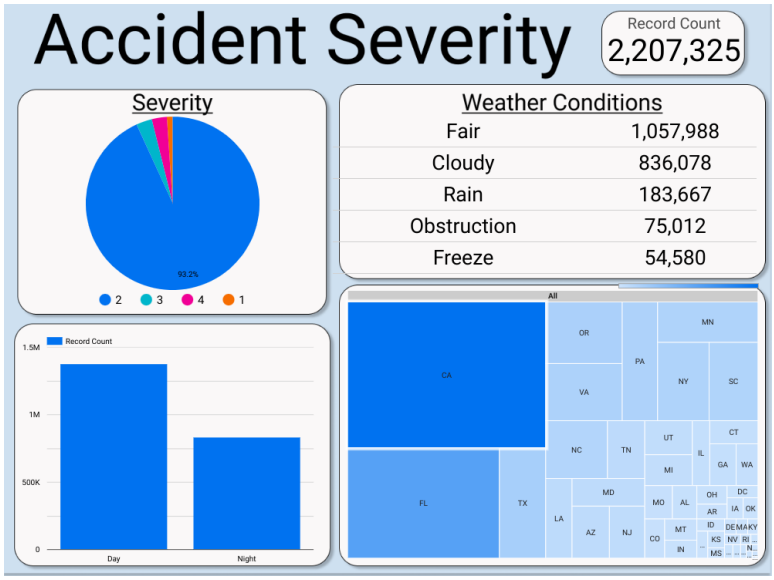
Accident Severity

Credit: Alexas_Fotos on Pixabay

Introduction:
Accidents happen when you least expect them, but wouldn't it be interesting to predict the accident's severity before it occurs? Our team used a dataset from Kaggle that used multiple traffic API systems in 49 states to see if we could build models to predict the severity. For our project, we used Google Cloud Computing to create dashboards, run an AutoML model, and used InterpretML to run models. We also used Sklearn gradient boosting in Google Colab to run our own model for comparison.

Data exploration
Originally, our dataset was 1.15 GB, but after cleaning our dataset is 820MB. In our dataset, the severity level of accidents ranged from 1-4, with 1 being the lowest and 4 being the highest. This is based on how long traffic was delayed after an accident. Our columns included location, a description of the event, weather, and road feature information. In Google Cloud, you can use BigQuery to use as a data warehouse to query and store your data. Google also has a platform called Data Studio where you can create easy-to-read visualizations. Our team created a dashboard to analyze our data.



There were initially 2.8 million accidents in our dataset, which was reduced to 2.2 million after data cleaning, which will be described in more detail later. With the graphs in this dashboard, we can see that 93% of the accidents were severity 2, so the majority of the data represents just this severity level. After simplifying the weather condition column, we learned that most of the accidents happened in fair weather, and we can also see that most happened during the day. It makes sense that the three most populous states, California, Florida, and Texas, also have the highest number of entries in the dataset.

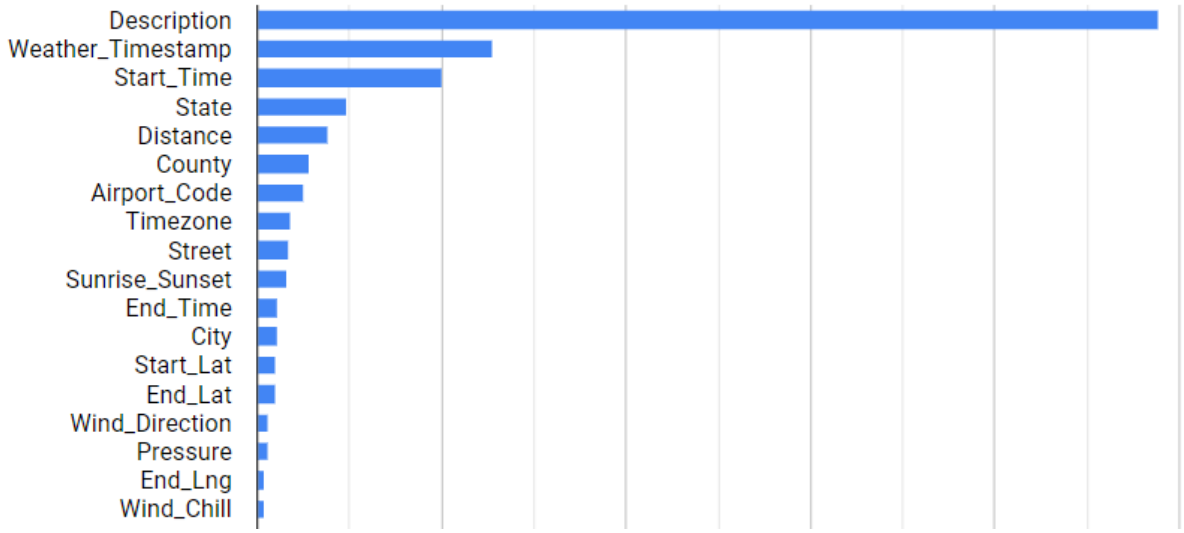
Data cleaning

Each of the models we built used slightly different selections of the variables available, but there were a few columns that we immediately identified as being unhelpful. The street number variable was missing for many of the entries since many accidents happened somewhere such as a highway where there was no street number. We also dropped three ‘twilight’ columns, which calculated whether the accident happened during the day or night based on three methods of measurement. There was already a ‘Sunrise_Sunset’ column that also stated whether the accident happened day or night, so we kept this one and removed the other three. We also dropped rows containing any NA values. With our dataset still containing over two million rows, we decided to optimize memory usage by reformatting some of the int64 columns to have the int8 data type. This limits the number of digits used to store the numerical data, reducing unnecessary memory usage.

When using a machine learning system, we chose Google Cloud’s AutoML as our best candidate. It's simple to use and is quite friendly to new users. It also generously gives a 400-dollar free trial. We began by importing our data into storage buckets. We then ran an AutoML test for classification looking at accident severity as our target column. Our test was optimized for log loss, and we ran it for 12 node hours. Our initial results were quite surprising and looked pretty good.

True label	Predicted label 2	3	4	1
2	99%	0%	0%	0%
3	29%	70%	0%	1%
4	35%	3%	62%	0%
1	20%	1%	0%	79%

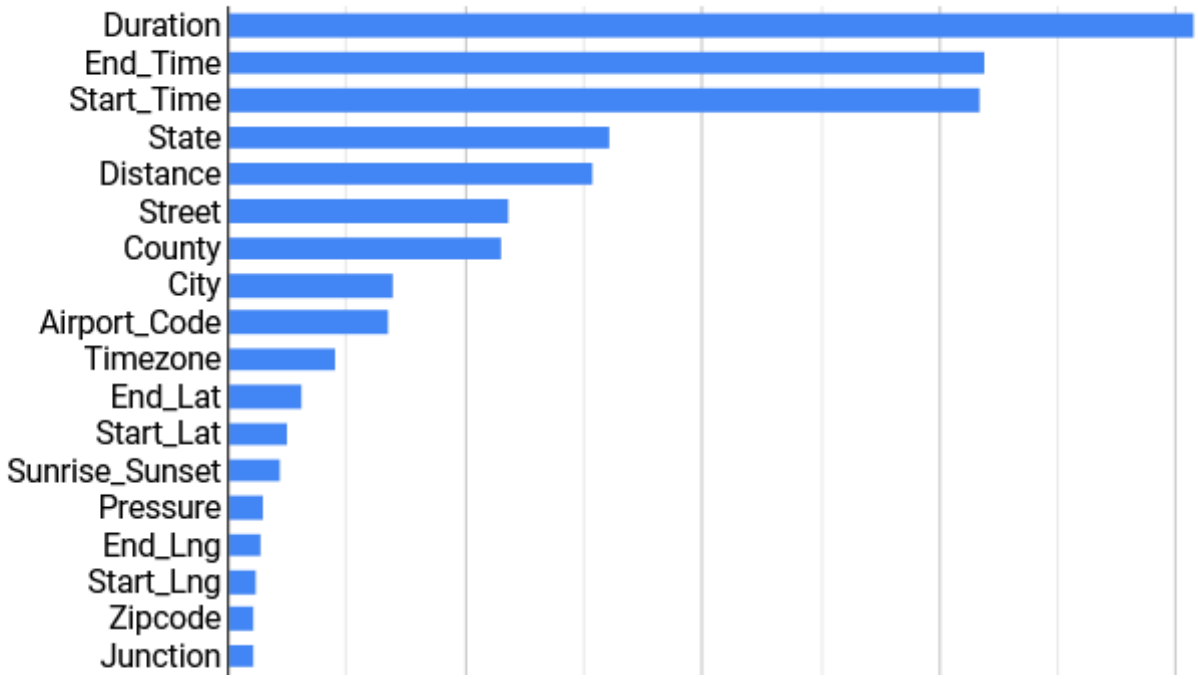
We got a pretty good model made based on the confusion matrix, however, upon looking at the feature importance graph, we had some concerns.



The feature importance graph above heavily relies on the description of the accident. We decided to run another model on AutoML. We dropped the country column after realizing that every accident happened in the same country and removed the ID column because it was irrelevant. We removed weather_timestamp because this is simply the time that the weather conditions were recorded, and is unrelated to the accidents themselves. Finally, we opted to remove the description column. The entries in this column were handwritten, usually stating that an event occurred and the accident's location, which we thought was repetitive in a dataset that includes city and street variables. We also created two new variables to use: duration and weather_condition_simple. Duration was the result of subtracting the accident start time from the end time. The original weather condition variable had over 100 possibilities, including overly specific labels like "Smoke / Windy" and vague conditions like "Wintry Mix". Our new column, weather_condition_simple, has only five possibilities: fair, cloudy, rain, freeze (snow, hail, etc), or

obstruction (for conditions blocking vision like fog or haze). We assigned these to the original assortment of conditions by hand. The resulting AutoML model performed worse than the first one, incorrectly predicting severity 2 for many entries with severity 3 or 4 accidents. Here is what our outputs are:

True label \ Predicted label	2	3	4	1
2	99%	0%	0%	0%
3	39%	55%	3%	2%
4	60%	5%	35%	1%
1	16%	3%	0%	82%



The new duration column was prioritized by AutoML, but it seemed that dropping the description column caused crucial context for the AI to be lost. After seeing this, we knew we would need to take a closer look at the description column to see why it was important.

Investigating Description

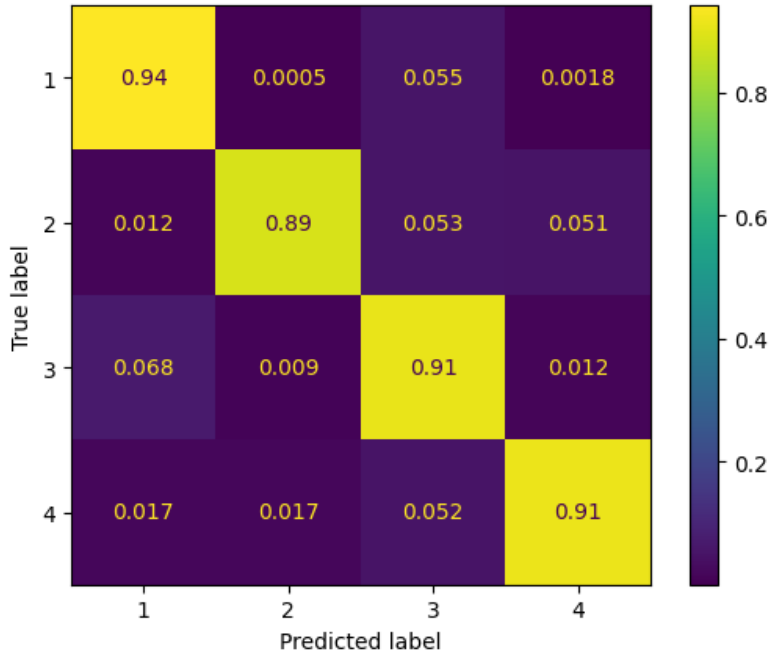
We found several description keywords that had a relationship with severity. ‘Close’ was found almost exclusively in severity 4 descriptions, indicating roads were closed for these entries. ‘Block’, for blocked roads, was found more often in severity 1 and 3. ‘Accident’, while common in all severity levels, was much more frequent in severity 1 and 3. ‘Traffic’ was found almost exclusively in severity 2 descriptions. The keyword ‘construction’, while rare, was also of interest because we discovered that not every single entry was actually representing a car accident - a few rows were only describing the impact of construction on traffic flow. After searching for these terms, we had a better understanding of why the description variable was important for prediction accuracy.

	Description Keyword				
Severity	**'Close'**	**'Block'**	**'Construction'**	**'Accident'**	**'Traffic'**
1	0.05	0.26	0	0.99	(1 row)
2	0.05	0.05	0.003	0.53	0.31
3	0.01	0.37	0.002	0.98	0.01
4	0.9	0.04	0.02	0.62	0.02

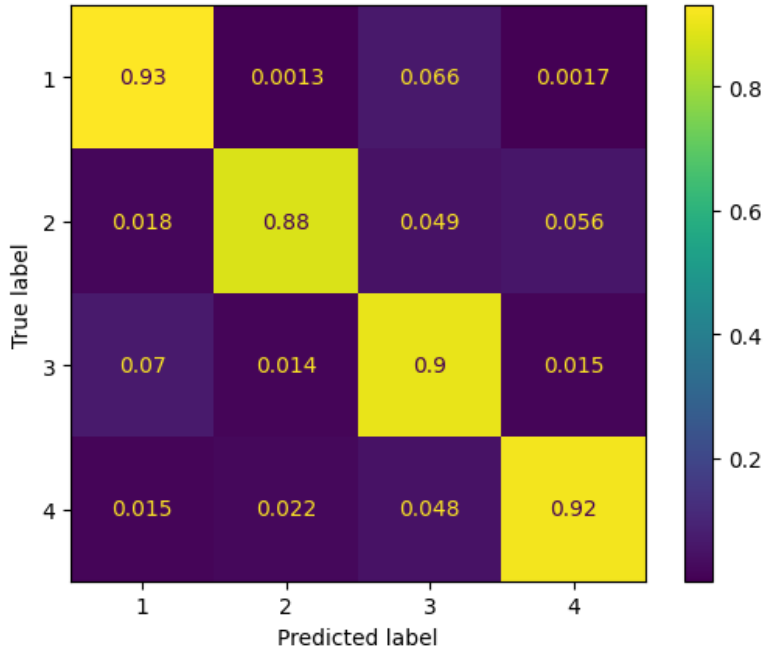
Sklearn Modeling

We wanted to build a gradient boost model with Sklearn to compare with AutoML’s results. This type of model required more data preparation, as the gradient boost classifier can only use numerical data. Boolean columns were created for the previous description keywords, indicating whether a keyword appeared in that row’s description, while description itself was dropped. The columns for roadside and day/night were easily converted to booleans as well. Dummy variables were created for the following columns: state, timezone, wind direction, and weather condition simple. This function allows us to convert columns containing text into numerical data. Some variables like city and county were dropped entirely, as there were too many options to be easily converted to numerical form with dummy variables.

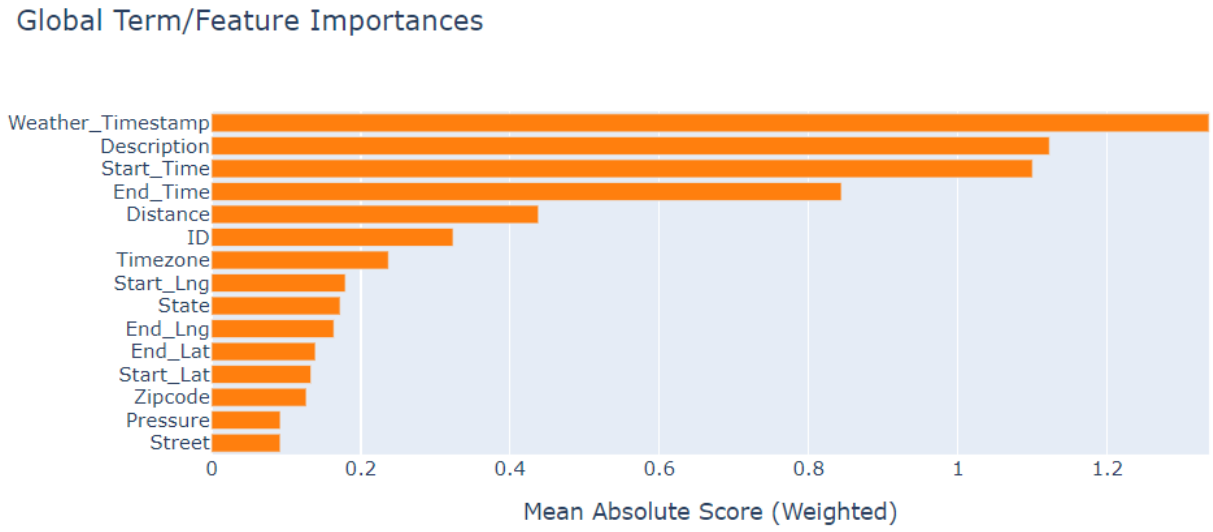
One advantage of Sklearn is the ability to easily balance data for training and testing, something that we did not encounter with AutoML. Because severity 2 was so much more frequent than the other severity levels, we wanted to balance the data so that the model could use equal amounts of each severity to predict, and hopefully avoid assigning severity 2 to everything. Severity 1 was the rarest at about 27,000 rows out of the 2 million row data set, so a random 27,000 rows from each of the four levels was selected to use. We used the following model parameters: 200 estimators, .05 learning rate, and a max depth of 10. The gradient boost took about 15 minutes to run and predicted severity with high accuracy, with accuracies ranging from 89% to 94%. Unfortunately, Sklearn doesn’t support feature importance for multiclass variables, which is what we’re predicting, so we couldn’t see exactly which features the model was utilizing the most.



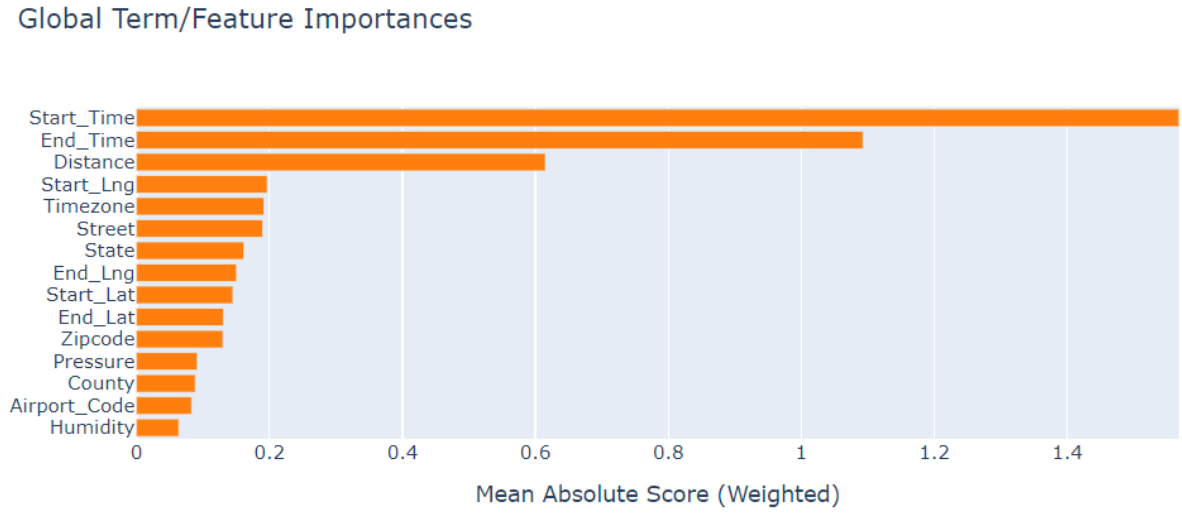
However, we tried running the same model again but without the duration_minutes column, and found that the model made predictions with almost identical accuracies as the first run. This showed that Sklearn wasn't solely relying on this variable, and probably used columns like the start and end time, description keywords, and distance of road affected to make decisions, based on what AutoML found to be important.



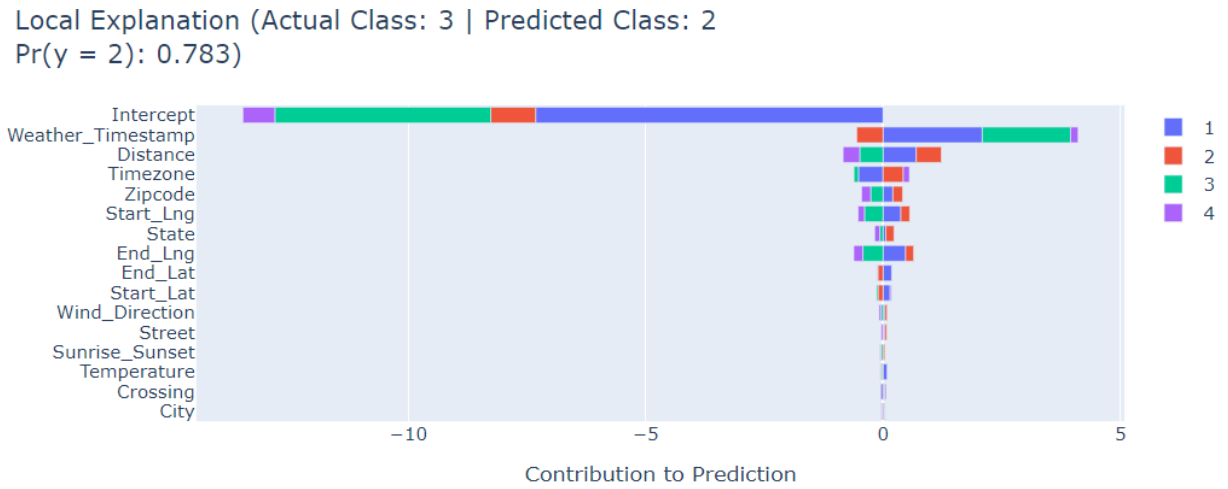
A final platform we wanted to explore was InterpretML. This package lets users get a closer look at feature importance. InterpretML is an easy-to-use Python package that allows users to identify what columns are impacting the machine's decision in machine learning. In this project, we put in our data and got the following results for feature importance:



As you can see, this model heavily relies on the description to determine its decisions. To see what the model would decide without a description, we dropped that column, as well as ID and Weather_Timestamp to see how InterpretML would decide. We then got this output for its feature importance. It heavily uses the time of the accident as well as the distance, which are similar columns to what other models used.



Another thing that InterpretML can do is make local explanations by row. In this instance, the model predicted a 2 for severity, when in reality it was actually a 3.



InterpretML gives the most impactful columns, as well as a score for each of the severities. This is helpful to identify when making machine learning models, as understanding what the model is looking for when making decisions could help make better predictions.

Conclusion

When handling big datasets, it's important to use the right tools for the job. AutoML requires the creation of a Google Cloud account, and careful monitoring of your budget, but it supports a variety of data types. It allows for customization if desired while only requiring minimal input from the user. Sklearn's predictive models are still robust, especially for being a free option, but aren't ideal for large datasets, and most models require extensive data cleaning and preparation in order to be useful. InterpretML is unique in that it can provide feature importance at both the global and local level. Only after reviewing each platform's strengths and weaknesses did we fully understand the value of certain features like the description column. We concluded that AutoML was the best platform for our dataset, as it was fairly easy to use and could handle the large size of our data, and made acceptably accurate predictions for severity. Duration and description were the most important features for its predictions. Sklearn gradient boosting had higher accuracies than AutoML's model, but only used a small portion of the dataset, so it is likely less applicable to the dataset as a whole. InterpretML seemed less suitable for using on its own, and is more helpful for getting additional insight on how different variables impact predictions.