**Introduction**

* We would like to create a predictive model for accident severity based on certain features obtained from a dataset on accident history. This could be used within current GPS technology such as Google Maps, to provide users with a safer route.
* Thus, our audience will be a client like Google, hoping to offer a “Safer Route” feature in their GPS application.

**The Data Set and how it will be used**

* I will use the example set provided which is the accident data for Seattle City. My intention is to create a model that makes predictions solely on conditions outside of the drivers’ control. This means it will consider strictly environmental factors and eliminate biases such as speeding. Thus, I will use each environmental factor included in the dataset.

Predictive Features Used:

ADDRTYPE, COLLISIONTYPE, VEHCOUNT, WEATHER, ROADCOND, LIGHTCOND, SPEEDING (only included so we can drop data points that include speeding)

Target Feature:

SEVERITYCODE

**Methodology**

1. Remove Bias (speeding)

* Created a new data set with my selected features. Eliminated rows in which the driver was speeding in order to focus on solely external influences.

1. Standardize certain Features
2. LIGHTCOND

* Determine different categorical variables using value\_counts()
* Make everything Dark or Light, remove rows where this is unknown.
* Dusk, Dawn, Dark- Street Lights On were all considered Light
* Dark- Street lights off, Dark- no street lights obviously considered Dark

1. ROADCOND

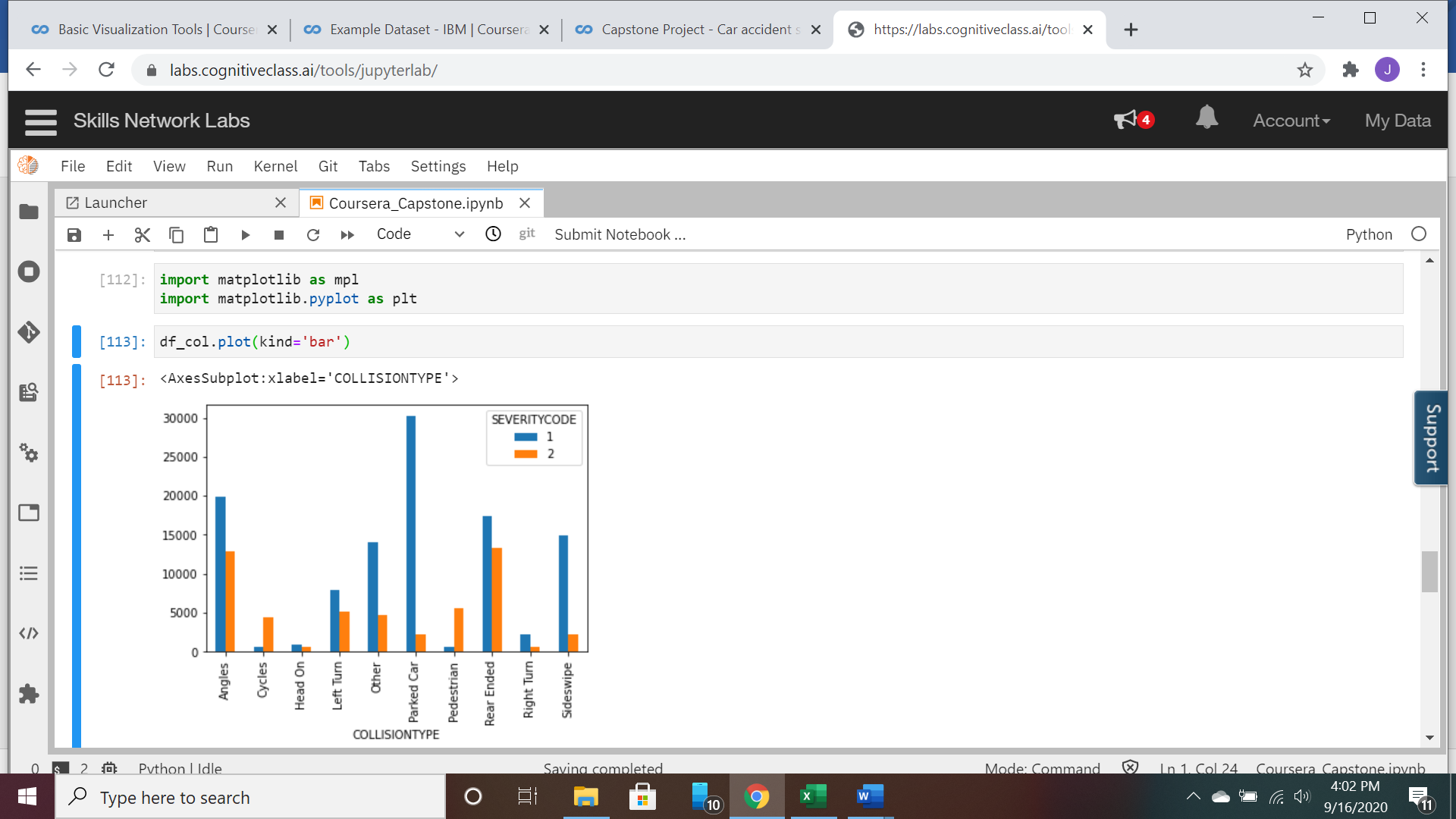
* Make everything Dry, Wet, Snow/Slush, Ice
* All other categories removed from feature, as they comprised a very small portion of the data and could be accurately represented among the categories above

1. WEATHER

* Make everything Clear, Not Clear
* Rather straight forward, any condition potentially affecting visibility was converted to Not Clear while those not impacting visibility (ie. Overcast, Partly Cloudy etc.) converted to Clear

1. COLLISIONTYPE

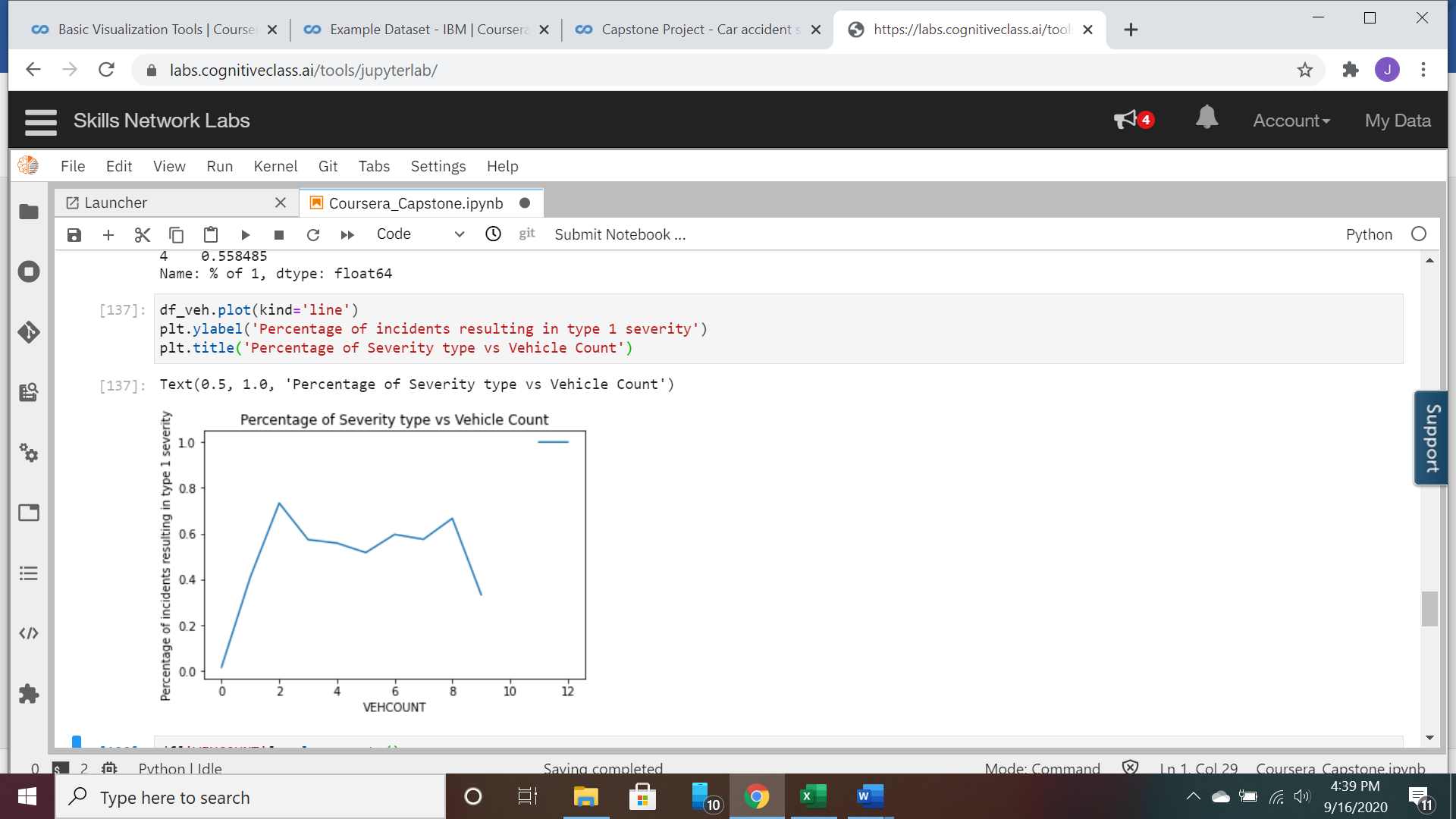
* For this feature, I performed some exploratory analysis in order to combine some categories if they had likely severity outcomes. This would help yield a more balanced dataset. I did so by creating a barchart where we could easily see COLLISIONTYPE’s that skewed towards a certain severity



* As indicated above, “Cycles” and “Pedestrian” incidents led to a severity level of 2 drastically more often than 1, thus they will be combined as “High” representative of yielding a high severity accident
* “Sideswipe”, “Right Turn”, “Parked Car” and “Other” likely end up with a severity of 1, thus we will combine them as “Low” representing low severity cases.
* “Angles “, “Head On”, “Left Turn”, “Rear Ended” are the more ambiguous cases and we will thus combine them as “Medium”

1. VEHCOUNT

* Since, VEHCOUNT was a feature I was on the fence about including due to its unpredictable nature, I performed some exploratory analysis on whether it was worth including in my data set. By “worth including”, I mean wether it had a significant impact on the severity outcome.



* As, you can see above there is no trend whatsoever, between the number of vehicles involved and the severity of the incident. Thus, we will disclude it from the data set.

1. Preparing the data for each Machine Learning Model

* The Binary columns LIGHTCOND and WEATHER can be integer-encoded as 0 and 1’s
* COLLISIONTYPE will be integer-encoded as 1,2,3 since it is ordinal in nature.
* The rest of the features will need to be one-hot encoded and then have the original columns dropped

1. KNN

* This ML model ran very slowly. I was not able to optimize for K because the ‘for loop’ would have ran for hours. Instead, I tested the model with K=1 and proceeded to the next model
* I used Scikit Learn’s built in accuracy score for determining the quality of the model. In binary classification, it is equal to the Jaccard similarity score

1. Logistic Regression

* In order to run this model, I only had to create new arrays for my y\_test/train, that converted the SEVERITYCODE from 1’s and 2’s, to 0’s and 1’s in order to match the outputs from the Logistic Regression model.

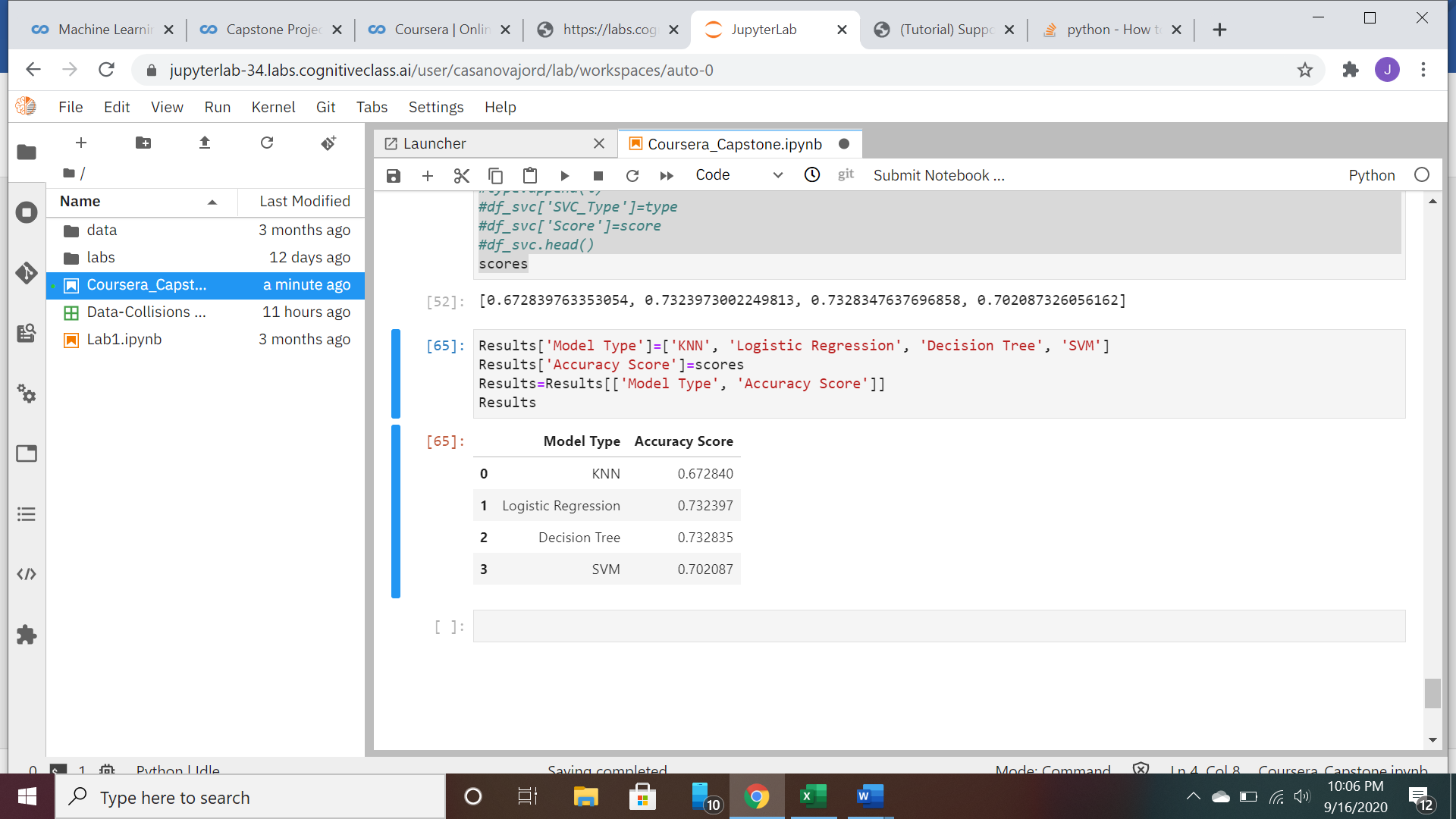
1. Decision Tree

* All I did here was take my original, cleaned and wrangled data set, and integer encoded every feature. The decision tree converged rather quickly with only 2 levels necessary.

1. Support Vector Machine

* SVM could receive the exact same test and train data set that KNN used.
* This model was also very computationally expensive. I only ran it for the linear kernel case as any additional kernels would have taken substantial computing time.

**Results**

****

**Discussion**

Unfortunately, I was unable to determine the optimum model with certainty. Both K nearest neighbor and Support Vector Machine were too computationally expensive for my PC. While I could run the models with the simplest parameters, I was unable to perform parameter tuning to optimize results. With that being said, the Decision Tree model yielded the highest accuracy. It is also useful to note that this model converged to the optimum solution extremely quickly. I would speculate that this has something to do with small number of features used. It is likely that SVM model would have outperformed the Decision tree if I was able to run it for each kernel and choose the optimal, but again, due to computational expense it was not possible. The same could be said for KNN.

**Conclusion**

Concluding this report, I can say with confidence we have a reasonably good model to predict accident severity based on external conditions. In order to incorporate this model into a piece of GPS technology, the model would need an element that maps it to a location, and each feature would need to be input at that specific location. The COLLISIONTYPE would likely have to be estimated based on the probable collision at a given location. With this information, probabilities and severities of accidents could be predicted at given locations.