Predicting Impact of Road Accidents on Traffic Flow

Tsu-Hao Fu
Koushiki Basu
Yohan Berg
Gopireddy Avinash Reddy

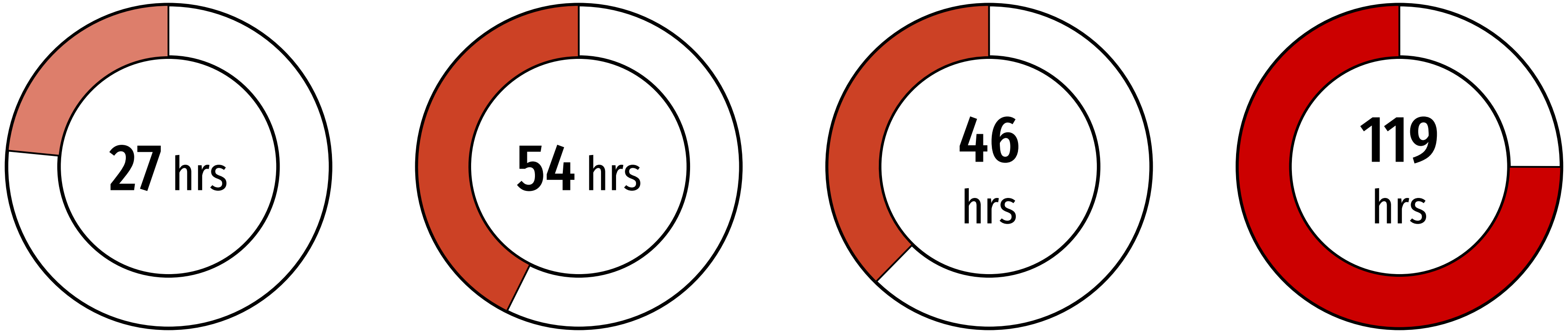
PURDUE UNIVERSITY | Department of Computer Science

Submitted on:4/15/24
Presenting on: 4/23/24

Problem Introduction

Source: https://pxhere.com/en/photo/694335

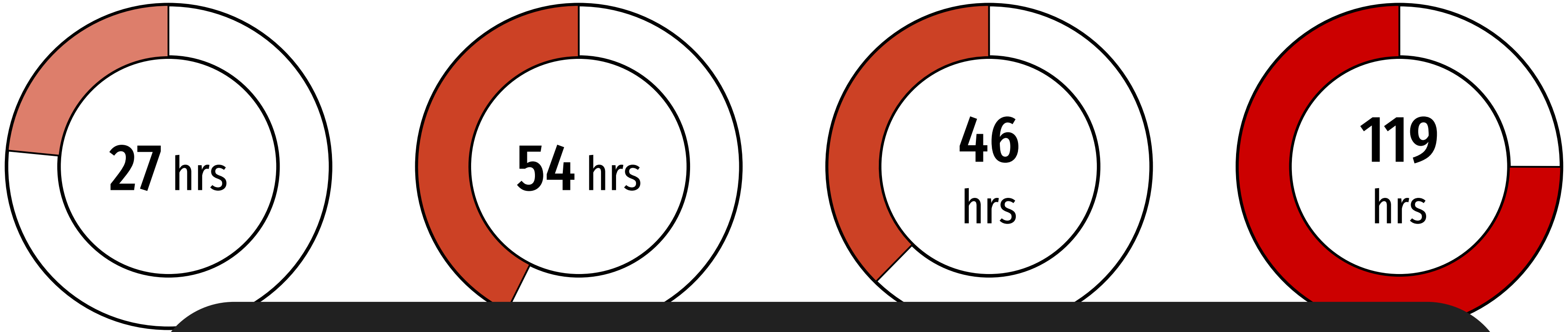
Average Annual Delay per Commuter



2020	2021	2020	2021
US Average	US Average	LA, Long Beach - Anaheim, CA	LA, Long Beach - Anaheim, CA

David Schrank, Luke Albert, Bill Eisele, Tim Lomax. "2021 Urban Mobility Report and Appendices." The Texas A&M Transportation Institute, 2021.

Average Annual Delay per Commuter



2

US A

h

A

A large cause for traffic delays are traffic accidents!

Zeng J, Qian Y, Wang B, Wang T, Wei X. The Impact of Traffic Crashes on Urban Network Traffic Flow. Sustainability. 2019; 11(14):3956. https://doi.org/10.3390/su11143956

Appendices."
The Texas A&M Transportation Institute, 2021.

Our Project Goal

Traffic-Flow Analysis is difficult because of Large, Complex Data impacted with Noise and Real-Time changes in conditions.

- **Analyze several data mining technique and algorithms**
- **Create performant models on the data**
- **Identify the models' distinct advantages and drawbacks**

Dataset

Dataset Used: US Accidents data available on Kaggle, a country-wide dataset of traffic accidents combining multiple APIs such as "MapQuest Traffic" and "Microsoft Bing Map Traffic."

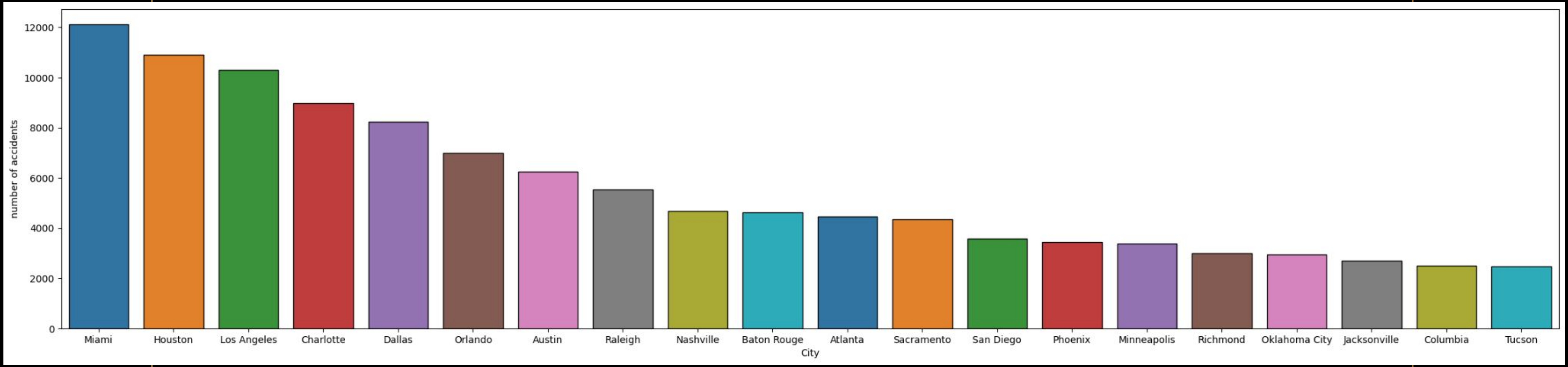
1. 7.7 million entries
2. 46 total attributes
3. Target attribute is accident severity (measures traffic flow impact)

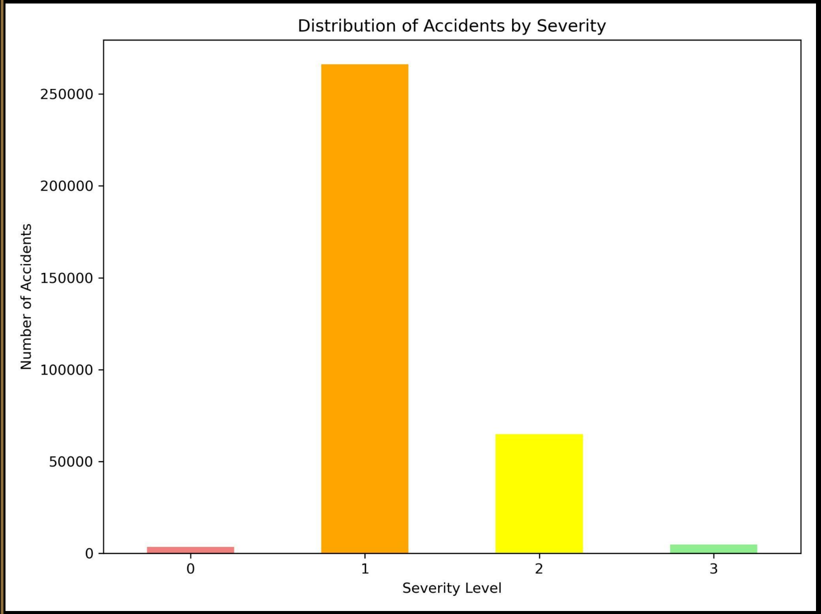
Table1. Details of attributes

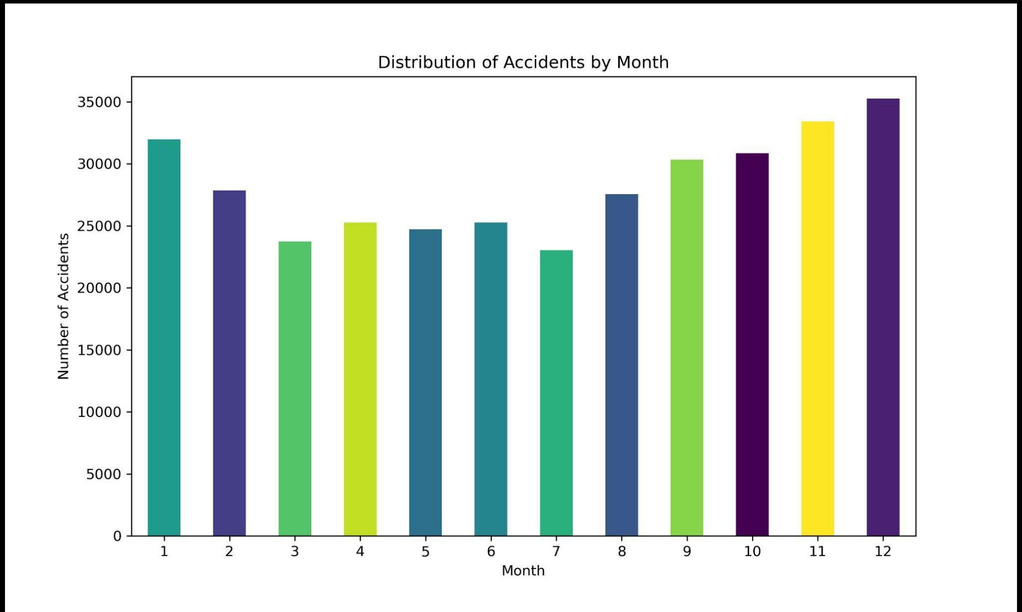
Traffic Attributes	ID, source, severity, start_time, end_time, start_lattitude, end_lattitude, start_longitude, end_longitude, distance, and description
Address Attributes	street, city, county, state, zip-code, country, timezone, and airport_code
Weather Attributes	time, temperature, wind_chill, humidity, pressure, visibility, wind_direction, wind_speed, precipitation, and condition (e.g., rain, snow, etc.)
POI Attributes	amenity, bump, crossing, give-way, junction, no-exit, railway, roundabout, station, stop, traffic calming, traffic signal, and turning loop
Period-of-Day	Sunrise/Sunset, Civil Twilight, Nautical Twilight, and Astronomical Twilight

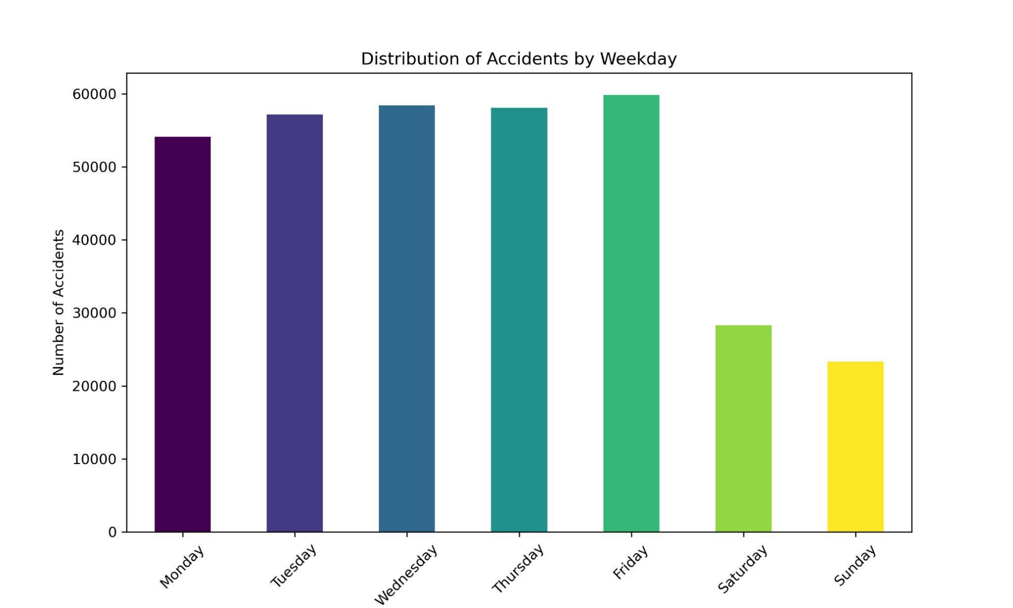
Moosavi, Sobhan, Mohammad Hossein Samavatian, Srinivasan Parthasarathy, and Rajiv Ramnath. "A Countrywide Traffic Accident Dataset.", 2019.

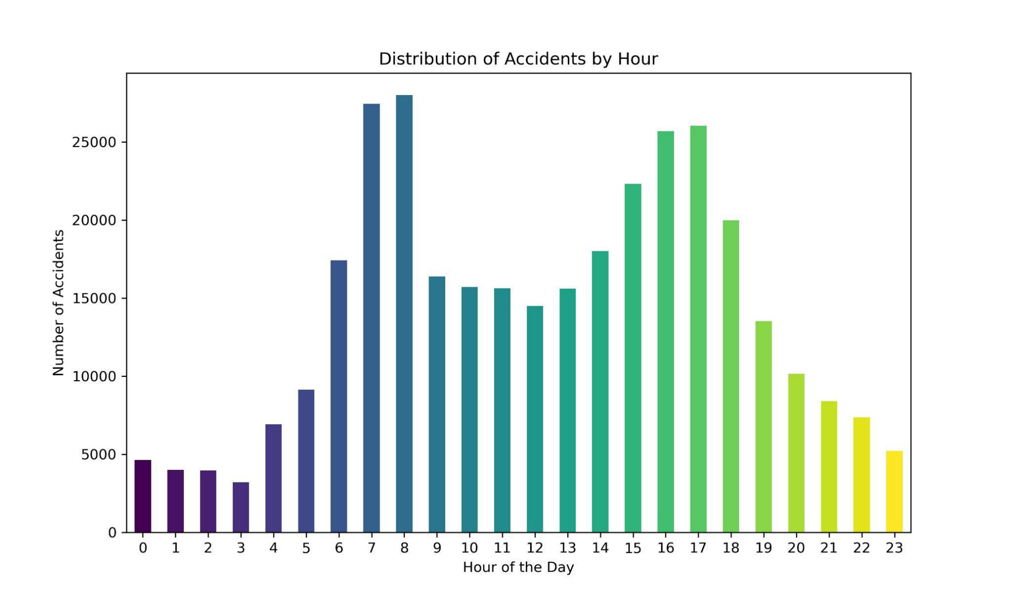
Data Visualization

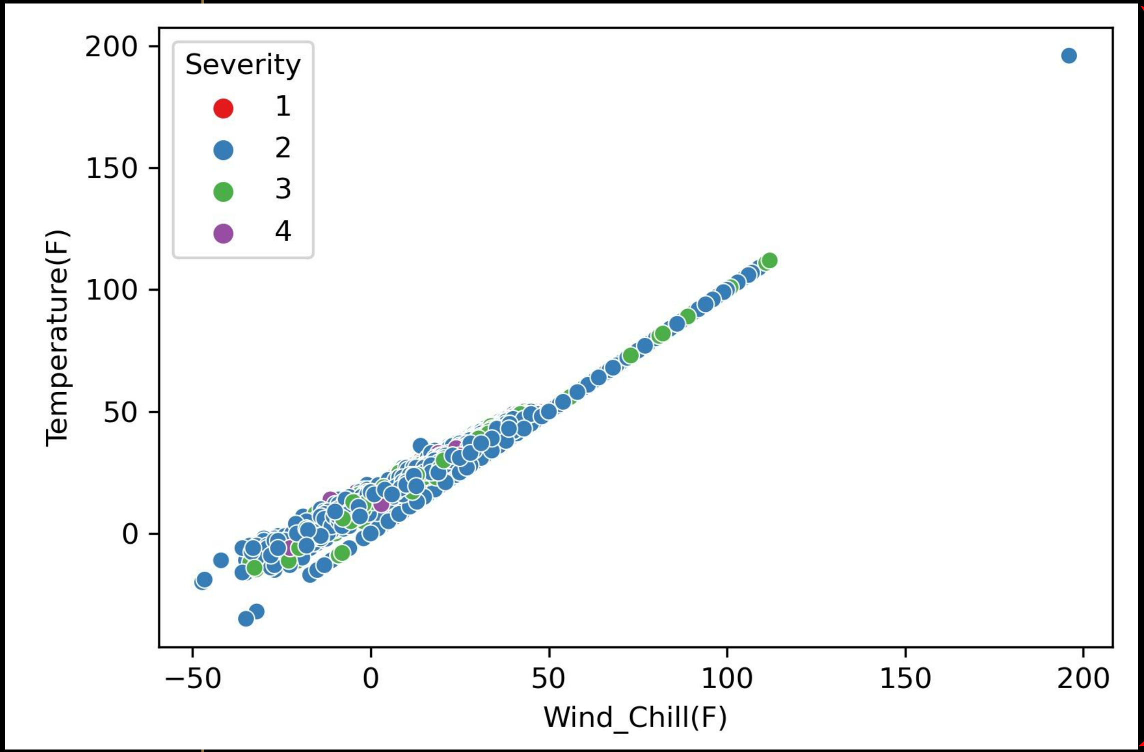


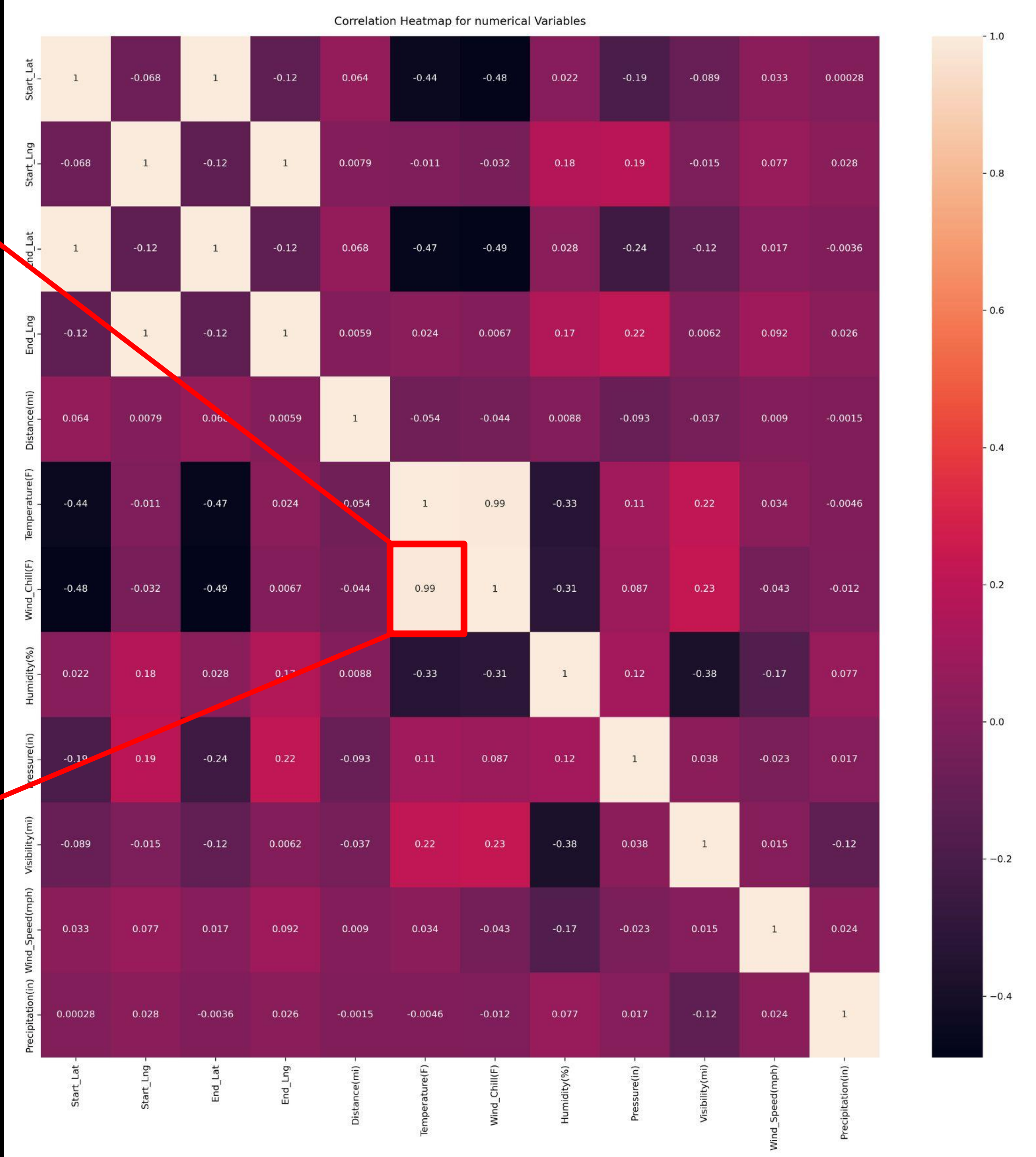

Distribution of Accidents by Severity
Number of Accidents
Severity Level
0
1
2
3
250000
200000
150000
100000
50000


Distribution of Accidents by Month
Number of Accidents
Month
35000
30000
25000
20000
15000
10000
5000


Distribution of Accidents by Weekday
Number of Accidents
60000
50000
40000
30000
20000
10000
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday


Distribution of Accidents by Hour
Number of Accidents
Hour of the Day
25000
20000
15000
10000
5000





PURDUE UNIVERSITY

Department of Computer Science

Tsu-Hao Fu

EDA - Data Preprocessing

Handling Missing Values: High-missing-rate data like "End Lat" and "End Lng" are removed. Other missing values are either dropped or replaced with similar data or common values based on certain groupings.

Outlier Detection and Handling: Outliers are identified using a specific method and are processed or adjusted to make the data more uniform.

Standardization: Numerical data is adjusted to standard scales (mean of 0 and standard deviation of 1), which is important for some machine learning models to perform well.

EDA - Feature Engineering

Binning: Groups complex variables like Wind direction and Weather conditions into simpler categories (e.g., weather into sunny, cloudy, rain; wind direction into north, east, south, west).

Time Extraction: Breaks down detailed DateTime information into easier-to-understand parts like Year, Month, Day, and Hour. It also creates a new feature, "Accident Duration," from the accident start and end times.

Label Encoding: Converts categorical data into numbers by assigning each category a unique integer, making the data suitable for machine learning models.

Frequency Encoding: Replaces categories with the frequency of their occurrence in the dataset, which helps retain the importance of more common categories.

Model Evaluation

- We used the below evaluation metrics for the models:

  Macro Average of F1 Score
  Accuracy
  Precision
  Recall
  F1 Score
  Confusion Matrix

Model Evaluation

- Accuracy is the ratio of correctly classified instances out of the total instances.

$$Accuracy = \frac{TP + TN}{Total}$$

- Precision is the ratio of true positive predictions among all positive predictions made by the model.

$$Precision = \frac{TP}{FP + TP}$$

Model Evaluation

- Recall (also called sensitivity or true positive rate) is the ratio of true positive predictions among all actual positive instances in the dataset.

$$Recall = \frac{TP}{TP + FN}$$

- F1 Score is the harmonic mean of precision and recall. It provides a balance between precision and recall.

$$F1\,Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

Logistic Regression

Introduction to Logistic Regression

- Logistic regression model estimates the probability that a given input belongs to a particular class.
- Primarily used for binary classification problems, but it can be extended to handle multi-class classification as well.
- It uses a strategy called "one-vs-rest" to handle multi-class classification.
- Each class is treated as a binary classification problem, and a separate logistic regression model is trained for each class.

Logistic Regression

Training Logistic Regression Models

We used the following for the regularization term r(W) via the penalty argument, where m is the number of features:

penalty	$r(W)$
`None`	0
ℓ_1	$\|W\|_{1,1} = \sum_{i=1}^{m} \sum_{j=1}^{K} \lvert W_{i,j} \rvert$
ℓ_2	$\frac{1}{2}\|W\|_F^2 = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{K} W_{i,j}^2$
`ElasticNet`	$\frac{1-\rho}{2}\|W\|_F^2 + \rho\|W\|_{1,1}$

Source: https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

Logistic Regression

Training Logistic Regression Models

- We trained the logistic regression model with the following solvers:

lbfgs
newton-cg
sag
saga

Logistic Regression

Training Logistic Regression Models

- The “lbfgs”, “newton-cg” and “sag” solvers only support *l2* regularization.
- The “sag” solver uses Stochastic Average Gradient descent.
- The “saga” solver is a variant of “sag” that also supports the non-smooth penalty="l1". It is also the only solver that supports penalty="elasticnet".
- The “lbfgs” is an optimization algorithm that approximates the Limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm, which belongs to quasi-Newton methods.

Source: https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

Logistic Regression

Classification report for Logistic regression:

Classification Report:

	precision	recall	f1-score	support
0	0.52	0.03	0.06	1081
1	0.86	0.95	0.90	79845
2	0.69	0.49	0.57	19392
3	1.00	0.00	0.00	1443
accuracy			0.84	101761
macro avg	0.77	0.37	0.38	101761
weighted avg	0.83	0.84	0.82	101761

SGDClassifier

- Linear classifiers (SVM, logistic regression, etc.) with SGD training.
- This estimator implements regularized linear models with stochastic gradient descent (SGD) learning.
- The gradient of the loss is estimated each sample at a time and the model is updated along the way with a decreasing strength schedule (i.e., learning rate).

Source:
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html#sklearn.linear_model.SGDClassifier

SGDClassifier

- We used the ‘loss’ parameter to select the loss function to be used.
- ‘hinge’ gives a linear SVM.
- ‘log_loss’ gives logistic regression, a probabilistic classifier.
- We used the default penalty (l2) in the modelling.
- The learning rate used is ‘optimal’.
- ‘optimal’: eta = 1.0 / (alpha * (t + t0)) where t0 is chosen by a heuristic proposed by Leon Bottou.
- Number of iterations with no improvement to wait before stopping fitting, ‘n_iter_no_change’ =5.

Source:
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html#sklearn.linear_model.SGDClassifier

SGDClassifier

Classification report for SGDClassifier with loss = 'hinge'

Classification Report:

	precision	recall	f1-score	support
0	0.35	0.10	0.15	1081
1	0.84	0.97	0.90	79845
2	0.73	0.37	0.49	19392
3	0.00	0.00	0.00	1443
accuracy			0.83	101761
macro avg	0.48	0.36	0.39	101761
weighted avg	0.80	0.83	0.80	101761

SGDClassifier

Classification report for SGDClassifier with loss = ‘log_loss’

```
Classification Report:
              precision    recall  f1-score   support

           0       0.31      0.00      0.01      1081
           1       0.84      0.96      0.90     79845
           2       0.70      0.40      0.51     19392
           3       0.00      0.00      0.00      1443

    accuracy                           0.83    101761
   macro avg       0.46      0.34      0.35    101761
weighted avg       0.80      0.83      0.80    101761
```

KNeighborsClassifier

- Neighbors-based classification is a type of instance-based learning.
- Classification is computed from a simple majority vote of the nearest neighbors of each point.
- A query point is assigned the data class which has the most representatives within the nearest neighbors of the point.
- **KNeighborsClassifier** implements learning based on the ***k*** nearest neighbors of each query point, where ***k*** is an integer value specified by the user.

Source: https://scikit-learn.org/stable/modules/neighbors.html#nearest-neighbors-classification

KNeighborsClassifier

Macro Average of F1 score for KNeighborsClassifier:

- The optimal choice of the value ***k*** is highly data-dependent.
- We used k = 5, 10, 15, 20 for our model.
-

k	**Macro Average**
5	0.39
10	0.35
15	0.34
20	0.33

KNeighborsClassifier

Classification report for KNeighborsClassifier, k=15:

Classification Report:

	precision	recall	f1-score	support
0	0.62	0.02	0.03	1081
1	0.83	0.97	0.90	79845
2	0.71	0.32	0.44	19392
3	1.00	0.00	0.00	1443
accuracy			0.82	101761
macro avg	0.79	0.33	0.34	101761
weighted avg	0.81	0.82	0.79	101761

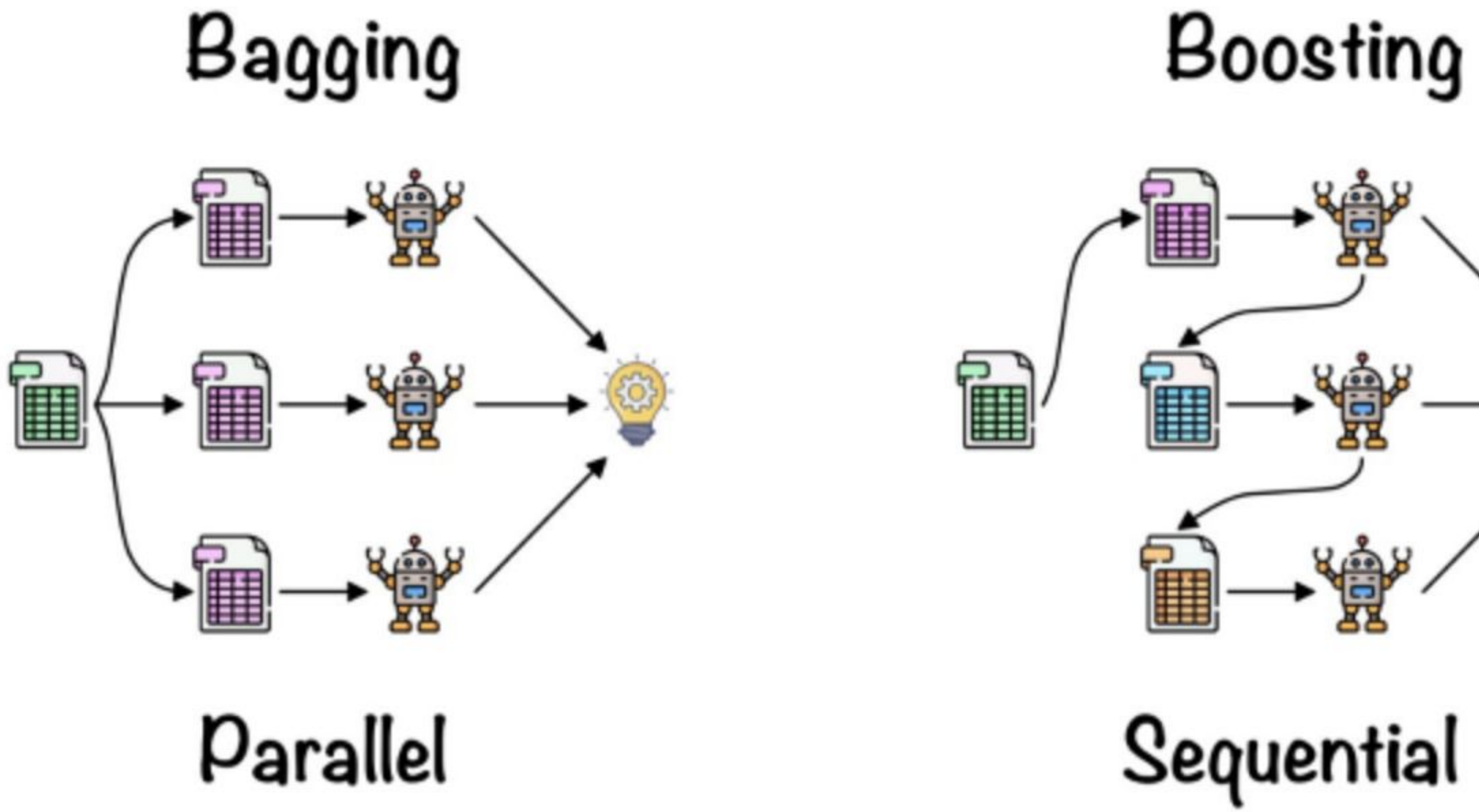
Decision Trees

- Decision trees efficiently capture intricate relationships between features and target variables, enabling interpretable models with robust performance in noisy datasets.
- Their innate ability to partition the feature space and focus on informative features allows them to represent complex decision boundaries and handle both numerical and categorical data effectively.

Classification Report:

	precision	recall	f1-score	support
0	0.54	0.55	0.54	1081
1	0.91	0.91	0.91	79845
2	0.68	0.69	0.69	19392
3	0.19	0.22	0.20	1443
accuracy			0.85	101761
macro avg	0.58	0.59	0.59	101761
weighted avg	0.85	0.85	0.85	101761

Ensemble Learning



https://www.kaggle.com/code/faridaelhusseinyy/ml-project-final#Auto-ML

Bagging

Parallel training: distribute the training of individual base models across multiple computing resources and enable simultaneous independent model training on different subsets of the data

Variance Reduction: Training several models on distinct data subsets yields an ensemble prediction that is more stable and less prone to overfitting.

Robustness: enhances robustness by combining predictions from varied models, rendering the ensemble less susceptible to outliers and noisy data points.

Model Agnosticism: utilized with any foundational learning algorithm, making it a flexible technique capable of improving the efficacy of a range of machine learning models (decision trees, neural networks, and support vector machines).

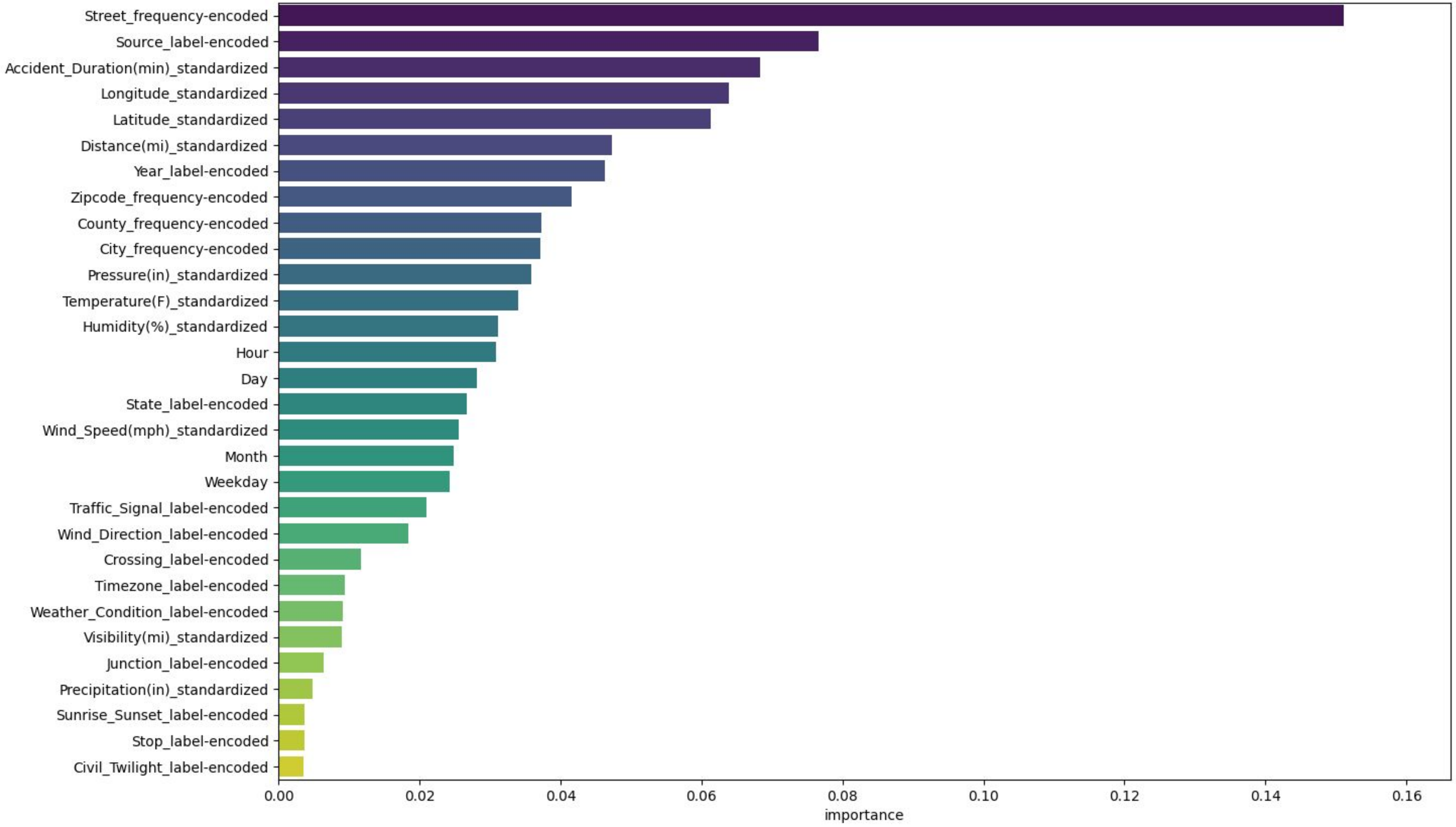
Bagging - Random Forest

- mitigate overfitting and improve generalization by aggregating predictions from multiple decision trees, offering high accuracy and robustness across diverse datasets
- ensemble approach, coupled with bagging and random feature selection, enhances model stability, scalability, and interpretability, making it suitable for handling high-dimensional data, and noisy environments.

Classification Report:

	precision	recall	f1-score	support
0	0.78	0.45	0.57	1081
1	0.91	0.96	0.94	79845
2	0.82	0.69	0.75	19392
3	0.59	0.07	0.13	1443
accuracy			0.89	101761
macro avg	0.78	0.55	0.60	101761
weighted avg	0.89	0.89	0.89	101761

Bagging - Random Forest



Boosting

Sequential Training: Each model is trained one after the other, learning specifically from the mistakes of the model before it.

Error Focus: The training focuses on instances that were previously incorrect, adjusting their importance in the next model.

Reduces Bias and Variance: Boosting starts with simple assumptions and gradually becomes more complex to improve prediction accuracy and reduce errors.

Shallow Trees for Interpretability: Unlike in bagging techniques like Random Forest where trees are fully grown, boosting uses shallow trees (trees with fewer splits), which enhances their interpretability.

Boosting - XGBoost

eXtreme Gradient Boosting enhances traditional gradient boosting methods by incorporating regularization to prevent overfitting, improving its generalization capabilities.

```
Classification Report:
              precision    recall  f1-score   support

           0       0.72      0.64      0.68      1081
           1       0.93      0.96      0.94     79845
           2       0.82      0.76      0.79     19392
           3       0.58      0.16      0.24      1443

    accuracy                           0.91    101761
   macro avg       0.76      0.63      0.66    101761
weighted avg       0.90      0.91      0.90    101761
```

Boosting - LightGBM

Light Gradient Boosting Machine optimizes performance through innovative techniques like Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB), which stream-line data processing and reduce dimensionality.

Classification Report:

	precision	recall	f1-score	support
0	0.71	0.63	0.67	1081
1	0.92	0.96	0.94	79845
2	0.81	0.74	0.78	19392
3	0.56	0.14	0.23	1443
accuracy			0.90	101761
macro avg	0.75	0.62	0.65	101761
weighted avg	0.89	0.90	0.90	101761

Boosting - CatBoost

CatBoost employs ordered boosting, a permutation-driven alternative to the classic gradient boosting method,which significantly reduces overfitting and improves the model’s accuracy.

Classification Report:

	precision	recall	f1-score	support
0	0.73	0.62	0.67	1081
1	0.92	0.96	0.94	79845
2	0.81	0.73	0.77	19392
3	0.56	0.15	0.23	1443
accuracy			0.90	101761
macro avg	0.76	0.61	0.65	101761
weighted avg	0.89	0.90	0.89	101761

Neural Networks

By combining many perceptrons, Neural Networks can learn non-linear patterns among the features for many tasks, including classification.

Expressive: Deep Neural Networks can learn very complex patterns in data. This can help the model learn feature relationships that other models may miss, but can also cause the model to easily overfit.

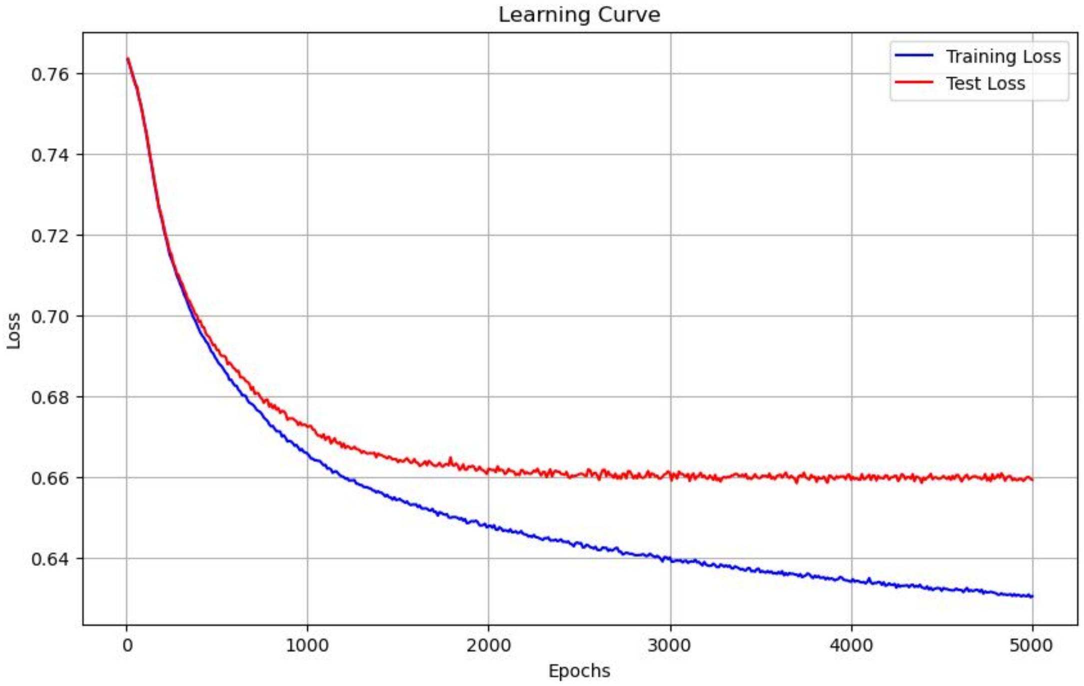
Regularization: Several techniques exist to help Neural Networks combat overfitting. Of these, Batch Normalization and Dropout were employed to allow the Neural Network to train longer without underperforming on test datasets.

Sensitive to Imbalanced Data: Neural Networks are sensitive to imbalanced data which is present in our dataset. We will see the effects of this in the results section.

Neural Networks - Model

- 2 Hidden Layers of size 128 and 64
- ReLU Activation function & Softmax output activation function
- Batchnorm and Dropout after each hidden layer for regularization.

Trained separately on full dataset and resampled dataset.



Batch Normalization and Dropout

Batch Normalization

Neural Network tend to perform better on normalized data.

A technique implemented as a layer that helps normalize the hidden layers while speeding up training.

Dropout

Sometimes, neurons can learn spurious patterns in the data called "conspiracies" that cause overfitting.

By randomly disabling neurons during learning, the model cannot rely on "conspiracies" and must learn stable and general patterns.

Doshi, K. (2021, May 29). *Batch norm explained visually-how it works, and Why Neural Networks Need it*. Medium.

Ryanholbrook. (2023, April 20). *Dropout and batch normalization*. Kaggle.

Neural Networks - Results

The Neural Network struggles to learn highly imbalanced data.

Classification Report:

	precision	recall	f1-score	support
0	0.00000	0.00000	0.00000	1081
1	0.88318	0.93917	0.91031	79845
2	0.70636	0.61391	0.65690	19392
3	0.00000	0.00000	0.00000	1443
accuracy			0.85389	101761
macro avg	0.39738	0.38827	0.39180	101761
weighted avg	0.82758	0.85389	0.83944	101761

Neural Networks - Results cont'd

Classification Report:

	precision	recall	f1-score	support
0	0.86847	0.93140	0.89883	1035
1	0.69532	0.51985	0.59492	1058
2	0.73136	0.78952	0.75933	1031
3	0.75089	0.83235	0.78952	1014
accuracy			0.76655	4138
macro avg	0.76151	0.76828	0.76065	4138
weighted avg	0.76122	0.76655	0.75958	4138

Imbalance Learning

Resampling Techniques: Adjust the number of samples from each class by either increasing the number of minority class samples or decreasing the number of majority class samples to balance the dataset.

Synthetic Data Generation: Create new, artificial samples to add to the minority class using algorithms like SMOTE, helping to balance the dataset without just copying existing samples.

Cost-sensitive Learning: Adjust the algorithm to penalize misclassifying the minority class more heavily than the majority class, encouraging better accuracy for the minority class.

Ensemble Methods: Build multiple models on different balanced subsets of the data, then combine their outputs to get a more fair and balanced result.

Imbalance Learning - SPE

The Self-paced Ensemble (SPE) technique adjusts the importance of difficult examples during training, gradually focusing more on them. It starts with simple cases and increasingly handles more complex ones, improving its accuracy over time.

```
Classification Report:
              precision    recall  f1-score   support

           0       0.31      0.94      0.47      1081
           1       0.98      0.66      0.79     79845
           2       0.51      0.89      0.65     19392
           3       0.11      0.80      0.19      1443

    accuracy                           0.71    101761
   macro avg       0.47      0.82      0.52    101761
weighted avg       0.87      0.71      0.75    101761
```

Imbalance Learning - Balanced Bagging

Balanced Bagging Classifier address class imbalance by employing balanced sampling strategies during model training, ensuring that minority classes receive adequate representation and preventing bias towards dominant classes

Classification Report:

	precision	recall	f1-score	support
0	0.29	0.94	0.44	1081
1	0.95	0.72	0.82	79845
2	0.55	0.79	0.65	19392
3	0.11	0.76	0.19	1443
accuracy			0.74	101761
macro avg	0.47	0.80	0.52	101761
weighted avg	0.85	0.74	0.77	101761

Voting

The Voting Classifier combines the predictions from the best models we got from the Ensemble Learning (**Random Forest + XGBoost + LightGBM)** and Imbalanced Learning (**SPE + Balanced Bagging**).

It predicts the class that receives the majority of votes from the combined models.

Classification Report:

	precision	recall	f1-score	support
0	0.61	0.81	0.70	1081
1	0.94	0.93	0.94	79845
2	0.78	0.81	0.80	19392
3	0.41	0.33	0.36	1443
accuracy			0.90	101761
macro avg	0.69	0.72	0.70	101761
weighted avg	0.90	0.90	0.90	101761

Results Comparison

Model	**Accuracy**	**Macro Average of F1 score**
Logistic Regression	0.84	0.38
SGD Classifier	0.83	0.39
KNN	0.81	0.39
Decision Trees	0.85	0.59
Bagging - Random Forest	0.89	0.60
Boosting - XGBoost	0.91	0.66
Boosting - LightGBM	0.90	0.65
Boosting - CatBoost	0.90	0.65
Deep Neural Network	0.85	0.39
Imblearn - Self-Paced Ensemble	0.71	0.52
Imblearn - Balanced Bagging	0.74	0.52
Voting	**0.90**	**0.70**

Conclusion

- Initially, we thought that the weather attributes would explain accident severity the most. After our analysis, it was shown that location attributes explained more of the accident severity.
- The linear models, KNN, and Deep Learning performed worse since they were not able to differentiate the rarer labels.
- Among all the models, the ensemble methods tend to perform better with boosting models outperforming bagging models.
- The best we achieved was with the Voting ensemble with 90% accuracy and 0.70 macro average on the f1-score.

Future Work (not planned to do before final report)

- Incorporate geospatial or time-series data into the models
- Improve methods for managing class imbalance
- Implement this model within a real-time accident risk prediction framework
- Investigate in detail the relationships between key factors and accident severity
- Examine the potential policy implications of this project

THANK YOU

Do you have any questions?

fu390@purdue.edu
gavinash@purdue.edu
yberg@purdue.edu
kbasu@purdue.edu