likelihood of an accident in general, not the severity of an accident nor other strictly post-accidental information, we removed features like collision location, severity, driver at fault, etc. Then, we used Principal Component Analysis (PCA) to further reduce the dimensionality of our dataset by determining which features were providing the least explanation for variance in the data. The covariance matrix that was created is shown below:



After running PCA, we were able to reduce from 26 features down to 5 principal axes while still retaining 99% of the variance in the data.

The data from PCA is found here.

We will now discuss the unsupervised algorithm we applied. We believed that information on the latitude and longitude (i.e. location) of the driver should play a role in determining the probability of an accident. To determine if a driver was in a higher accident risk zone vs a lower accident risk zone, we decided to run DBSCAN on the latitude and longitude training data.

Now, we will discuss the supervised models we implemented. Before we could run a supervised model, we needed to create a set of non-accident data. Our group decided to experiment with two different methods of creating this "dummy" data. Wilson (2018) initially randomly sampled some number of roads/times when accidents didn't occur, but found that there are a lot of times and roads when accidents simply do not occur often. In addition to this complication, another important problem to solve was differentiating between accident instances versus non-accident instances on roads where accidents happen frequently.

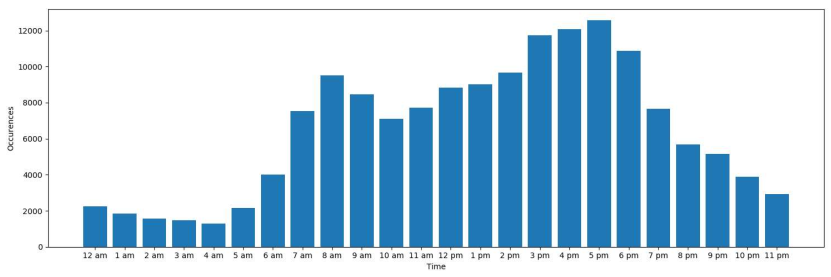
Taking inspiration from Wilson (2018), we chose to use a sampling approach that builds up a set of negative examples that are very similar to our positive examples so that the machine learning model can learn to find the fine differences between when there is and isn't an accident. There's an element of randomness to it as well so we also sample situations that are very different. The approach is as follows:

1. Randomly select an accident record from positive examples
2. Randomly alter one of: the road segment, the hour of the day, or the day of the year.
3. If the new sample isn't with in the accident records, add to the list of negative samples
4. Repeat until we have a large number of negative samples (three times the number of positive samples)

This gives us a training set that is challenging to work with because it's very hard to tell the positive and negative examples apart, which is great for our dataset since it makes sure that the model can focus on the finer differences. This is a commonly used approach for situations such as this. At the end, we get a dataset with 25% accident data points and 75% non-accident data points, with an 'accident binary' column that forms our label for our supervised methods. Numerical data such as speed limit, longitude, latitude were kept as is, but categorical variables such as Route Type, Road Name, Weather, Surface Condition, Light, Traffic Control and so on were one hot encoded. The date and time of the crash posed another challenge, since if we one-hot encoded that, it would take away useful information, so we opted on turning that into variables like day of the year, day of the week, time of the day in minutes and so on.

The non-accident data created through this method can be found here.

Another method involves randomizing the value of each feature for each dummy data point non-uniformly, using expected probabilities that reflect the real world probabilities. For example, if the weather condition feature of our data set had only two possible values, snow or not snow, our first approach will give half of the dummy data a value of snow, while our second approach may only give 5% of the dummy data a value of snow. We also made sure that relationships between features would be kept. For example, "Vehicle Maker" and "Vehicle Model" are tightly connected as Toyota could be matched with Corolla but not with X5. Thus, we made our dummy data as realistic as possible so that our models wouldn't have an easy time telling the real and dummy data apart. In doing so, we thoroughly analyzed patterns in our original data set and gained many valuable insights. One such case where the original data set informed our dummy data was in the crash time. A graph showing the frequencies of crash times is shown below:



Observing the peaks in this graph, we understood that accidents often happen during rush hour times. Thus, we made sure to incorporate higher frequencies of non-accident instances during rush hours reported for Baltimore (the city our data was centered around). More visualizations of our dataset can be found in Appendix A.

The non-accident data created through this method can be found here.

The first supervised model we implemented was Random Forest. The Random Forest algorithm is an ensemble learning technique that combines multiple decision trees to create a more powerful and accurate prediction model. It works by constructing a multitude of decision trees during the training phase and outputting the mode of the classes (classification) or the mean prediction (regression) of the individual trees. The key idea behind the algorithm is that a group of weak learners (decision trees) can be combined to form a strong learner (random forest) that delivers better performance and generalization capabilities.

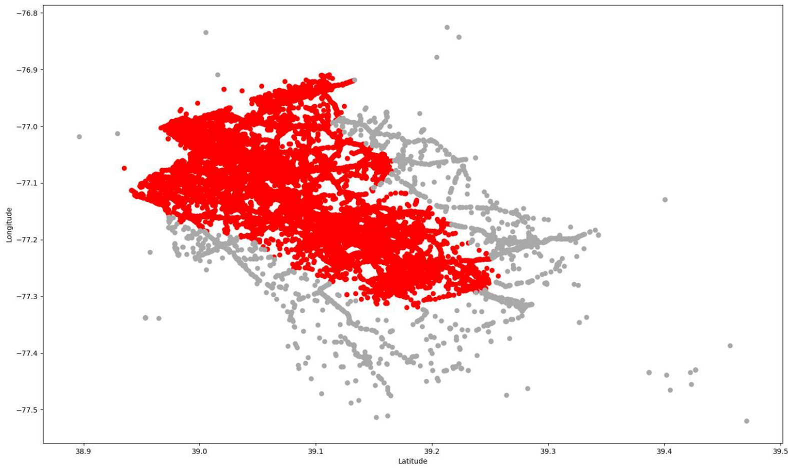
In our initial literature review, we found that most work in this area got higher accuracy when using random forest compared to some other algorithms used. In this particular case, using a Random Forest Classifier is appropriate for several reasons. Firstly, the algorithm can handle both categorical and numerical features, which makes it well-suited for the diverse dataset containing information about vehicle accidents and non-accidents. Secondly, the Random Forest Classifier is known for its robustness against overfitting, as it combines the predictions from multiple trees, reducing the impact of noise and outliers in the data. Lastly, the Random Forest Classifier is capable of handling imbalanced datasets, like the one in question, where there are 25% accidents and 75% non-accident data points. The algorithm can be easily tuned to account for this imbalance, ensuring that both classes are well-represented in the final prediction model. Overall, the Random Forest Classifier offers a flexible and powerful solution for this classification task, providing accurate results while being relatively easy to implement and interpret.

The second supervised model we implemented used gradient-boosted decision trees, specifically the XGBoost algorithm. The XGBoost algorithm is an implementation of gradient boosting that is useful for its speed and accuracy. Wilson (2018) used the same algorithm for their project but with different features. Their model used features such as solar azimuth and population density. The model excels in classification problems which is why we ultimately decided on implementing this model.

The third and final supervised mode we implemented was a 2-layer neural network. The neural network was structured similarly to the neural network built in homeworks, though we ended up using stochastic gradient descent when training the model.

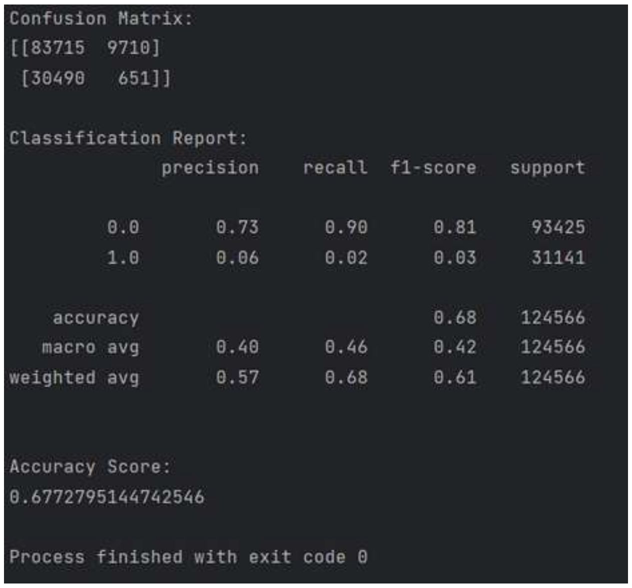
Potential Results and Discussion

The results of the DBscan with an epsilon of 0.03 and a minimum points of 1000 on the latitude and longitude of our data points are shown below:



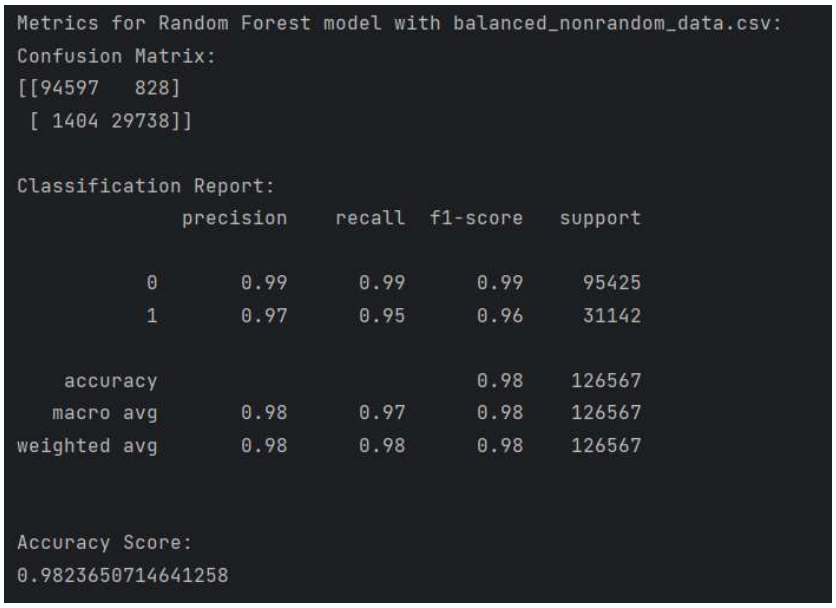
Note that there is only one cluster, shown in red. This makes sense as the cluster should follow the shape of Baltimore, where our data is centered around.

For the random forest algorithm on our balanced data (3:1 accident:nonaccident ratio), we decided to use the 'accident binary' as our label, where 1 corresponded to a positive value and 0 corresponded to a negative value. Once we divided our dataset into labels and features, we further split the dataset into training (80%) and testing (20%) sets, using the train_test_split function. We decided on running the classifier with 100 trees (n_estimators) and a random state of 42 for reproducibility. After training and predictions, we got an accuracy score of 0.6773:



```
Confusion Matrix:
[[83715  9710]
 [30490   651]]

Classification Report:
              precision    recall  f1-score   support

         0.0       0.73      0.90      0.81     93425
         1.0       0.06      0.02      0.03     31141

    accuracy                           0.68    124566
   macro avg       0.40      0.46      0.42    124566
weighted avg       0.57      0.68      0.61    124566


Accuracy Score:
0.6772795144742546

Process finished with exit code 0
```

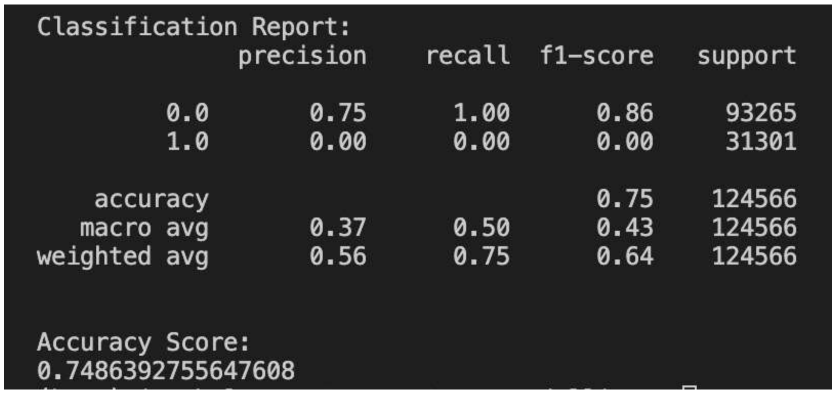
Given how challenging the final dataset was, this accuracy is around the expectations we had, but other metrics such as a recall score of 0.90 for non-accident points and 0.02 for accident points tells us that the model correctly predicts non-accidents often and almost never correctly guesses accidents. Putting these two together it seems that the model does not perform that well at all. However, when we run the same algorithm with the same setup of 100 trees (n_estimators) and a random state of 42 for reproducibility on the non-random dataset, the model performs much better:



```
Metrics for Random Forest model with balanced_nonrandom_data.csv:
Confusion Matrix:
[[94597   828]
 [ 1404 29738]]

Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99     95425
           1       0.97      0.95      0.96     31142

    accuracy                           0.98    126567
   macro avg       0.98      0.97      0.98    126567
weighted avg       0.98      0.98      0.98    126567


Accuracy Score:
0.9823650714641258
```

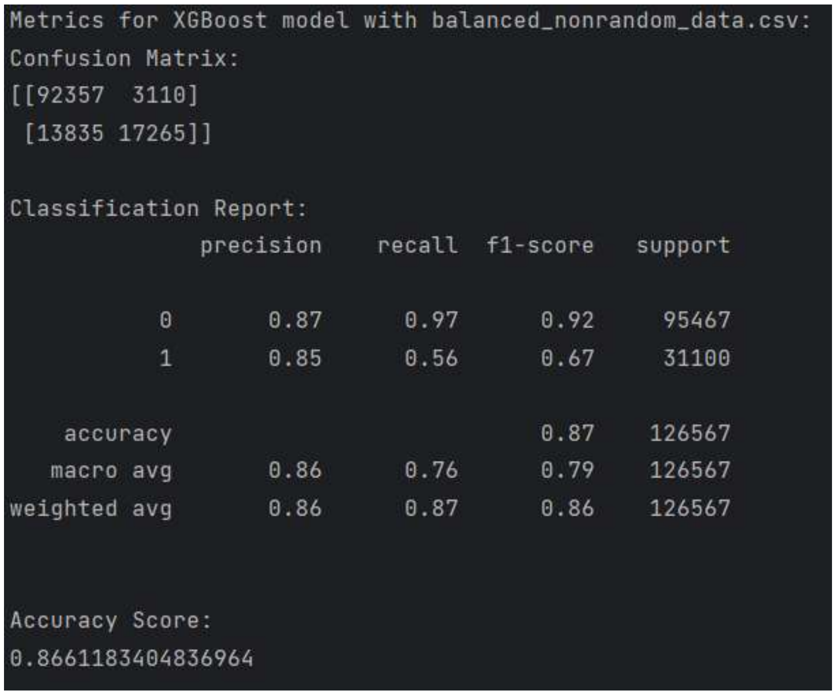
We now get an accuracy of 0.98 with good recall scores of 0.99 and 0.95, which means that our model is correctly able to classify accidents and non accidents most of the time. We found that a similar trend is followed for the other supervised methods as well where the performance is better with the 'nonrandom' dataset.

Running the XGBoost model on our dataset returned an accuracy of 0.71625, a decent score but better metrics to observe would be the recall and precision scores. The model produced a precision of 0.4 and a recall of 0.17224. This means that our model was only able to predict less than 20% of the car accidents and it was only correct about those predictions 40% of the time. These numbers are not great indicating that the model is currently not a good fit for our dataset. To try and improve the performance of the model, PCA was used on the dataset to reduce the dataset down to 5 features. By applying PCA to the data the following performance metrics were achieved.



```
Classification Report:
              precision    recall  f1-score   support

         0.0       0.75      1.00      0.86     93265
         1.0       0.00      0.00      0.00     31301

    accuracy                           0.75    124566
   macro avg       0.37      0.50      0.43    124566
weighted avg       0.56      0.75      0.64    124566


Accuracy Score:
0.7486392755647608
```

Looking strictly at the accuracy of ~75% it seems this model performs well, however, looking at the recall scores for positive and negative samples tells us more about the performance of the model. The recall score of 1.0 for non-accident points and 0.0 for accident points tells us that the model correctly predicts non-accidents 100% of the time and never correctly guesses accidents. Putting these two together it seems that the model predicts that there will not be an accident 100% of the time. This aligns with the accuracy score of 75% as non-accident samples take up 3/4ths of the dataset. There may be multiple reasons for why the model performs so poorly but the most likely reason would be that the dataset was constructed poorly. Reducing the number of non-accidents would possibly help the model perform better and not overfit to the possibility of there not being an accident. The way non-accident samples were created also could have been modified to create a more realistic dataset with more variation in the data. Finally, running the XGBoost model on a dataset with the dataset that is nonrandom produced the following results.



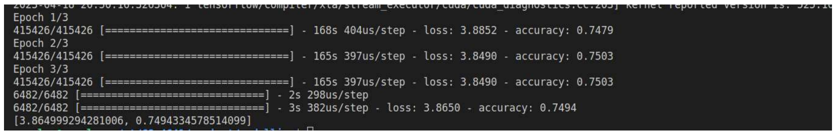
```
Metrics for XGBoost model with balanced_nonrandom_data.csv:
Confusion Matrix:
[[92357  3110]
 [13835 17265]]

Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.97      0.92     95467
           1       0.85      0.56      0.67     31100

    accuracy                           0.87    126567
   macro avg       0.86      0.76      0.79    126567
weighted avg       0.86      0.87      0.86    126567


Accuracy Score:
0.8661183404836964
```

The model performs significantly better on this dataset as the performance metrics seem to be more realistic across the board, even producing a higher accuracy. The reason why it may have performed better is that this nonrandom dataset may be more representative and accurate for real learning. However, the model may be overfitting to the dataset due to how the non-accident samples were created.

The final supervised learning model we implemented was a neural network. The neural network we created had two layers. The first layer had a number of perceptrons equal to the number of features in the dataset and had a tanh activation function. The final layer had one perceptron which had a ReLU activation function. After trying different optimizers and epochs, we found that the model generally performed best with stochastic gradient descent and tended to converge after 3 epochs.

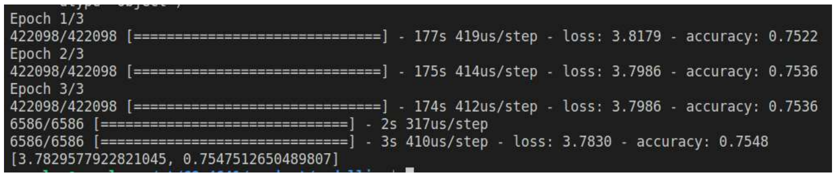
We trained the neural net using the dataset with nonrandom dummy non-accidents, the dataset with random dummy non-accidents, and a PCA of the dataset with random dummy non-accidents (which maintained 99% of the variance). Surprisingly, the neural network performed significantly worse when trained with data that had been run through a PCA (about 50% accuracy). It is possible that the principal components for the non-accident data were similar to that of the accident data, but this would seem unlikely as the dummy non-accident data was created with randomized values.

When the neural net was trained of the dataset with random dummy non-accidents, it performed with about 75% accuracy, as shown below:



```
Epoch 1/3
415426/415426 [==============================] - 168s 404us/step - loss: 3.8852 - accuracy: 0.7479
Epoch 2/3
415426/415426 [==============================] - 165s 397us/step - loss: 3.8490 - accuracy: 0.7503
Epoch 3/3
415426/415426 [==============================] - 165s 397us/step - loss: 3.8490 - accuracy: 0.7503
6482/6482 [==============================] - 2s 298us/step
6482/6482 [==============================] - 3s 382us/step - loss: 3.8650 - accuracy: 0.7494
[3.864999294281006, 0.7494334578514099]
```

The neural net performed similarly with the nonrandom dummy non-accident dataset, again having about 75% accuracy, as shown below:



```
Epoch 1/3
422098/422098 [==============================] - 177s 419us/step - loss: 3.8179 - accuracy: 0.7522
Epoch 2/3
422098/422098 [==============================] - 175s 414us/step - loss: 3.7986 - accuracy: 0.7536
Epoch 3/3
422098/422098 [==============================] - 174s 412us/step - loss: 3.7986 - accuracy: 0.7536
6586/6586 [==============================] - 2s 317us/step
6586/6586 [==============================] - 3s 410us/step - loss: 3.7830 - accuracy: 0.7548
[3.7829577922821045, 0.7547512650489807]
```

Conclusions

Through the implementation of this project, our group was able to work with a lot of the models taught in class on a real-world dataset. Observing the various accuracies and metrics of our differing models allowed us to create a greater understanding of the underlying concepts of machine learning. By collaborating as a group to clean the data, process the data, and run the models, we learned the difficulties of dealing with unprocessed data and learned how to use multiple machine learning tools such as PCA, random forest, XGBoost, and neural networks.

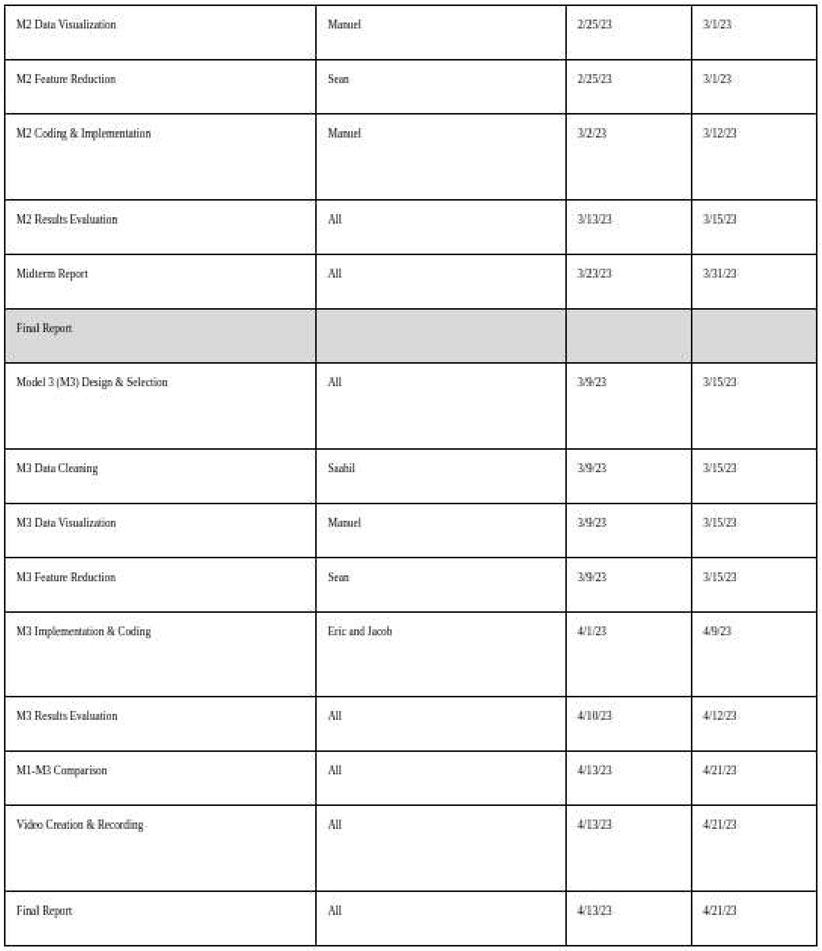
While we were able to run models that produced varying results, we believe that there are multiples ways to improve upon our project in the future:

- Improve upon our non-accidental data sets. As we saw from our models, using our randomized non-accidental data sets was not very good in distinguishing between non-accidents and accidents. Since randomized data sets are prominently used in literature, iterating upon our randomization process could lead to better results.
- Expand beyond Maryland data. Since our model uses data strictly from the Baltimore, Maryland area, our models could have overfit to features specific to Baltimore. Using our models on different data sets would allow us to make sure this overfitting isn't happening.
- Use different models. This is an obvious one. More models can always be used.

We are glad we were able to make progress in the problem of predicting when accidents are likely to occur, and hope that this work helps future projects in predicting and preventing accidents.

Proposed Timeline

TASK TITLE	TASK OWNER	START DATE	DUE DATE
Project Team Composition	All	1/17/23	2/3/23
Project Proposal			
Introduction & Background	Saahil	2/9/23	2/20/23
Problem Definition	Saahil	2/9/23	2/20/23
Methods	Manuel	2/9/23	2/21/23
Potential Dataset	All	2/9/23	2/20/23
Potential Results & Discussion	Sean	2/9/23	2/20/23
Video Creation & Recording	Jacob and Eric	2/17/23	2/24/23
GitHub Page	Manuel	2/17/23	2/24/23
Midterm Report			
Model 1 (M1) Design & Selection	All	2/25/23	3/2/23
M1 Data Cleaning	Saahil	2/25/23	3/2/23
M1 Data Visualization	All	2/25/23	3/2/23
M1 Feature Reduction	Sean	2/25/23	3/2/23
M1 Implementation & Coding	Eric and Jacob	2/25/23	3/12/23
M1 Results Evaluation	All	3/13/23	3/15/23
Model 2 (M2) Design & Selection	All	2/25/23	3/1/23
M2 Data Cleaning	Saahil	2/25/23	3/1/23



M2 Data Visualization	Manuel	2/25/23	3/1/23
M2 Feature Reduction	Sean	2/25/23	3/1/23
M2 Coding & Implementation	Manuel	3/2/23	3/12/23
M2 Results Evaluation	All	3/13/23	3/15/23
Midterm Report	All	3/23/23	3/31/23
Final Report			
Model 3 (M3) Design & Selection	All	3/9/23	3/15/23
M3 Data Cleaning	Saahil	3/9/23	3/15/23
M3 Data Visualization	Manuel	3/9/23	3/15/23
M3 Feature Reduction	Sean	3/9/23	3/15/23
M3 Implementation & Coding	Eric and Jacob	4/1/23	4/9/23
M3 Results Evaluation	All	4/10/23	4/12/23
M1-M3 Comparison	All	4/13/23	4/21/23
Video Creation & Recording	All	4/13/23	4/21/23
Final Report	All	4/13/23	4/21/23

Contribution table

Member	Contribution
Eric Lim	Cleaning and formatting the data set and helping with PCA. Created final presentation video.
Sean Jung	Creation of non-random dummy data, running PCA for feature reduction, and writing the final report.
Jacob Yu	Creation of random non-accident data and XGBoost model.
Manuel Roglin	GitHub page, DBSCAN, results and discussion, training of the neural network model, and creation of the final presentation video.
Saahil Sanganeria	Creation and formatting of random non accident data and running random forest.

Dataset

https://catalog.data.gov/dataset/crash-reporting-drivers-data

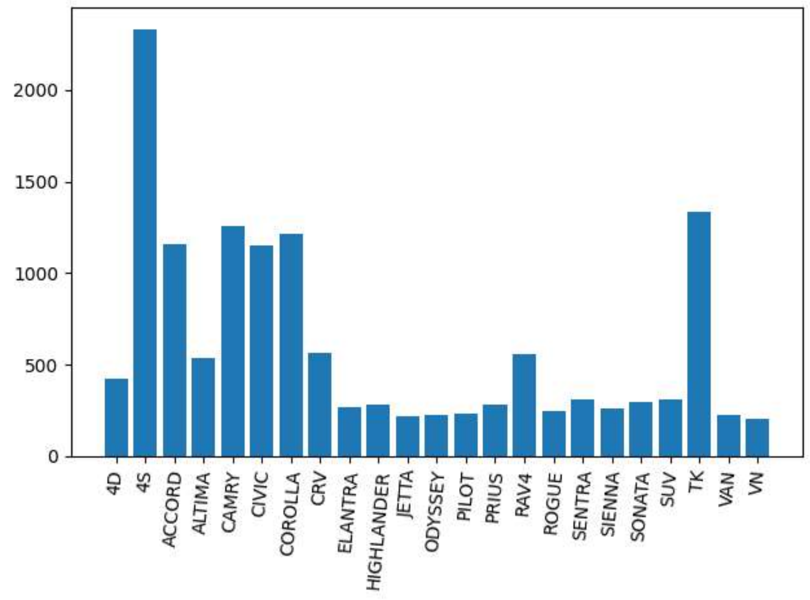
References

World Health Organisation. Global status report on road safety 2018; 17th June 2018; Available at https://www.who.int/publications/i/item/9789241565684
Centers for Disease Control and Prevention. WISQARS (Web-based Injury Statistics Query and Reporting System) [online]; 2020. Available at https://www.cdc.gov/injury/wisqars/index.html
Daniel Wilson. Using Machine Learning to Predict Car Accident Risk; 3rd May 2018; Available at https://medium.com/geoai/using-machine-learning-to-predict-car-accident-risk-4d92c91a7d57
Cigdem, A., and Cevher Ozden. "Predicting the severity of motor vehicle accident injuries in Adana-Turkey using machine learning methods and detailed meteorological data." Int. J. Intell. Syst. Appl. Eng 6.1 (2018): 72-79. Available at https://www.researchgate.net/profile/Cigdem-Aci/publication/324101053_Predicting_the_Severity_of_Motor_Vehicle_Accident_Injuries_in_Adana-Turkey_Using_Machine_Learning_Methods_and_Detailed_Meteorological_Data/links/604a024345851543166bac22/Predicting-

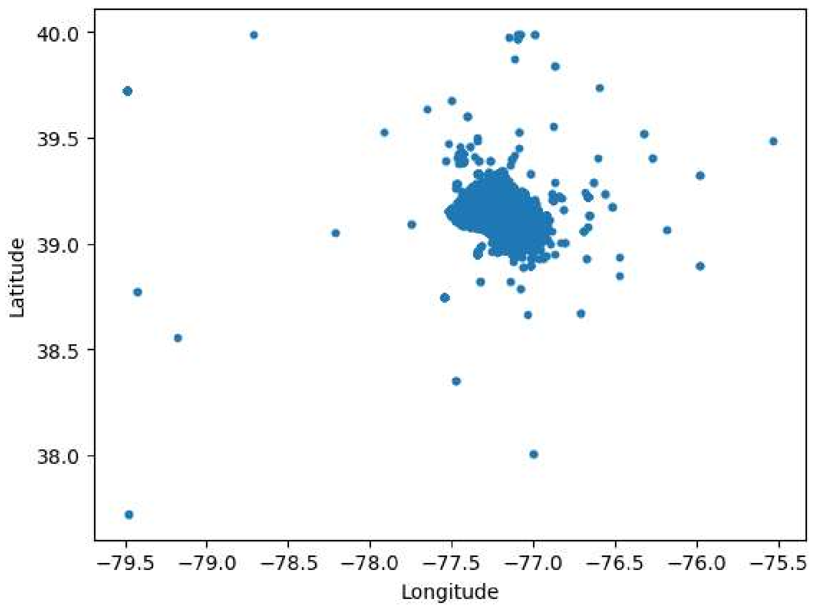
the-Severity-of-Motor-Vehicle-Accident-Injuries-in-Adana-Turkey-Using-Machine-Learning-Methods-and-Detailed-Meteorological-Data.pdf?_sg%5B0%5D=started_experiment_milestone&origin=journalDetail&_rtd=e30%3D
Al-Mistarehi, B. W., et al. "Using machine learning models to forecast severity level of traffic crashes by r studio and arcgis. front." Machine Learning Applications in Civil Engineering 16648714 (2022): 31. Available at https://www.researchgate.net/profile/Rana-Imam/publication/360065159_Using_Machine_Learning_Models_to_Forecast_Severity_Level_of_Traffic_Crashes_by_R_Studio_and_ArcGIS/links/62
Machine-Learning-Models-to-Forecast-Severity-Level-of-Traffic-Crashes-by-R-Studio-and-ArcGIS.pdf

Appendix A: Visualizations for the Dataset

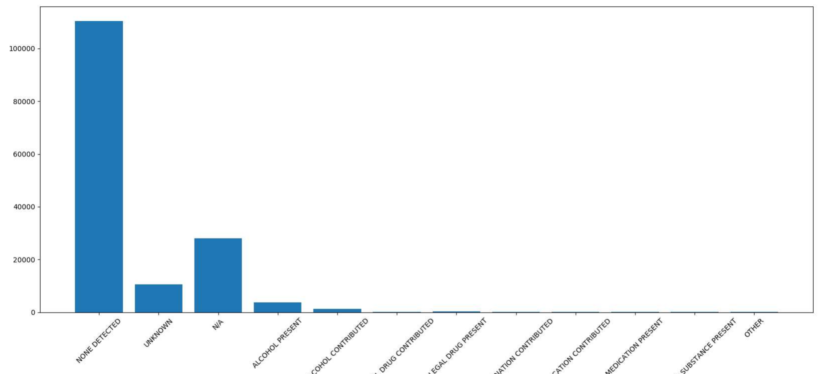
Number of Injuries based on car model:



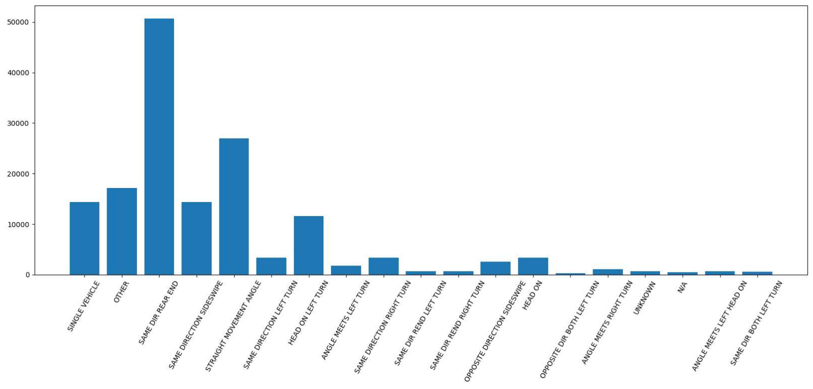
Latitude and Longitude of the Crashes:



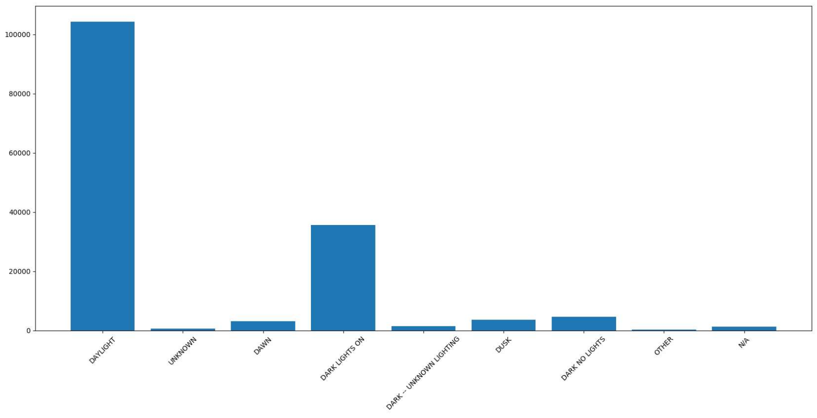
Frequencies of Driver substance abuse:



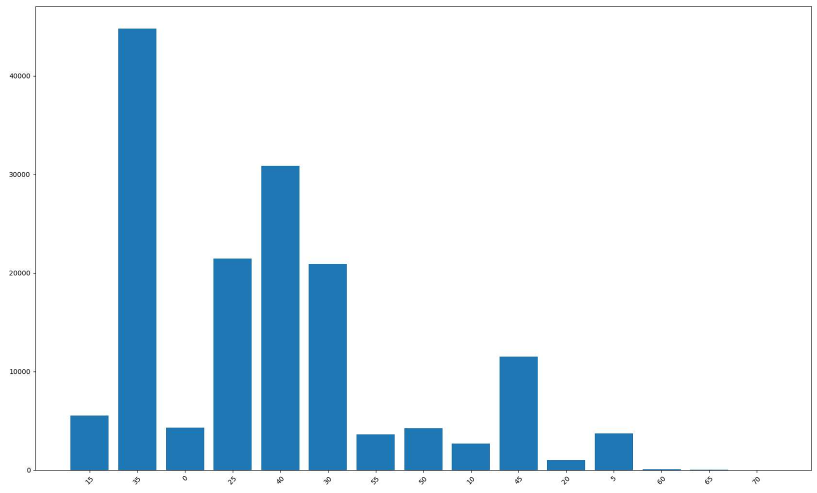
Frequencies of Collision types:



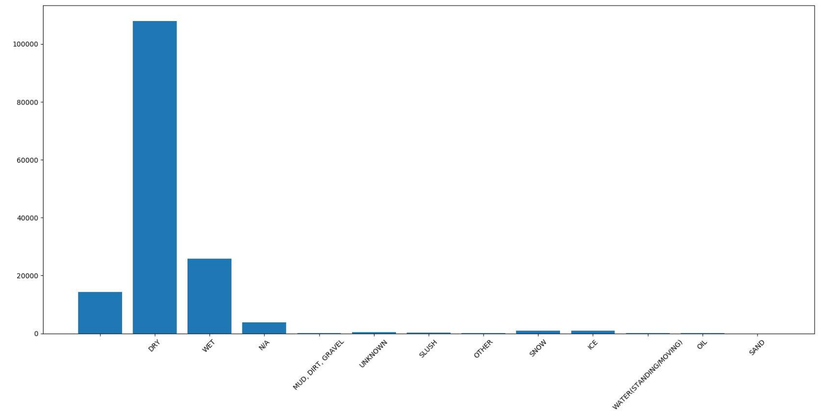
Frequencies of present light:



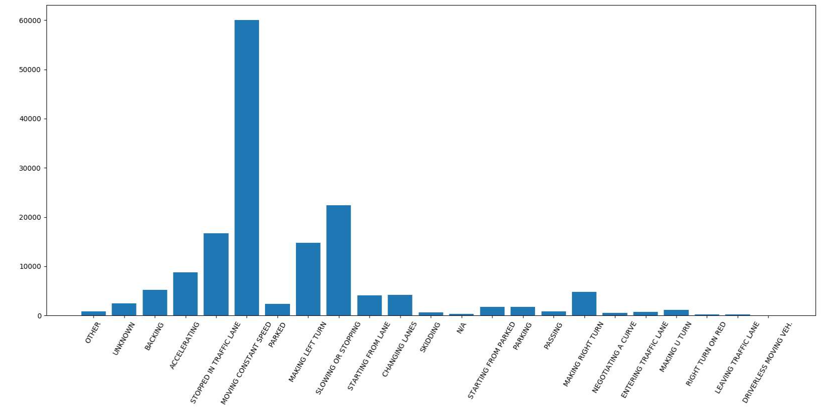
Frequencies of Speed limit:



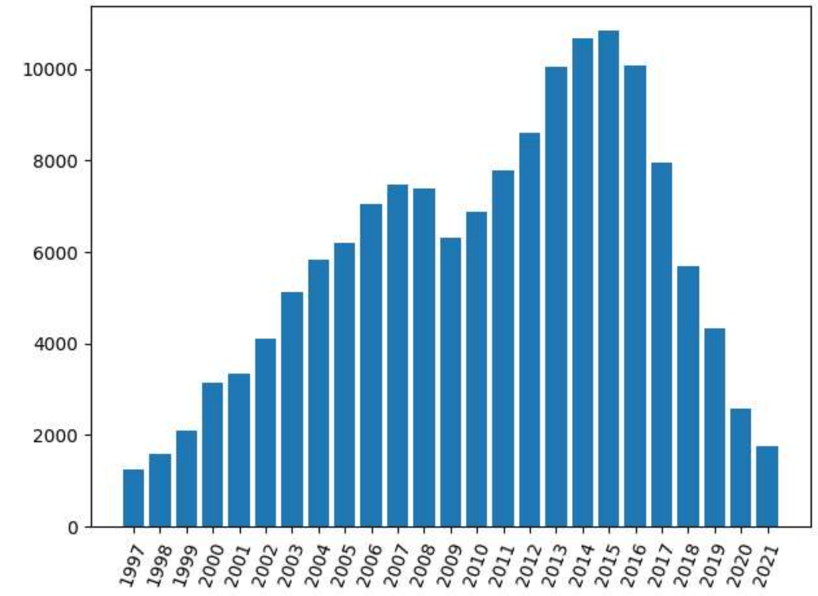
Frequencies of Surface Condition:



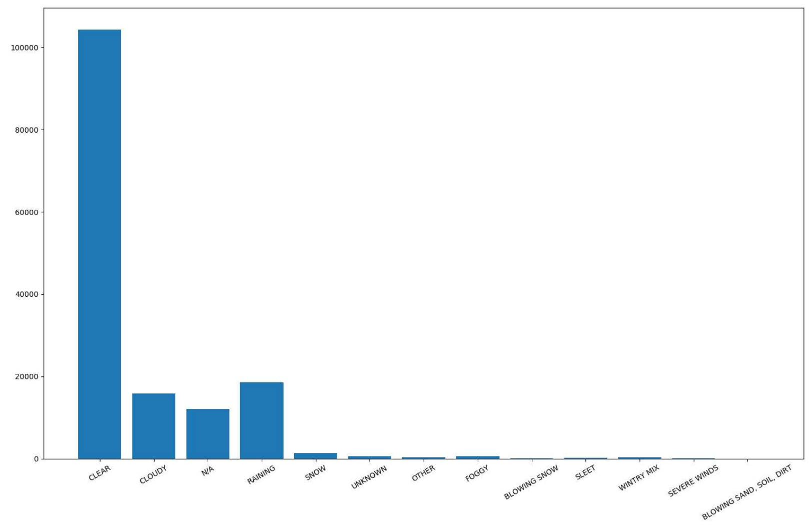
Frequencies of Vehicle Movement:



Frequencies of Vehicle Year:



Frequencies of Weather conditions:


100000
80000
60000
40000
20000
0
CLEAR
CLOUDY
N/A
RAINING
SNOW
UNKNOWN
OTHER
FOGGY
BLOWING SNOW
SLEET
WINTRY MIX
SEVERE WINDS
BLOWING SAND, SOIL, DIRT