Melissa Mullen

MET CS 777

10/15/2024

**Final Project: Classifying US Motor Vehicle Accident Severity**

**Introduction:**

The data used in this project is sampled from the *US Accidents (2016-2023)* dataset, which was retrieved from Kaggle ([link](https://www.kaggle.com/datasets/sobhanmoosavi/us-accidents/data?select=US_Accidents_March23.csv)). This dataset contains information regarding motor vehicle accidents ranging from February 2016 to March 2023, consisting of 46 columns and 7,728,394 rows. This data was compiled by several APIs, and contains columns relevant to accident severity, weather conditions at the time of the accident, road structures, time of day, and more. I will include a full data dictionary for your reference at the end of this report.

Please note, while the data was readily available to download from Kaggle as a CSV, I had to use a sampled version of the data to run the code locally on my machine. Kaggle offered a sampled version of the original dataset with 500,000 rows, which was still too large for my machine. Consequently, I took a sample of 10,000 rows from the sampled dataset to use for this project. I uploaded both datasets to my Boston University Google Drive for ease of download, as listed below.

* Sampled dataset available on Kaggle (500,000 rows): [Sampled Data (500k)](https://drive.google.com/file/d/1m2w1Ml6OHxm6jfXk7UxWPcZFhZTQwJwE/view?usp=sharing)
* Sampled dataset used for project (10,000 rows): [Project Data (10k)](https://drive.google.com/file/d/1JaLGE2h3UE4brkAsIAvnBq3rEaOTXepO/view?usp=sharing)

The objective of this project was to determine whether you could predict the severity of traffic accidents from weather conditions using various machine learning models in PySpark. To implement this, I used classification via Logistic Regression, Support Vector Machine, and Random Forest models. These models were evaluated via accuracy, precision, recall, and F1 score. The goal was to create at least one model with an accuracy of at least 70% that produces less false negatives than false positives, meaning the model would be more likely to overpredict the severity of an accident rather than underpredict.

To ensure understanding, accuracy measures the number of correct predictions (True Positives + True Negatives) out of all predictions, while precision measures how many positive predictions (True Positives + False Positives) were correct. Recall measures how many actual positive cases (True Positive + False Negatives) were correctly identified. The F1 score is a metric that balances precision and recall values.

**Methodology:**

The dataset required a decent amount of preparation before it could be used in a machine learning model. Beginning with Data\_Preparation.py, where the sample of 10,000 rows was collected, rows with any missing information were dropped, and the accident severity column was transformed into a binary target variable. Severity 0 indicated non-severe accidents (severity levels 1 and 2), and severity 1 indicated severe accidents (severity levels 3 and 4). With this structure, the data was sampled in a way to ensure the dataset was roughly equally distributed, with about 5,000 non-severe accidents and 5,000 severe accidents. This was accomplished by oversampling the minority class (severity 1), as there were significantly more non-severe accidents than severe accidents. I decided to oversample the minority rather than under sample the majority to ensure we had enough data to produce a decent model. Once the sample was created, it was exported to a CSV file (US\_Accidents\_Sampled.csv).

This sampled dataset was loaded into Final\_Project.py, where the remainder of the project code took place. Please note, you could simply load the US\_Accidents\_Sampled.csv file here and avoid using Data\_Preparation.py altogether; I just wanted to include the code that generated the sample data for this project. Within Final\_Project.py, I had to create the same binary target severity variable, which was essentially the same code and process as in Data\_Preparation.py. This resulted in 5049 non-severe accidents, and 4951 severe accidents. From here, I extracted all the different string Weather\_Condition values and categorized each value into one of the following categories - Clear/Fair, Cloudy, Rain, Fog/Haze/Smoke, Thunderstorm, Snow/Ice/Freezing, and Unknown. This categorization would reduce the amount of noise in the Weather\_Condition column and improve model results. I also created a Duration\_Minutes column, which is simply the difference between the Start\_Time and End\_Time columns and represents how long the accident-related traffic took to resolve. Lastly, I selected relevant columns and renamed them for ease of use.

In preparation for the models, I had to perform feature encoding on the Weather\_Condition column, as it contained string values. This was implemented by a combination of StringIndexer() and OneHotEncoder(). StringIndexer() is a feature transformation tool used for encoding categorial string columns into numerical values, while OneHotEncoder() converts these numerical values into unique binary vectors for the machine learning models. I also scaled the relevant columns via StandardScaler(), which is simply a data normalization tool that standardizes values by scaling them to have a consistent range where the average value is zero. Four separate machine learning models were trained and evaluated – Logistic Regression, Support Vector Machine (SVM), Random Forest, and Random Forest with cross-validation. Cross-validation was applied to the last model to find the best hyperparameters (specifically the number of trees and maximum depth of each tree), thus improving the model’s performance. Each model was trained using 80% of the data, while 20% was used for testing and evaluation.

**Results**

I began by using just weather-related columns – Humidity, Distance, Pressure, Temperature, Visibility, Wind\_Speed, Precipitation, and Weather\_Condition. I experimented with the Wind\_Direction and Weather\_Timestamp columns as well, but they did not seem to impact the model performance much. Using these input columns, I had the following results:

**Logistic Regression:**

|  |  |
| --- | --- |
| Confusion Matrix | |
| TN: 576 | FP: 476 |
| FN: 378 | TP: 508 |

* Accuracy: 55.93%
* Precision: 0.5163
* Recall: 0.5734
* F1 Score: 0.5433

576 instances were correctly predicted as non-severe, while 508 instances were correctly predicted as severe. 476 instances were incorrectly predicted as severe but were non-severe, and 378 instances were incorrectly predicted non-severe but were severe.

**Support Vector Machine:**

|  |  |
| --- | --- |
| Confusion Matrix | |
| TN: 492 | FP: 417 |
| FN: 462 | TP: 567 |

* Accuracy: 54.64%
* Precision: 0.5762
* Recall: 0.551
* F1 Score: 0.5633

492 instances were correctly predicted as non-severe, while 567 instances were correctly predicted as severe. 417 instances were incorrectly predicted as severe but were non-severe, and 462 instances were incorrectly predicted non-severe but were severe.

**Random Forest:**

|  |  |
| --- | --- |
| Confusion Matrix | |
| TN: 553 | FP: 343 |
| FN: 401 | TP: 641 |

* Accuracy: 61.61%
* Precision: 0.6514
* Recall: 0.6152
* F1 Score: 0.6328

553 instances were correctly predicted as non-severe, while 641 instances were correctly predicted as severe. 343 instances were incorrectly predicted as severe but were non-severe, and 401 instances were incorrectly predicted non-severe but were severe.

**Random Forest Classifier with Cross-Validation:**

|  |  |
| --- | --- |
| Confusion Matrix | |
| TN: 640 | FP: 301 |
| FN: 314 | TP: 683 |

* Accuracy: 68.27%
* Precision: 0.6941
* Recall: 0.6851
* F1 Score: 0.6896

640 instances were correctly predicted as non-severe, while 683 instances were correctly predicted as severe. 301 instances were incorrectly predicted as severe but were non-severe, and 314 instances were incorrectly predicted non-severe but were severe.

The Random Forest model with cross-validation outperformed the other three models in all three key metrics, indicating that it was the best performing model. However, the goal in this project was to achieve an accuracy of at least 70%, which proved to be problematic. I spent hours playing with hyperparameters and input columns, and was unable to find better results than those listed above. Consequently, I began including columns beyond the weather conditions in the models, and discovered that adding Duration\_Minutes significantly improved the Random Forest model results. Adding Duration\_Minutes to the models produced the following results:

**Logistic Regression:**

|  |  |
| --- | --- |
| Confusion Matrix | |
| TN: 580 | FP: 481 |
| FN: 374 | TP: 503 |

* Accuracy: 55.88%
* Precision: 0.5112
* Recall: 0.5735
* F1 Score: 0.5406

580 instances were correctly predicted as non-severe, while 503 instances were correctly predicted as severe. 481 instances were incorrectly predicted as severe but were non-severe, and 374 instances were incorrectly predicted non-severe but were severe.

**Support Vector Machine:**

|  |  |
| --- | --- |
| Confusion Matrix | |
| TN: 501 | FP: 420 |
| FN: 453 | TP: 564 |

* Accuracy: 54.95%
* Precision: 0.5732
* Recall: 0.5546
* F1 Score: 0.5637

501 instances were correctly predicted as non-severe, while 564 instances were correctly predicted as severe. 420 instances were incorrectly predicted as severe but were non-severe, and 453 instances were incorrectly predicted non-severe but were severe.

**Random Forest:**

|  |  |
| --- | --- |
| Confusion Matrix | |
| TN: 816 | FP: 441 |
| FN: 138 | TP: 543 |

* Accuracy: 70.12%
* Precision: 0.5518
* Recall: 0.7974
* F1 Score: 0.6523

816 instances were correctly predicted as non-severe, while 543 instances were correctly predicted as severe. 441 instances were incorrectly predicted as severe but were non-severe, and 138 instances were incorrectly predicted non-severe but were severe.

**Random Forest Classifier with Cross-Validation:**

|  |  |
| --- | --- |
| Confusion Matrix | |
| TN: 767 | FP: 236 |
| FN: 187 | TP: 748 |

* Accuracy: 78.17%
* Precision: 0.7602
* Recall: 0.800
* F1 Score: 0.7796

767 instances were correctly predicted as non-severe, while 748 instances were correctly predicted as severe. 236 instances were incorrectly predicted as severe but were non-severe, and 187 instances were incorrectly predicted non-severe but were severe.

Like the models without Duration\_Minutes, the Random Forest Model with cross-validation outperformed the other models in all key metrics, and produced an accuracy score greater than 70%. Plus, it produces less false negatives than false positives, meaning the model overpredicts the severity of an accident rather than underpredicts. I hoped the model would produce these results as it makes more sense to overpredict the severity of an accident, especially if this model is used for emergency preparedness or traffic planning purposes.

**Discussion:**

As mentioned above, the Random Forest model was the most effective in classifying accident severity, especially after implementing cross-validation for hyperparameter tuning. This makes sense, as Random Forest models typically manage complex relationships between features and non-linear patterns in data better than Logistic Regression or SVM models.

While it was discouraging to not produce an accuracy greater than 70% with just weather-related input columns, it does make sense. In this dataset, accident severity was gauged based on how long the accident impacted traffic, with 4 being a significant impact on traffic, and 1 being the least impact on traffic. From this perspective, weather plays an interesting role. To explain, while it is fair to say that severe weather is associated with car accidents, severe weather may not be as closely related to traffic delays as originally expected. For example, there are typically less people on the roads during heavy snowstorms, meaning there may be less traffic from a snow-related accident. However, on a perfectly clear day, you could have a multi-car pileup on a highway from an impaired driver, which would cause significant traffic delays. There are a variety of factors associated with car accidents and their consequential traffic impacts, and it would be hard to build a model with high accuracy without variables such as speed limit at the scene of the accident, whether the driver was impaired, and the number of vehicles involved in the accident. After all, higher speeds, impaired driving, and multiple vehicles being involved all typically lead to more severe accidents.

The dataset did include some columns relevant to the layout of the road at the scene of the accident, such as if an amenity were nearby, or if a junction, roundabout, or stop sign, or traffic signal was present. Surprisingly, adding these columns to the models did not have much impact on results. The Duration\_Minutes column was the only added feature that boosted the models’ key metrics significantly, which makes sense, considering accidents that caused traffic for longer periods of time were likely labeled as more severe than those that caused traffic for a brief period of time. By adding Duration\_Minutes to the model, I was able to surpass my accuracy goal of at least 70%.

**Conclusion:**

This project demonstrated how to apply machine learning models using PySpark for the analysis of the *US Accidents (2016-2023)* dataset. After a decent amount of data preparation, many different input column combinations, and much hyperparameter tuning, the Random Forest model with cross-validation produced the best results, and is consequently the best model for classifying accident severity. With the Duration\_Minutes column included, I was able to build a model with 78.17% accuracy that is more likely to overpredict the severity of an accident rather than underpredict. In implementing this model, I learned that there is an association between accident severity and weather conditions, and future work could include identifying the specific higher risk weather conditions that tend to be present with more severe accidents. By identifying these high-risk weather conditions, emergency personnel could best prepare for accident response by having an appropriate number of staff and resources available before these weather conditions occur.

**Appendix**

**Data Dictionary:**

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| ID | String | A unique identifier of the accident record. |
| Source | String | Source of the raw accident data |
| Severity | Decimal | The severity of the accident, ranging from 1 (least impact on traffic) to 4 (significant impact on traffic). |
| Start\_Time | Date | Start time of the accident (in local time zone) |
| End\_Time | Date | End time of the accident (in local time zone). Indicative of when the impact on traffic has resolved. |
| Start\_Lat | Decimal | Latitude in GPS coordinates of accident starting point. |
| Start\_Lng | Decimal | Longitude in GPS coordinates of accident starting point. |
| End\_Lat | Decimal | Latitude in GPS coordinates of accident ending point. |
| End\_Lng | Decimal | Longitude in GPS coordinates of accident ending point. |
| Distance(mi) | Decimal | Length of road (in miles) affected by the accident. |
| Description | String | Description of the accident (reported by human source). |
| Street | String | Street name of address of accident. |
| City | String | City of address of accident. |
| County | String | County of address of accident. |
| State | String | State of address of accident. |
| Zipcode | String | Zip code of address of accident. |
| Country | String | Country of address of accident. |
| Timezone | String | Time zone of accident. |
| Airport\_Code | String | An airport-based weather station that is closest to the accident. |
| Weather\_Timestamp | Date | A timestamp of the weather observed during the accident (in local time). |
| Temperature(F) | Decimal | Temperature at time of accident (in Fahrenheit). |
| Wind\_Chill(F) | Decimal | Wind chill at time of accident (in Fahrenheit). |
| Humidity(%) | Decimal | Humidity at time of accident (in percentage). |
| Pressure(in) | Decimal | Air pressure at time of accident (in inches). |
| Visibility(mi) | Decimal | Visibility at time of accident (in miles). |
| Wind\_Direction | String | Wind direction at time of accident. |
| Wind\_Speed(mph) | Decimal | Wind speed at time of accident (in miles per hour). |
| Precipitation(in) | Decimal | Precipitation (in inches) at time of accident (if any). |
| Weather\_Condition | String | Weather conditions at time of accident (rain, snow, thunderstorm, fog, etc.). |
| Amenity | Boolean | Presence of amenity near the accident. |
| Bump | Boolean | Presence of speed bump near the accident. |
| Crossing | Boolean | Presence of a crosswalk near the accident. |
| Give\_Way | Boolean | Presence of a give\_way near the accident. |
| Junction | Boolean | Presence of a junction near the accident. |
| No\_Exit | Boolean | Presence of a no\_exit near the accident. |
| Railway | Boolean | Presence of a railway near the accident. |
| Roundabout | Boolean | Presence of a roundabout near the accident. |
| Station | Boolean | Presence of a station near the accident. |
| Stop | Boolean | Presence of a stop near the accident. |
| Traffic\_Calming | Boolean | Presence of traffic\_calming near the accident. |
| Traffic\_Signal | Boolean | Presence of a traffic signal near the accident. |
| Turning\_Loop | Boolean | Presence of a turning\_loop near the accident. |
| Sunrise\_Sunset | String | Period of day (day or night) based on sunrise/sunset at time of accident. |
| Civil\_Twilight | String | Period of day (day or night) based on civil twilight at time of accident. |
| Nautical\_Twilight | String | Period of day (day or night) based on nautical twilight at time of accident. |
| Astronomical\_Twilight | String | Period of day (day or night) based on astronomical twilight at time of accident. |