

Real-Time Accident Prediction Deployment

Maryam Saamin

Abstract. This study investigates what makes traffic accidents more severe using data from February 2016 to March 2023, covering all 49 states in the USA. I analyzed 7.7 million accident records with 46 different features, such as weather conditions, time of day, and traffic-related factors. I used machine learning models like Random Forests, XGBoost and Decision Tree classifier to predict accident severity, achieving an accuracy range of 85% to 87%. My findings reveal significant differences in precision, recall, and F1-score across different levels of severity. This research provides valuable insights for developing strategies to prevent and reduce the impact of accidents. By understanding these factors, transportation authorities and urban planners can work towards creating safer roads and minimizing tragic incidents. Additionally, I implemented a simple Flask application to make my findings more accessible and usable.

Keywords. US-Accident, Severity of Accident, ML Prediction, Real-Time Prediction, Deployment, Big Data Analytics

1. Introduction

In this study, I'm exploring what makes traffic accidents more severe in the USA from February 2016 to March 2023. I'm using a huge dataset with 7.7 million accident records from all 49 states as a Big Data. I'm looking at things like weather, time of day, location and other factors to see how they affect accident severity. I'm using machine learning tools like Random Forests, XGBoost, and Decision Trees to predict how bad accidents might be.

The results so far show that I can predict severity with an accuracy range of 85% to 87%, but there are some differences depending on how severe the accidents are.

To help understand the data better, I've made some visualizations:

1. I show the top 15 weather conditions that are most common in accidents, including the worst ones (severity level 4).
2. I've looked at how severe accidents are at different times of the day, both on weekdays and weekends, to see if there are any patterns.
3. I've also looked at how many accidents happen each month, including the really bad ones (severity level 4).

I'm also using different machine learning methods to make better predictions about accident severity. By using methods like Random Forests and XGBoost, I hope to improve our understanding and accuracy.

Lastly, I've made a simple Flask application that anyone can use to see and interact with my findings.

2. Objectives

1. Data Preprocessing:
 - Relevant data on traffic accidents, including attributes such as weather conditions, time, POI, and traffic-related variables. (Region, Accident and incident, etc.)
 - Clean and preprocess the data to handle missing values, outliers.
2. Feature Engineering:

- Create meaningful features from raw data. For example:
 - Extract time-related features (hour of the day, day of the week).
 - Encode categorical features (e.g., weather conditions, POI, etc.) using one-hot encoding or embeddings.
- 3. Exploratory Data Analysis:
 - Visualize the relationships between features and accident severity.
 - Identify any trends, anomalies, or patterns.
- 4. Model Building and Evaluation:
 - Develop predictive models using ML techniques (e.g., random forests, gradient boosting, decision tree and neural network model).
 - Split the dataset into training and validation sets.
 - Evaluate model performance using appropriate metrics (accuracy, precision, recall, F1-score).
- 5. Using Model for running Flask Application
 - After creating Flask Application, running locally.
 - Running the application in Microsoft Azure with Web App service.

3. Key Data Points and Columns

1. Location and Time:

- Start_Time: The timestamp when the accident was reported.
- End_Time: The timestamp when the accident was cleared or resolved.
- Start_Lat, Start_Lng: Latitude and longitude coordinates of the accident location.
- End_Lat, End_Lng: Latitude and longitude coordinates of the accident's end location (if applicable).
- Distance(mi): The distance of the accident from a reference point (e.g., intersection, landmark).

2. Description and Context:

- Description: A natural language description of the accident.
- Weather_Timestamp: The timestamp when weather conditions were recorded.
- Weather_Condition: Describes the weather conditions at the time of the accident.
- Timezone: The time zone of the accident location.

3. Severity and Impact:

- Severity: The severity level of the accident (e.g., minor, moderate, severe).
- Amenity, Bump, Crossing, etc.: Binary indicators for the presence of certain amenities or road features near the accident location.

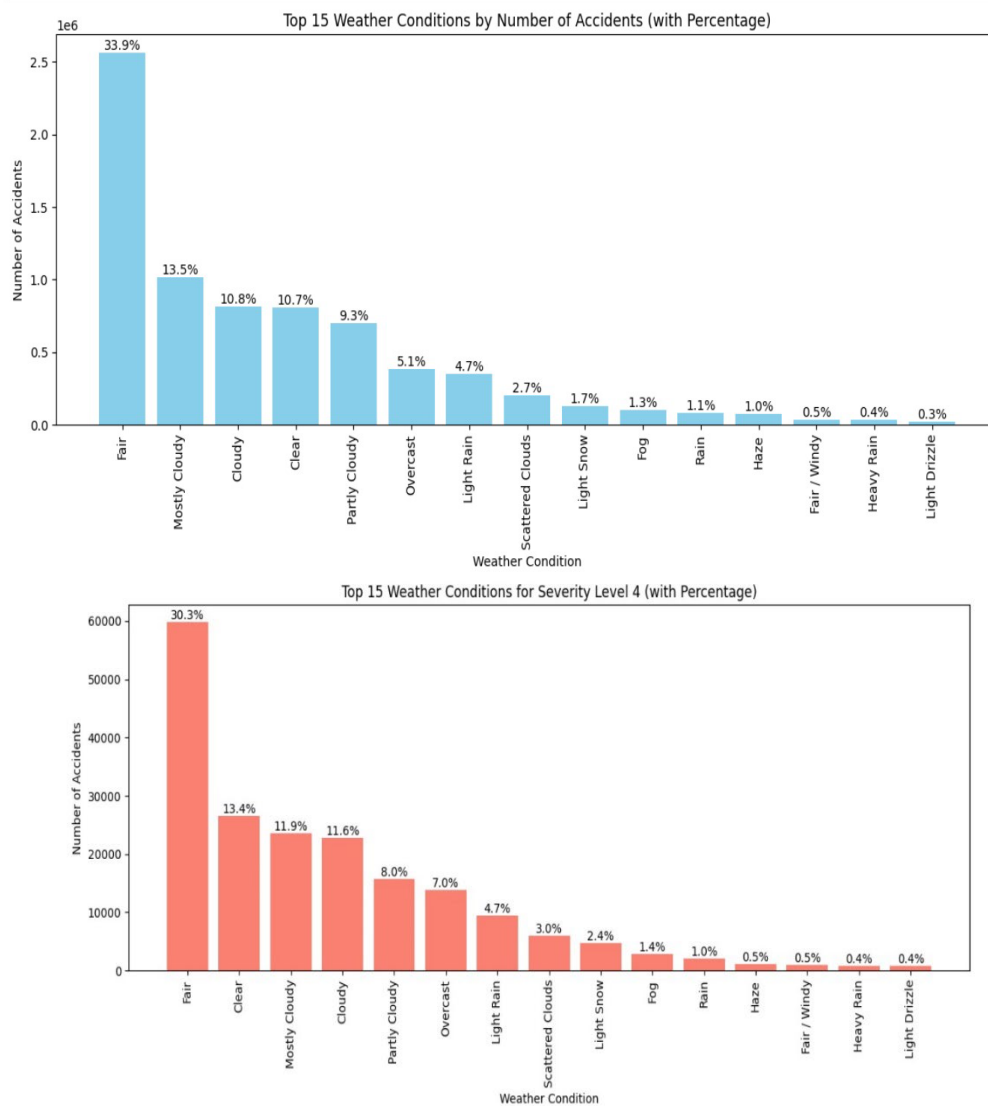
4. Points of Interest (POI):

- Railway, Station, Traffic_Signal, etc.: Binary indicators for the presence of specific points of interest near the accident location.

4. Data Visualization

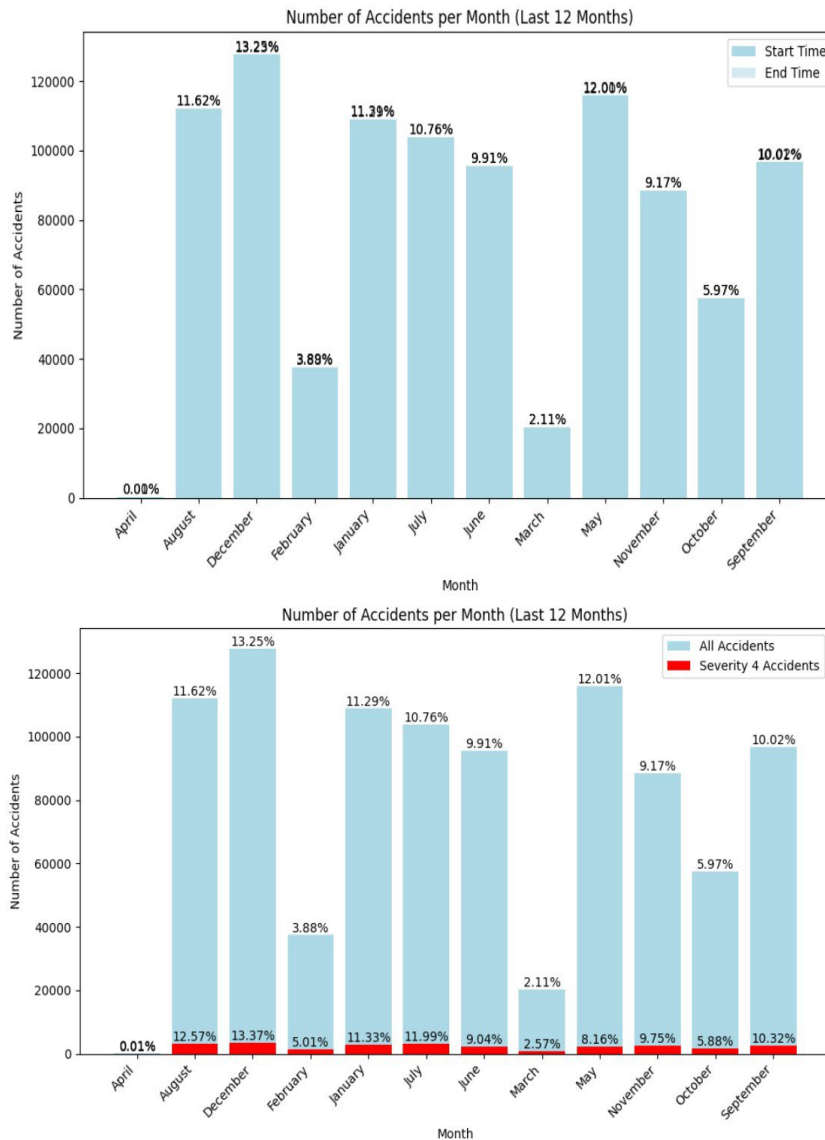
Weather condition:

Out of the 144 different weather types, fair weather is the most common among the top 15 conditions for both total accidents and those marked as severity level 4. This shows that even in good weather, a lot of accidents happen, so it's important to figure out why. I added the percentages.



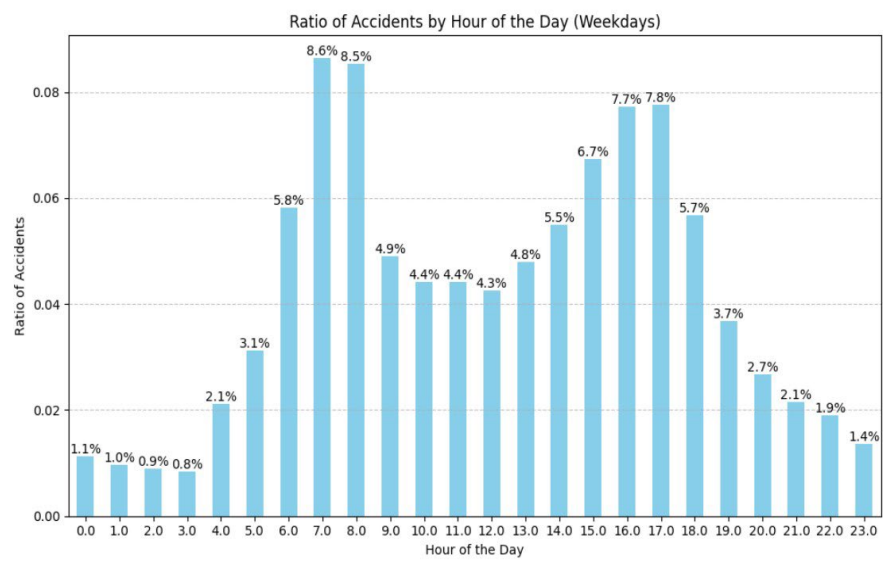
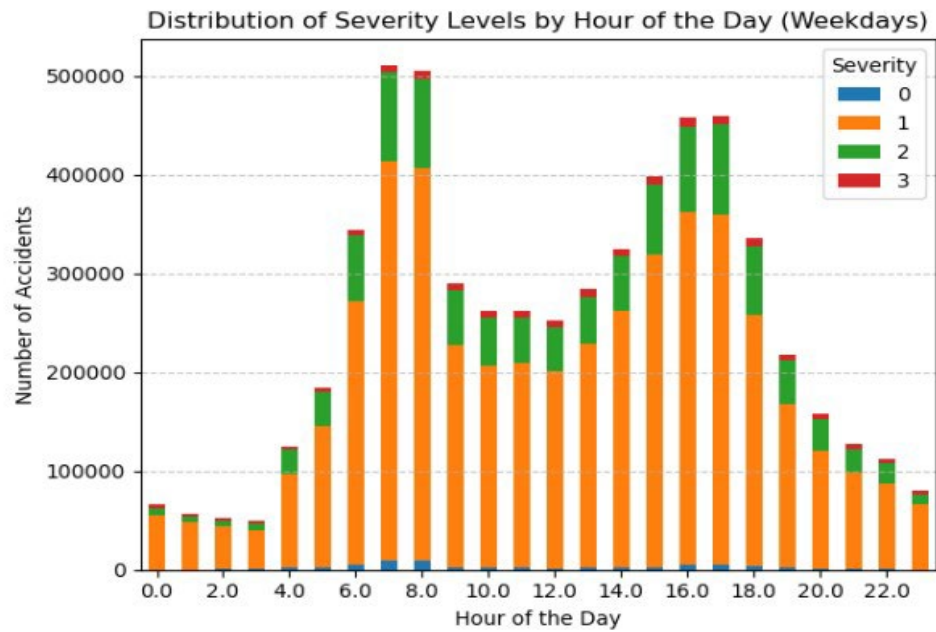
Accidents per month:

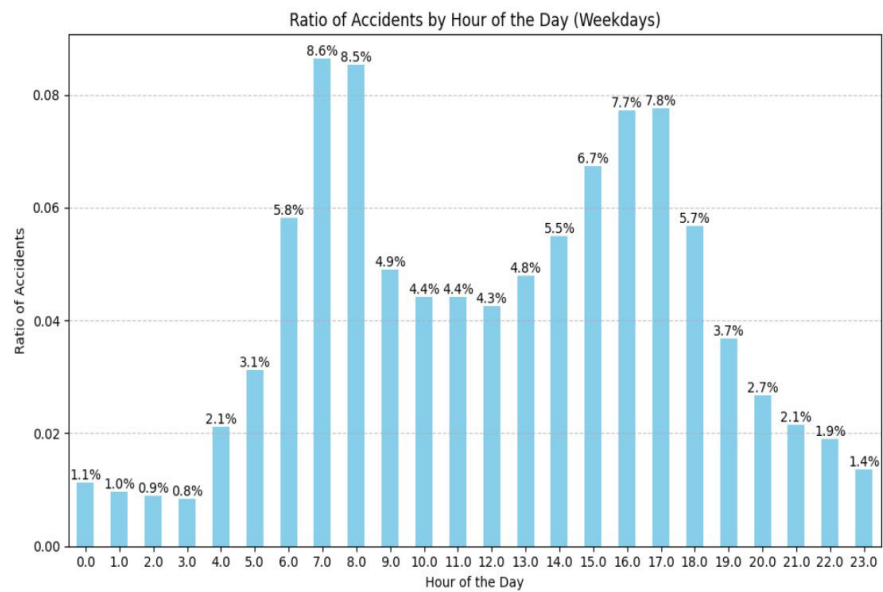
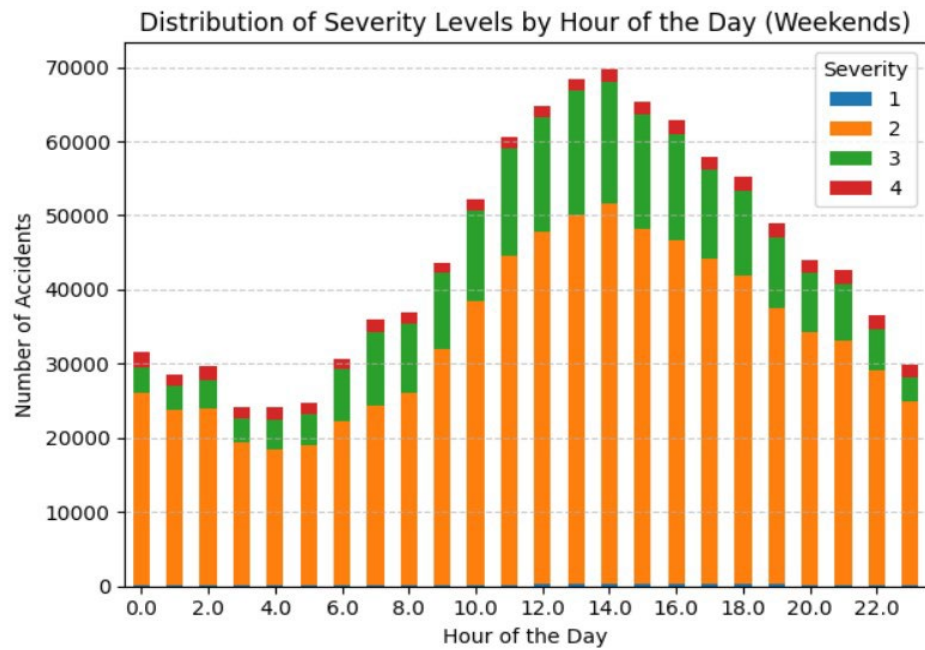
In the distribution of severity levels by months, April consistently records the smallest number of accidents.



Peak Accident Hours on Weekdays and Weekends:

During weekdays, the highest number of accidents occurs between 7 to 8 AM and then again between 4 to 5 PM. However, during weekends, there's a notable increase in accidents between 12 to 2 PM, followed by a decrease afterwards.





5. Models' Accuracy and insights

XGBoost Model:

XGBoost's accuracy sits at around 85%, which means it's getting most predictions right. It's really good at spotting the most common class (class 2), with about 87% precision and 96% recall, meaning it's not missing many of those. But for the other classes, especially the less common ones like class 1 and class 4, its performance isn't as strong. The AUC score of 0.91 indicates that overall, XGBoost is pretty good at telling apart the different classes in the data.

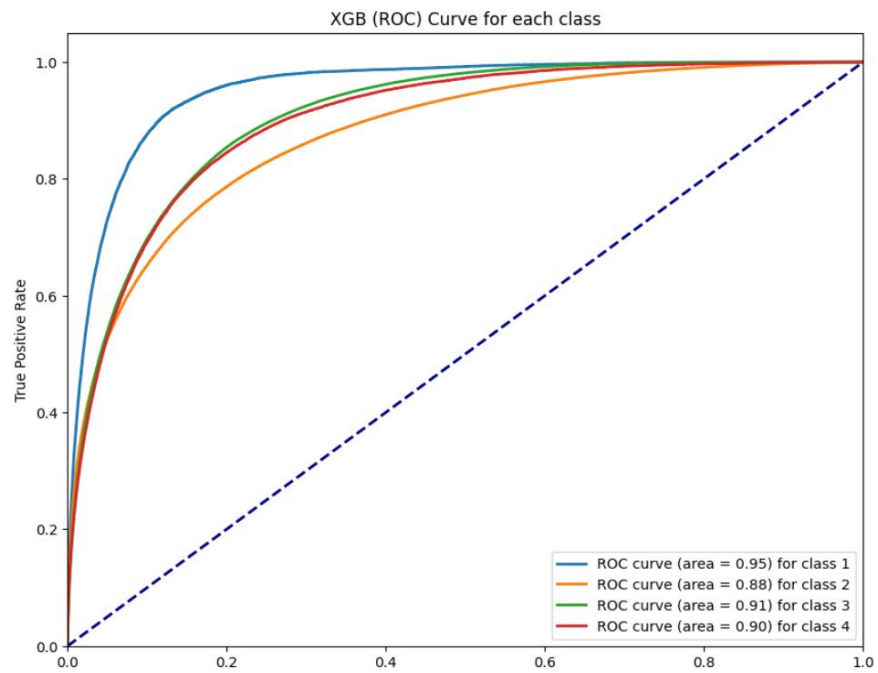
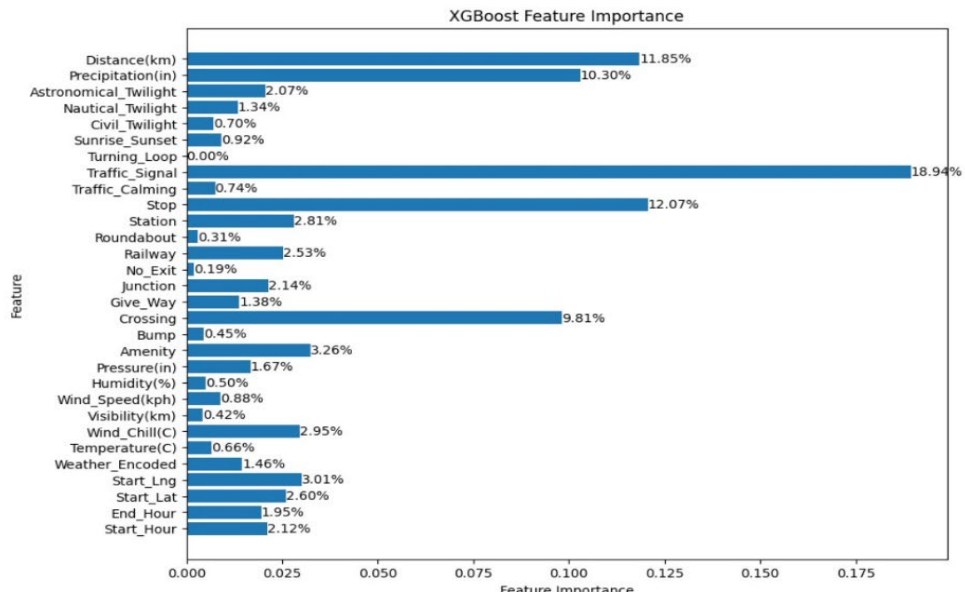
Table 1.

```
XGBoost Accuracy: 0.8496220754762147
Classification Report:
              precision    recall  f1-score   support

     1             0.66       0.05       0.09       13509
     2             0.87       0.96       0.91      1230523
     3             0.71       0.49       0.58       260525
     4             0.62       0.10       0.17        41122

 accuracy              0.85      1545679
  macro avg              0.72       0.40       0.44      1545679
 weighted avg           0.83       0.85       0.83      1545679
AUC Score: 0.909784310883487
```

Figure 1.



Decision Tree Model:

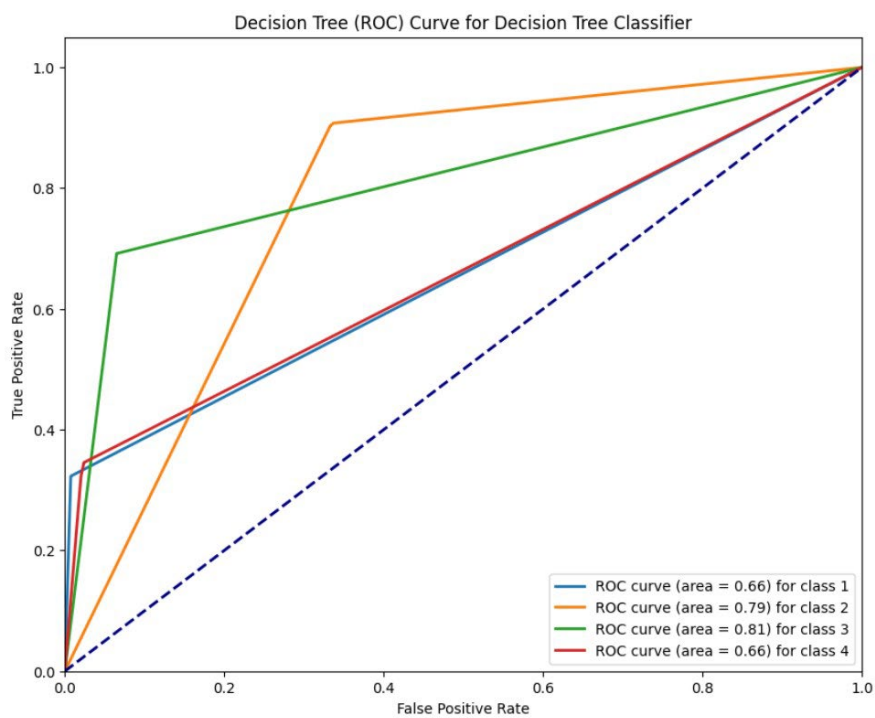
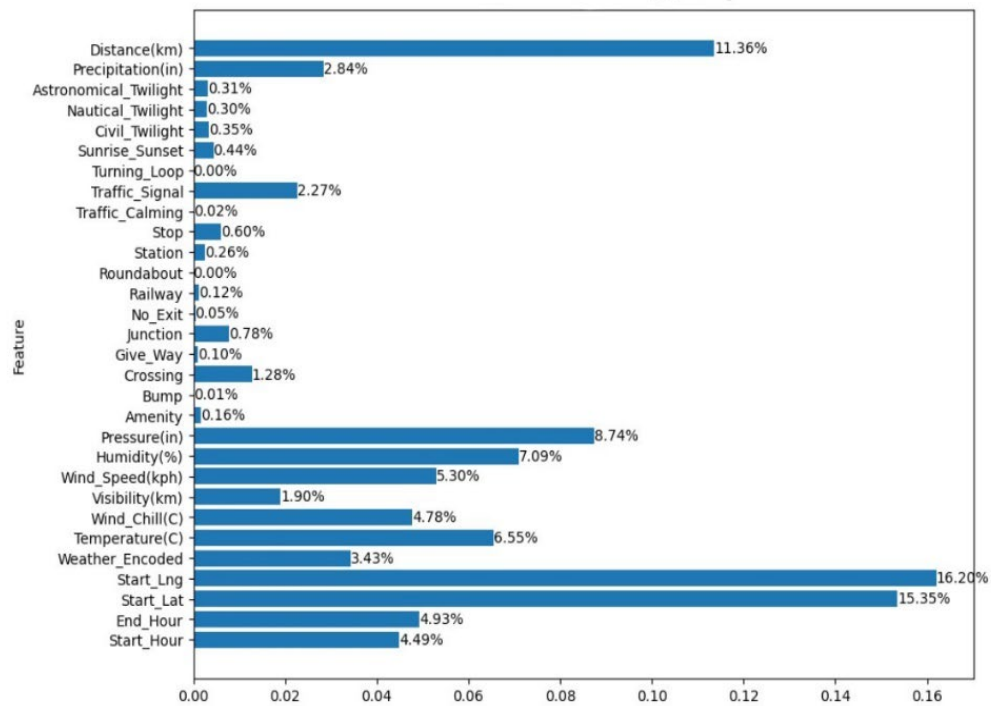
The Decision Tree Classifier achieved an accuracy of about 85%, which means it's doing decently well overall. It's really good at

predicting class 2, with high precision and recall. However, its performance for other classes, like class 1 and class 4, isn't as strong. The AUC score of 0.73 suggests that while it's able to differentiate between classes, it might not be as effective as some other models at doing so.

Table 2.

Decision Tree Classifier Accuracy:				
0.8503499109452868				
Decision Tree Classification Report:				
	precision	recall	f1-score	support
1	0.28	0.32	0.30	13509
2	0.91	0.91	0.91	1230523
3	0.68	0.69	0.69	260525
4	0.30	0.32	0.31	41122
accuracy			0.85	1545679
macro avg	0.54	0.56	0.55	1545679
weighted avg	0.85	0.85	0.85	1545679
AUC Score: 0.7296713700162467				

Figure 2.



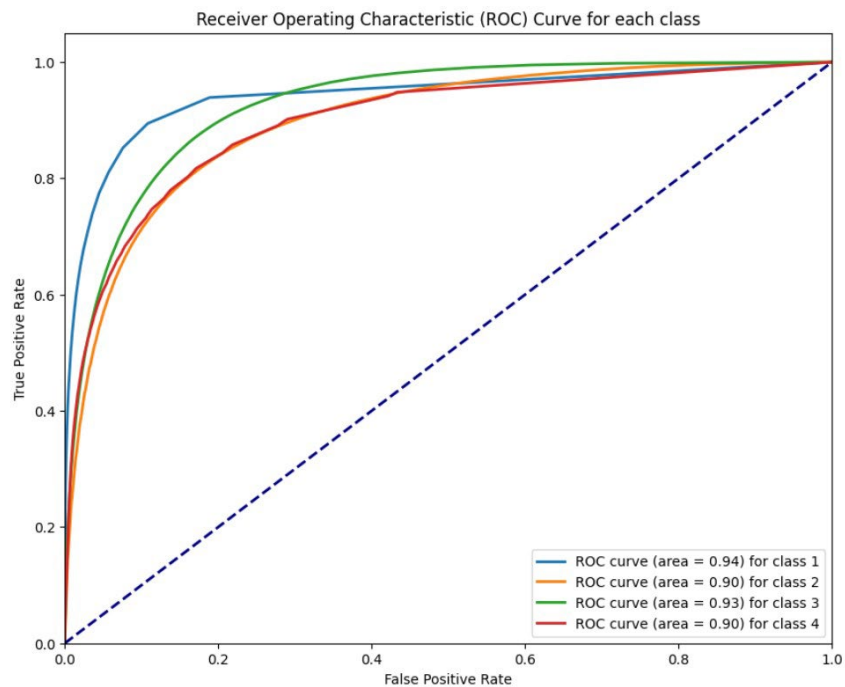
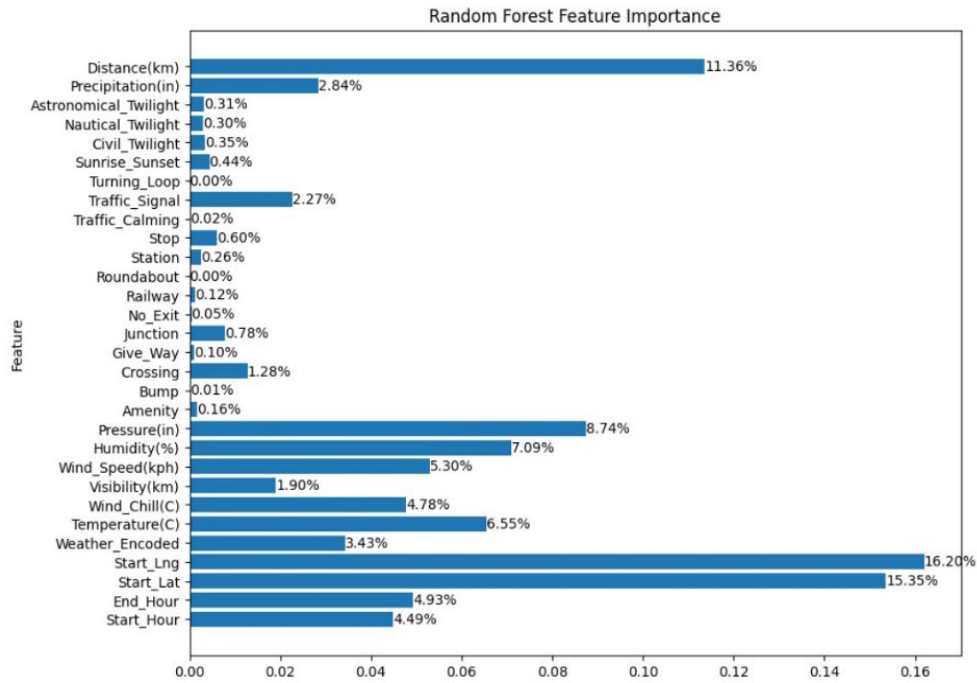
Random Forest Model:

The Random Forest model achieved an accuracy of about 87%, indicating strong overall performance. It excelled in predicting class 2, with high precision and recall. However, its performance varied for other classes, such as class 1 and class 4, where precision and recall were lower. The AUC score of 0.92 suggests that the model is excellent at distinguishing between different classes, making it a robust choice for classification tasks.

Table 3.

Random Forest Accuracy: 0.8652145756007554				
Classification Report:				
	precision	recall	f1-score	support
1	0.77	0.21	0.33	13509
2	0.89	0.96	0.92	1230523
3	0.76	0.56	0.65	260525
4	0.55	0.25	0.34	41122
accuracy			0.87	1545679
macro avg	0.74	0.50	0.56	1545679
weighted avg	0.85	0.87	0.85	1545679
AUC Score: 0.9164990976086071				

Figure 3.



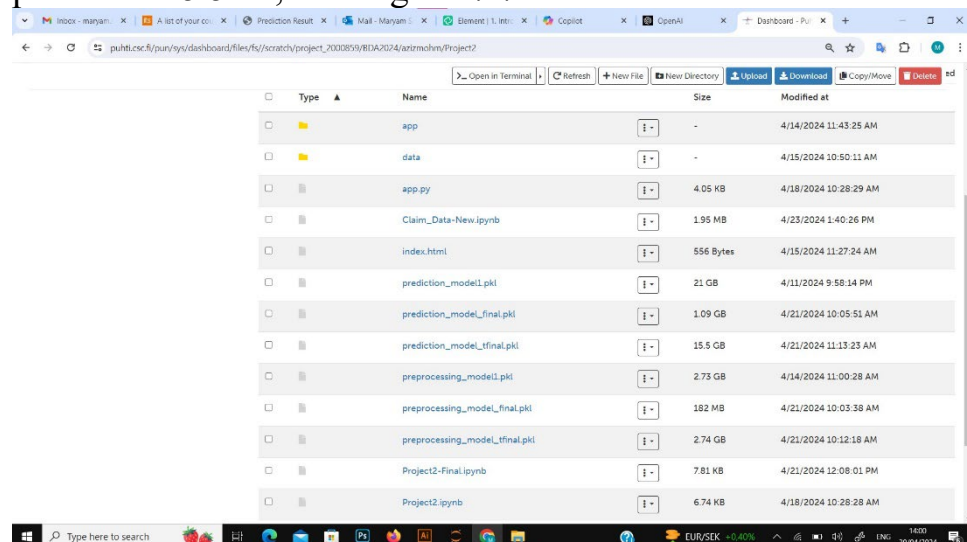
6. Deployment

I used Flask for deployment. Flask is a Python web framework that's lightweight and easy to use. It's great for starting web application projects due to its flexibility and scalability. For map visualizations, Dash by Plotly was utilized. Lastly, Microsoft Azure was employed to host the Flask application and deploy a real-time accident prediction web app.

1. Creating PKL files

- Writing Python code and creating Preprocessing pkl file to pre-process data and save the file to use repetitively
- The same, creating Prediction pkl file to get data and give accuracy as the result and save the file to use repetitively

The preprocessing pkl file is 2.75 G and prediction model pkl file is 15.5 G , running for 7.7 million records.



Type	Name	Size	Modified at
Folder	app	-	4/14/2024 11:43:25 AM
Folder	data	-	4/15/2024 10:50:11 AM
File	app.py	4.05 KB	4/18/2024 10:28:29 AM
File	Claim_Data-New.ipynb	1.95 MB	4/23/2024 1:40:26 PM
File	index.html	556 Bytes	4/15/2024 11:27:24 AM
File	prediction_model.pkl	21 GB	4/11/2024 9:58:14 PM
File	prediction_model_final.pkl	1.09 GB	4/21/2024 10:05:51 AM
File	prediction_model_tf.pkl	15.5 GB	4/21/2024 11:13:23 AM
File	preprocessing_model.pkl	2.73 GB	4/14/2024 11:00:28 AM
File	preprocessing_model_final.pkl	182 MB	4/21/2024 10:03:38 AM
File	preprocessing_model_tf.pkl	2.74 GB	4/21/2024 10:12:18 AM
File	Project2-Final.ipynb	7.81 KB	4/21/2024 12:08:01 PM
File	Project2.ipynb	6.74 KB	4/18/2024 10:28:28 AM

2. Writing Flask application code

- Use the relevant library like Flask, request and ... also Dash for plotting on the map

- Creating the routes and folders needing in running Flask application like Templates folder, Although I simplified the routes
- Creating index.html to get data and result.html to show the result, here the result is illustrating the severity of the accident and showing on the map

```

project_name/
|
├── data/
│   └── US_Accidents_March23.csv
|
├── app/
│   ├── __init__.py
│   ├── routes.py
│   └── templates/
│       └── index.html
|
└── run.py

```

board	Organise	New	Open	Select	Backup
> Project2					
<input type="checkbox"/>	Name	Date modified	Type	Size	
	predict	17/04/2024 11:25	File folder		
	Templates	17/04/2024 11:49	File folder		
	app.py	17/04/2024 11:44	Python Source File	5 KB	
	prediction_model.pkl	15/04/2024 11:29	PKL File	1,464,964 ...	
	preprocessing_model.pkl	15/04/2024 11:28	PKL File	186,135 KB	

Project2 > Templates				
<input type="checkbox"/>	Name	Date modified	Type	Size
	index2.html	17/04/2024 12:31	Chrome HTML Do...	7 KB
	result.html	17/04/2024 10:54	Chrome HTML Do...	1 KB
	result1.html	15/04/2024 11:30	Chrome HTML Do...	1 KB

- First of all, running for small size on the data (500,000 records) on localhost 127.0.0.1/5000 using command prompt and simply run python app.py (name of the Flask Application), the port 5000 is usually considered for this type of application.

Administrator: Command Prompt - python app.py

```

:Users\User\Desktop\Project2>python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
press CTRL+C to quit
* Restarting with watchdog (windowsapi)
* Debugger is active!
* Debugger PIN: 979-889-601

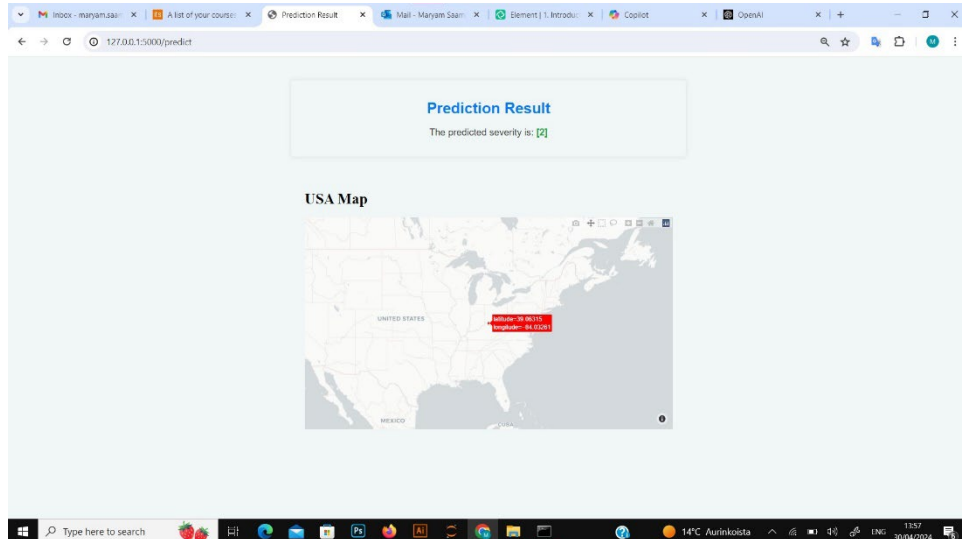
```

127.0.0.1:5000

Accident Severity Prediction

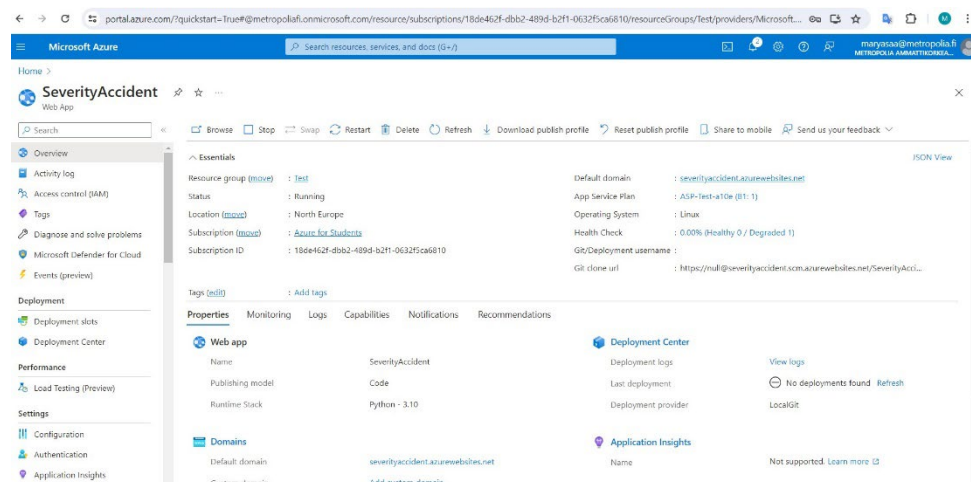
Feature	Value	Feature	Value
Temperature(F)	36	Wind_Chill(F)	33.3
Visibility(mi)	10.9	Wind_Speed(mph)	3.9
Humidity(%)	100.0	Pressure(in)	29.87
Civil_Twilight	high	Nautical_Twilight	Day
Astronomical_Twilight	Day	Precipitation(in)	0
Distance(mi)	0.01	Start_Latitude	39.06
Start_Longitude	-84.0	Start_Time	06
End_Time	07	Weather_Condition	7
Rainor_Sunrise	high	Turning_Left	<input type="checkbox"/>
Alcohol	<input type="checkbox"/>	Bump	<input type="checkbox"/>
Crossing	<input type="checkbox"/>	Give_Way	<input type="checkbox"/>
Junction	<input type="checkbox"/>	No_Fall	<input type="checkbox"/>
Railway	<input type="checkbox"/>	Roundabout	<input type="checkbox"/>
Station	<input type="checkbox"/>	Stop	<input type="checkbox"/>
Traffic_Calming	<input type="checkbox"/>	Traffic_Signal	<input checked="" type="checkbox"/>

Predict

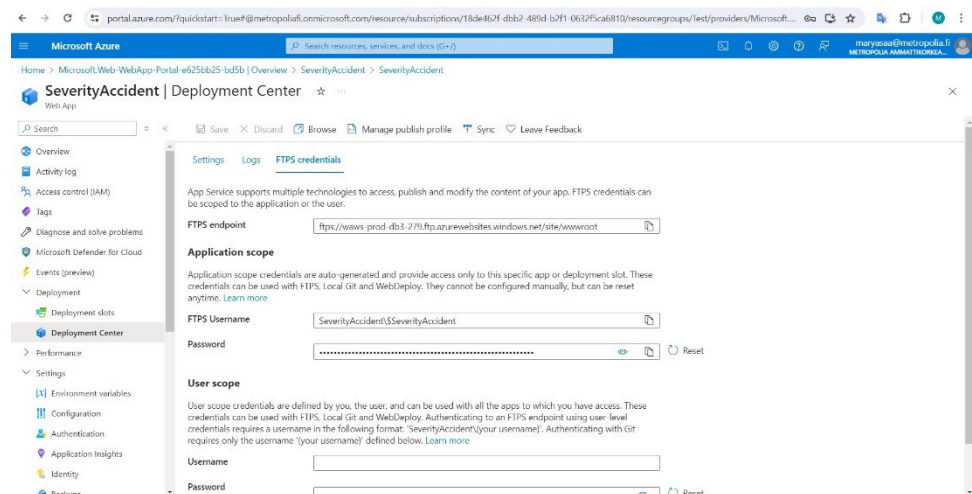
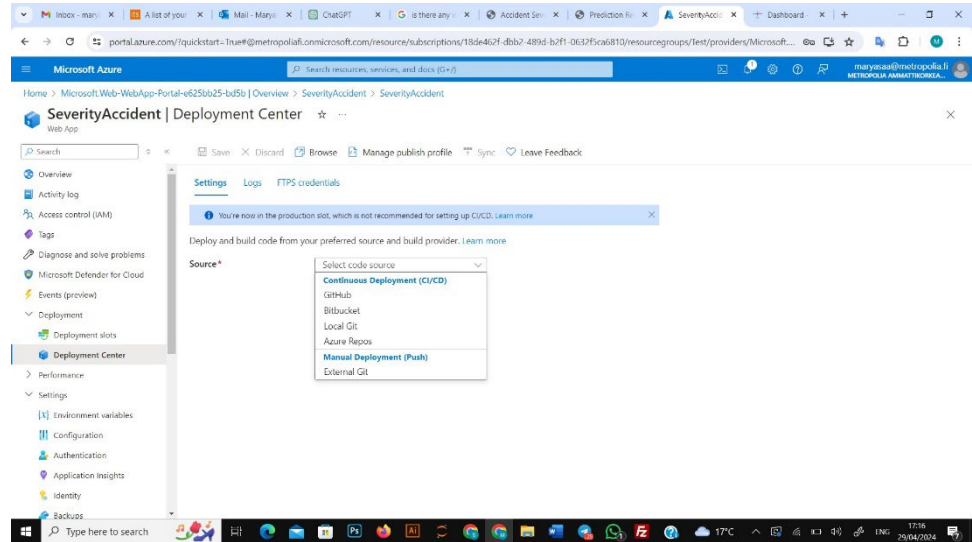


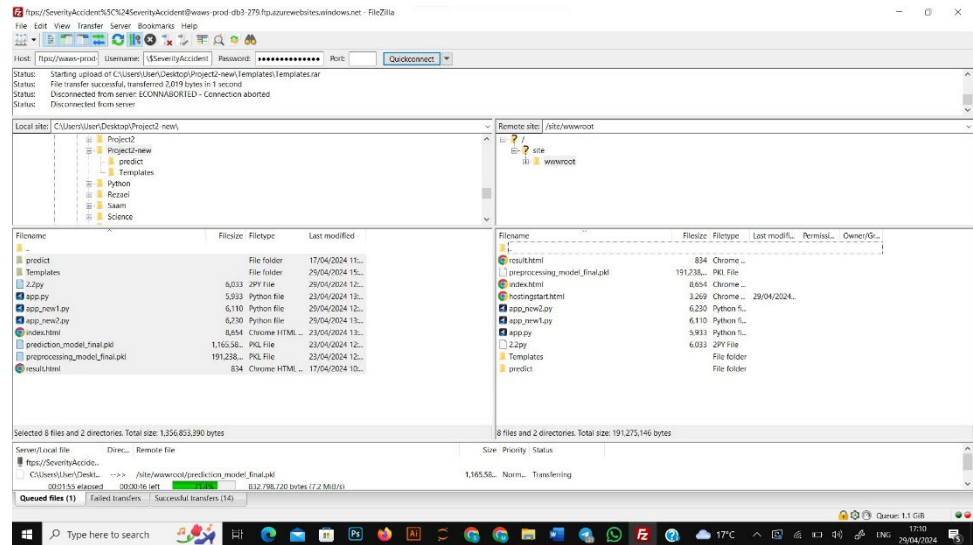
4. Finally, Running through Microsoft Azure

- Creating a Web app through the Azure service



- Some setting in the configuration and environmental variables
- Selecting in Deployment Centre suitable git to access Flask application code files and.pkl files



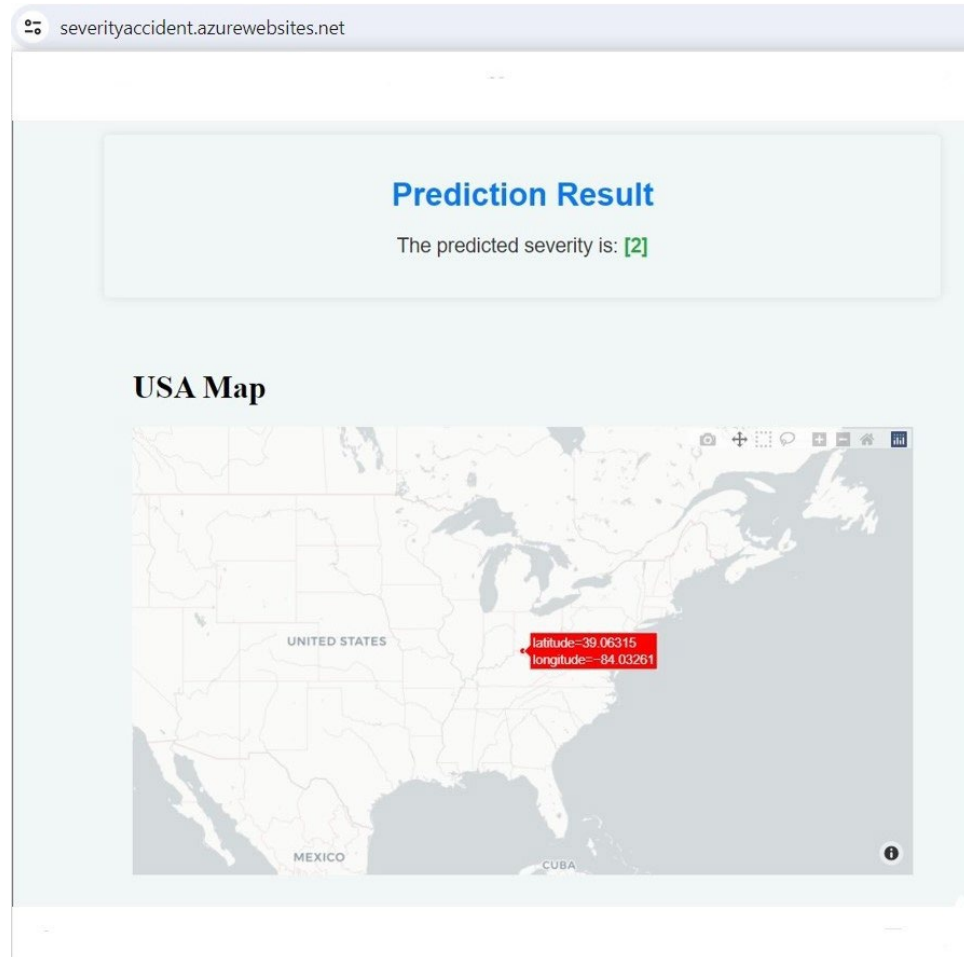


- After successful deployment, click on the web link provided by Azure Web App to run Flask web link

Accident Severity Prediction

Feature	Value	Feature	Value
Temperature(F)	<input type="text" value="36"/>	Wind_Chill(F)	<input type="text" value="33.3"/>
Visibility(mi)	<input type="text" value="10.0"/>	Wind_Speed(mph)	<input type="text" value="3.5"/>
Humidity(%)	<input type="text" value="100.0"/>	Pressure(in)	<input type="text" value="29.67"/>
Civil_Twilight	<input type="text" value="Night"/>	Nautical_Twilight	<input type="text" value="Day"/>
Astronomical_Twilight	<input type="text" value="Day"/>	Precipitation(in)	<input type="text" value="0"/>
Distance(mi)	<input type="text" value="0.01"/>	Start_Latitude	<input type="text" value="39.06"/>
Start_Longitude	<input type="text" value="-84.0"/>	Start_Time	<input type="text" value="06"/>
End_Time	<input type="text" value="07"/>	Weather_Condition	<input type="text" value="2"/>
Sunrise_Sunset	<input type="text" value="Night"/>	Turning_Loop	<input type="checkbox"/>
Amenity	<input type="checkbox"/>	Bump	<input type="checkbox"/>
Crossing	<input type="checkbox"/>	Give_Way	<input type="checkbox"/>
Junction	<input type="checkbox"/>	No_Exit	<input type="checkbox"/>
Railway	<input type="checkbox"/>	Roundabout	<input type="checkbox"/>
Station	<input type="checkbox"/>	Stop	<input type="checkbox"/>
Traffic_Calming	<input type="checkbox"/>	Traffic_Signal	<input checked="" type="checkbox"/>

Predict



7. Conclusion

In summary, this study offers significant insights into understanding and predicting traffic accident severity in the USA. By analyzing vast datasets and employing machine learning models, the research identifies crucial factors such as weather

conditions and temporal patterns. The deployment of a Flask application enhances accessibility to these insights, potentially informing strategies for improving road safety.

References

- [1] Moosavi, Sobhan, Mohammad Hossein Samavatian, Srinivasan Parthasarathy, and Rajiv Ramnath. [“A Countrywide Traffic Accident Dataset.”](#), arXiv preprint arXiv:1906.05409 (2019)
- [2] Moosavi, Sobhan, Mohammad Hossein Samavatian, Srinivasan Parthasarathy, Radu Teodorescu, and Rajiv Ramnath. [“Accident Risk Prediction based on Heterogeneous Sparse Data: New Dataset and Insights.”](#) In proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM, 2019.
- [3] https://smoosavi.org/datasets/us_accidents
- [4] <https://www.kaggle.com/datasets/sobhanmoosavi/us-accidents>