**Injury Severity Prediction**

**Abstract**

This project aims to better prepare first responders by providing them with an accurate prediction of the injuries that have occured at the site of a car accident. This paper describes the workflow of data cleaning, data engineering, model building, model training and model optimization that ultimately led to a neural network with poor accuracy.

**Introduction**

A primary flaw in today's first responder structure is that the only source of information as to the extent of the emergency comes from those there at the time of the emergency. People tend to over or underreact in high pressure situations which could lead to exaggeration or understatement of the severity of the damages incurred during a car accident. If first responders have accurate information about the most likely injuries that occured before arriving at an accident they can be better prepared to respond in the most efficient manner to any injuries on the scene. Given certain characteristics about an accident such as the road conditions, the speed limit, the types of cars, and the area in which the car was hit we can predict the severity of injury in a car accident. At the moment, there is no implementation of any system of this kind that we could find. Some first responders have sets of questions they ask those calling in an emergency to try to get a better understanding of the accident, however, no predictive model is in use.

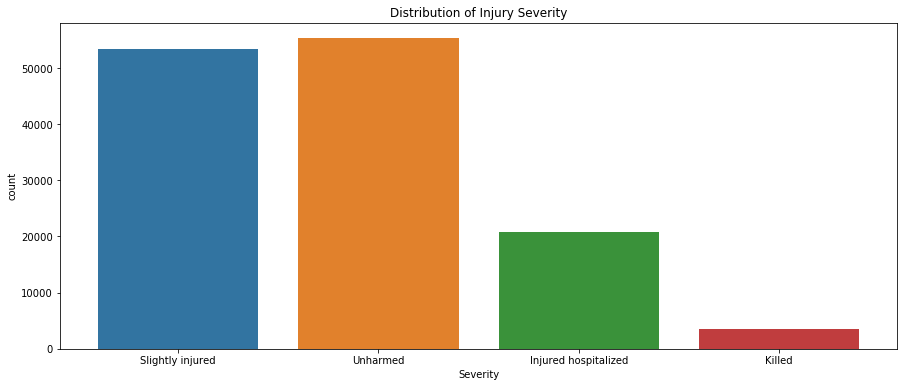
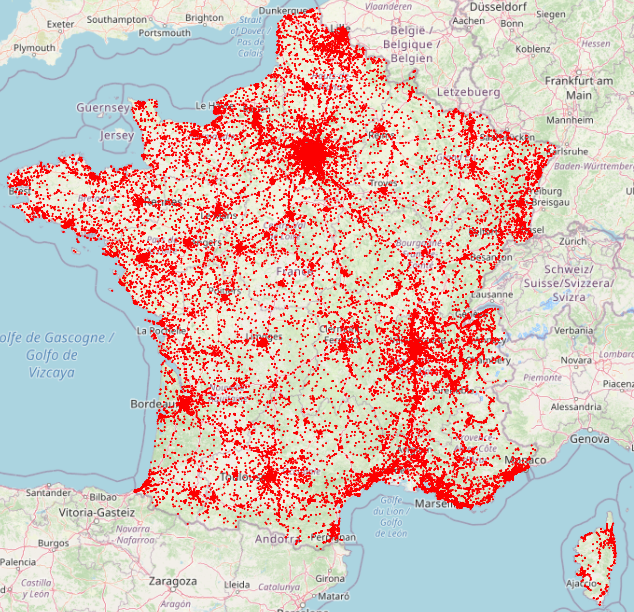
**Data**

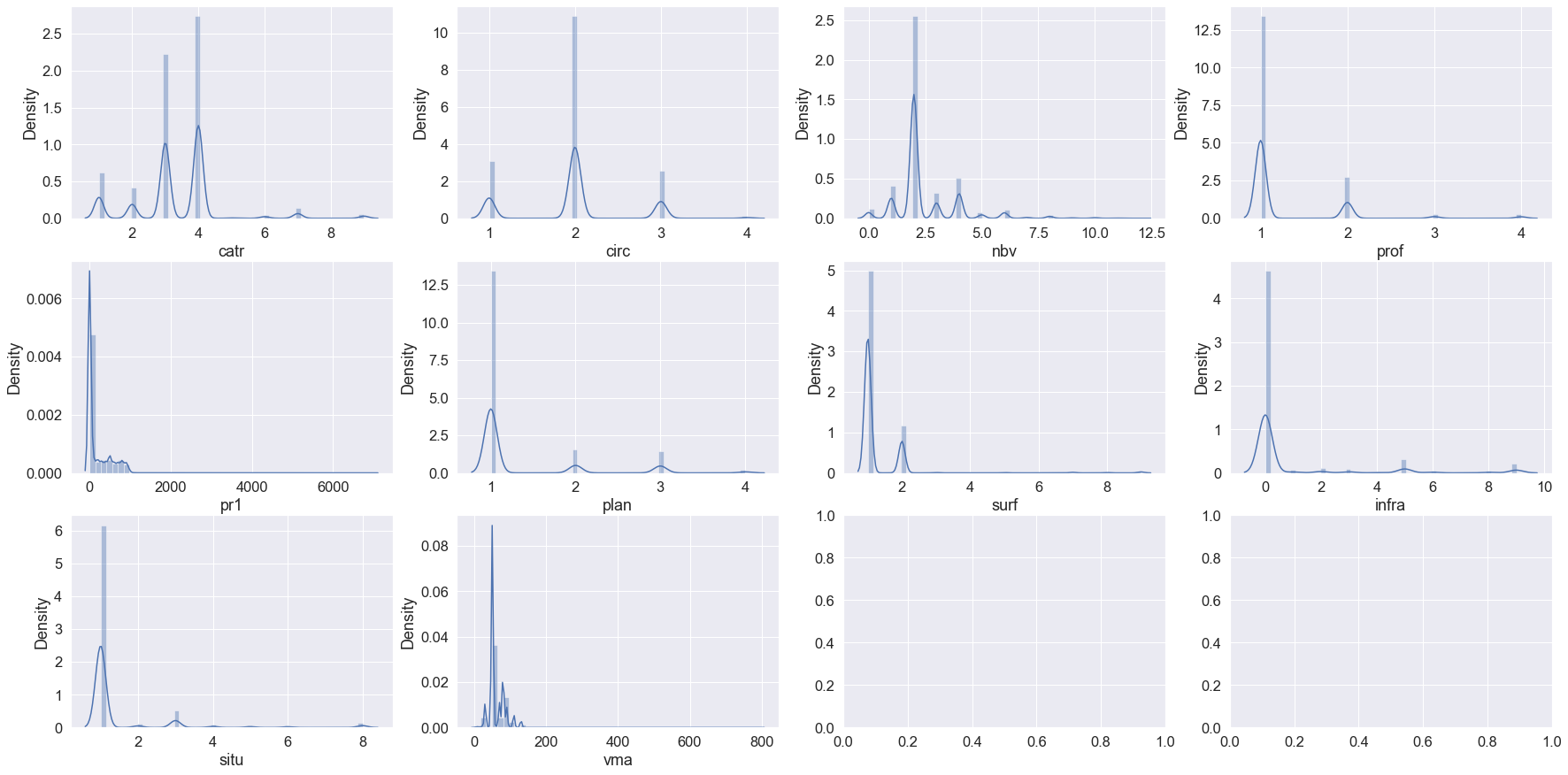
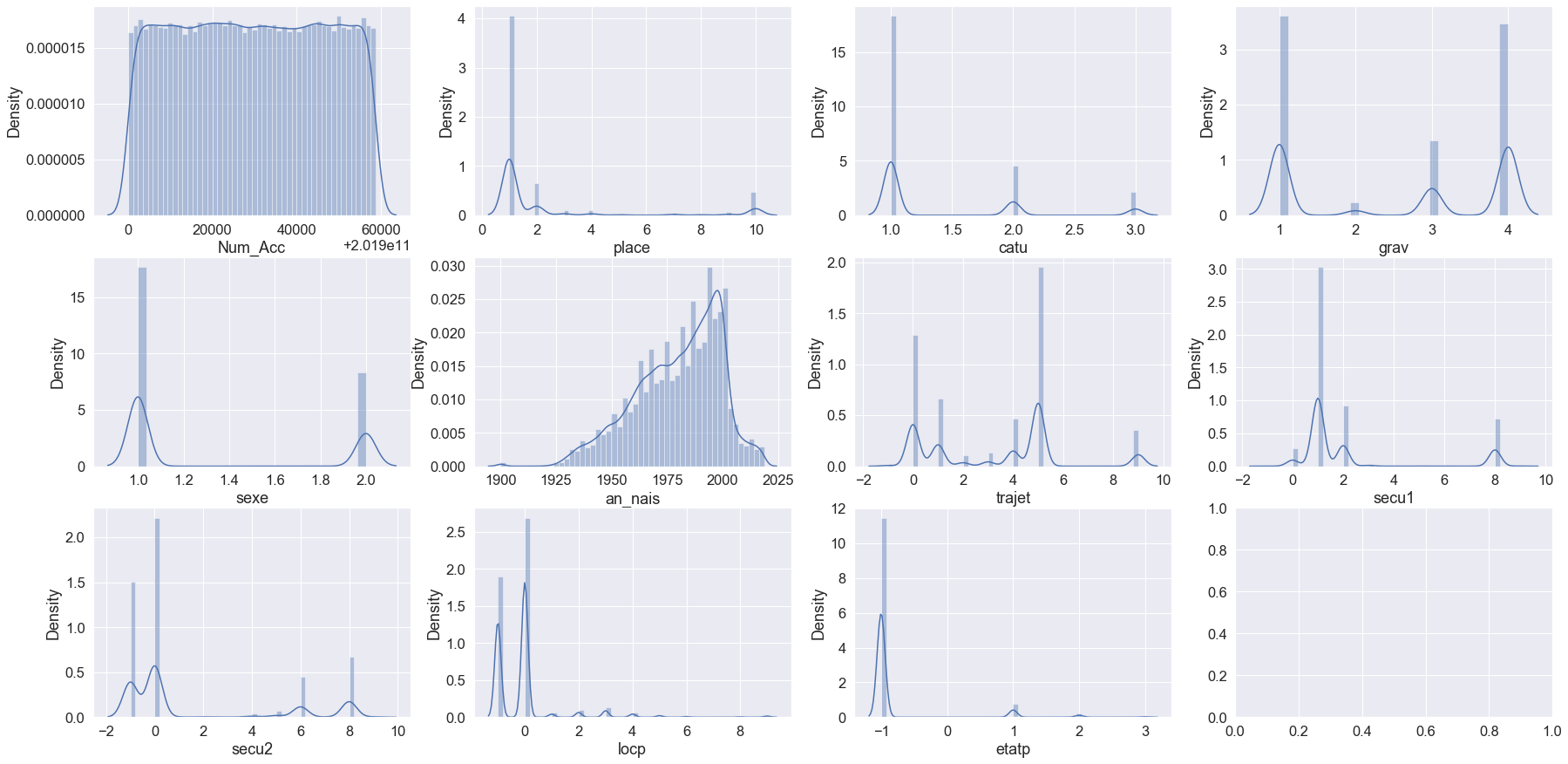
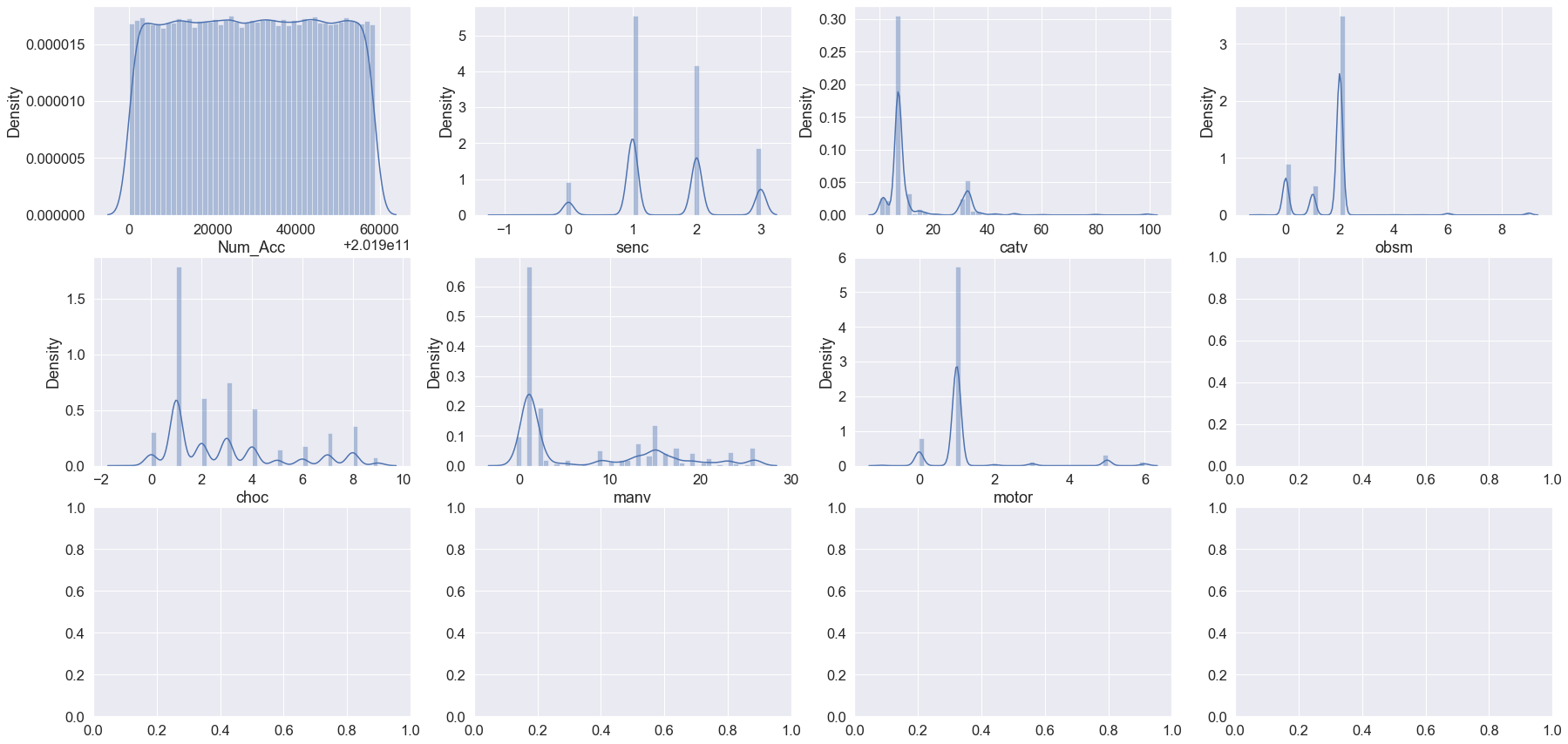
Our data was found on Kaggle.com and consisted of four csv files which contain information from the Traffic Body Accident Analysis Bulletins (TBAA) about bodily injuries that occured in traffic accidents in mainland France as well as its departments and territories in the year 2019. A summary of the four files is provided below:

1. Characteristics - Describes the general characteristics of the accident such as the time of day and the department in which the accident occured.
2. Places- Describes the characteristics of the location of the accident such as the presence of a median, stop sign, or curvature in the road.
3. Vehicles - Contains information about the vehicles involved in the accident such as the category of vehicle and the number of people in the vehicle.
4. Drivers - Contains information about the people involved in the accident such as the use of a seatbelt or and whether the person was driving the vehicle or a passenger.

The files and their columns are labeled in French which presents a slightly different type of challenge while handling the data.

**Exploratory Data Analysis**

We began by looking at the number of missing variables in each data frame and found that there were few columns missing data, however, of those few there were some which were missing over 50% of their values. These columns were dropped. We also visualized the distribution of our target variable, injury severity in our data (shown below). Thankfully, we see that a relatively low percentage of those involved in traffic accidents are killed. However, we may need to balance the distribution of our target variable to produce more accurate results from our models. We also visualized the count of accidents by department in our data. This showed us that about ten of the departments make up over half of the traffic accidents. This was no surprise as there are a few departments with large, densely populated cities as can be seen in the plot to the right which shows the location of each accident in our data.

Next we visualized the number of accidents by month of the year, this did not provide any real insight that would help predict the severity of an injury, simply showing us that the summer months, June and July had the highest number of accidents. We also compared the types of injury between drivers, passengers, and pedestrians involved in accidents and found that pedestrians are much more likely to be killed and much less likely to be unharmed than the two other groups. 

We then plotted the frequency distributions of all categorical columns for each of the three data frames as seen above. We finally viewed the summary statistics for the rest of the data.

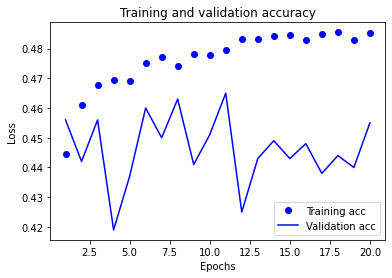
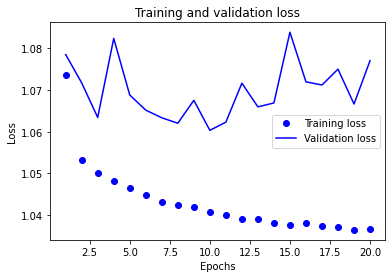
From our exploratory data analysis we found that the longitude and latitude needed to be adjusted by a factor of 10,000,000 as well as some columns containing values that made no logical sense, were typos, or were meant to be in different columns. We also observed that most of our columns contained nominal categorical variables which would need to be turned into dummy variables. Some of our columns also contained 50+ categorical variables so we knew the data to be used for modeling was going to be very wide. Throughout our exploratory data analysis we also found about 20 of our columns to not be relevant to our problem and dropped them from the data frame before any preprocessing was done.

**Modeling**

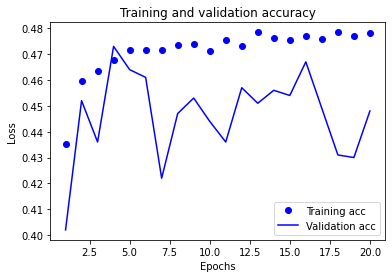
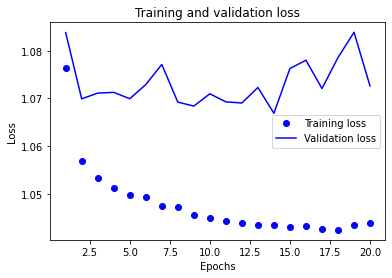
Prior to building our neural network model we had to ensure our data to the correct format to be passed into our model. We first made dummy variables of the categorical variables we identified as having predictive ability in regards to our question. Next, we converted any float variables into type int before doing our train test split, creating a training set with 67% of the data and a test set with the remaining 33% of the data. The x train and test sets just made were then turned into numpy arrays so that they could be vectorized. We also made the y train and test sets into encoded categorical labels.

For our neural network model we started with two hidden layers with 64 hidden units and a rectified linear unit (relu) activation function. The output layer had five hidden units and a sigmoid activation function which would provide us with probabilities for each prediction made by the model. We user rmsprop for our optimizer, categorical cross entropy for our loss function, and accuracy as our metric. We trained our model over 20 epochs and used a batch size of 32. We evaluated the model on the training data as well as the validation data.

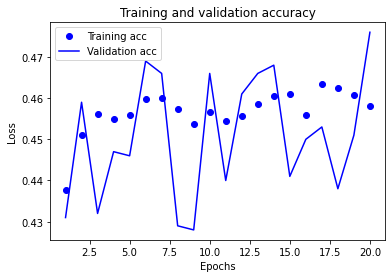
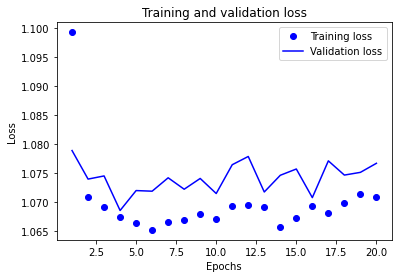
In this model, we were not able to receive the results we would like. We got a training loss value of 1.0366 and validation loss of 1.077. Our final training accuracy was only 0.4851, and the final validation loss came out to 0.455. Using these values over the course of the 20 epochs, we were able to generate plots. From this it is clear that our test results on the training data were very erratic, while the validation tests received steadily changing results.



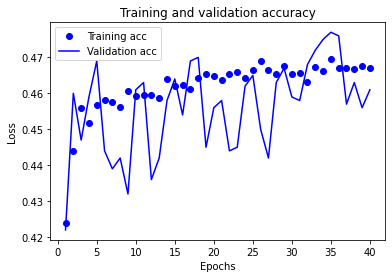
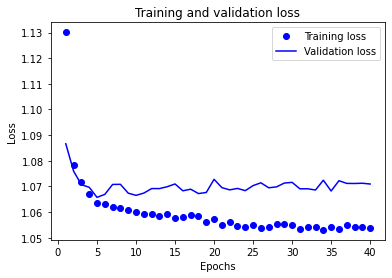
Next, we tried optimizing our model by adding a third hidden layer but this yielded very similar results and prompted us to add a fourth layer which still yielded very similar results. Our training loss came out to 1.0439 and validation loss came out to 1.0726. We got training accuracy of 0.4782 and validation accuracy of 0.448. We once again generated plots for the model evaluations. Again, we found that the results of the training tests were very consistent, while the tests on the validation data were all over the place.



Next we made a third model with dropout regularization as we felt our models may have been using parameter values that didn’t have a large impact on our model’s predictions. In our dropout model we achieved training loss of 1.0709 and val loss of 1.0767. For accuracy, we got training accuracy of 0.458 and validation accuracy of 0.476. We generated more plots for this model and we did find again that the training test results were much more all over the place, but this time it was much closer to the validation results.



As another attempt to improve the model, we decided to change the batch size as well as the number of epochs. We changed our batch size to 128 and doubled the number of epochs to 40. This did not yield better results for us. We got a training loss of 1.0539 and validation loss of 1.0709. Our training accuracy was 0.4671 and our validation accuracy was 0.4610. In this round of plots, we found similar results as before, except that our training and validation loss plots did turn out fairly similar.



**Conclusion**

Our models did not achieve a high enough accuracy to be applicable in any real world setting, especially one potentially dealing with life threatening situations. Further engineering can certainly be done to our data to improve our results, and we can also test other model types on our data. A random forest intuitively seems like it would yield the best results. We struggled with overfitting as is common with any machine learning model. Perhaps lowering the amount of noise or passing a dataset that is less wide could improve our results.