Traffic Accident Severity Prediction

Chekuri Naveen Kumar

Project Overview:

The objective of this project is to build a machine learning model that predicts the severity of traffic accidents based on various factors such as weather conditions, time of day, road conditions, and driver demographics. The predictions can be used by traffic authorities to implement targeted safety measures and reduce accident impact.

1. Who is your stakeholder?

Traffic authorities, emergency response teams, and urban planners are the primary stakeholders. They are interested in identifying high-risk conditions and implementing measures to reduce accident severity and improve response efficiency.

2. What is the problem they are trying to solve?

Traffic accidents pose a significant threat to public safety. The stakeholders aim to predict the severity of traffic accidents using machine learning, allowing for:

- Proactive measures to prevent accidents under high-risk conditions.
- Optimization of emergency response to save lives.

3. Where your dataset is from?

- Source: The dataset used in this project is publicly available from Kaggle
- Description: The dataset includes variables like time, weather, road conditions, driver demographics, and accident outcomes.

4. What models did you try, and why did you choose those models?

Models Tried:

- Logistic Regression: Baseline model for binary/multiclass classification.
- Decision Tree: To capture non-linear relationships and interpret feature importance.
- Random Forest: To handle overfitting and improve generalization.
- Gradient Boosting (XGBoost/LightGBM): For robust and accurate predictions in tabular data.
- Support Vector Machine (SVM): For high-dimensional data representation.
- K-Nearest Neighbors (KNN): To understand proximity-based relationships.

These models was chosen to compare simplicity, interpretability, and accuracy.

5. What features did you select/engineer? How did you choose those?

Selected Features:

- Time: Captures accident frequency during peak/off-peak hours.
- Weather Conditions: Correlation with road visibility and slipperiness.
- Road Type: Differences in traffic patterns across urban/rural areas.
- Driver Demographics: Identifies age or gender groups with higher risks.

Feature Engineering:

- One-hot encoding for categorical variables.
- Standard scaling for numerical features.

Features were selected based on their potential correlation with accident severity.

6. How did you evaluate the model? What evaluation metrics did you use? Why?

Evaluation Metrics:

- Accuracy: To measure overall correctness.
- Precision: Focuses on the relevance of predictions (important for emergency planning).
- Recall: Emphasizes capturing all severe cases (critical for safety interventions).
- F1 Score: Balances precision and recall for imbalanced classes.
- Confusion Matrix: Provides insights into model performance for each class.

Metrics like recall are prioritized to ensure all severe accidents are flagged.

7. What would you do differently next time or given more time what would your future work be?

Collect additional data points like:

Traffic density or speed limits, vehicle types and load, explore advanced techniques like deep learning models (e.g., LSTMs for time-series data) and deploy the model for real-time predictions using a web application.

8. Do you recommend your client use this model?

Recommendation:

LogisticRegression is : 0.84375

DecisionTreeClassifier is : 0.7337662337662337

```
SVM  is : 0.84375
KNeighborsClassifier  is : 0.8262987012987013
GNB  is : 0.8145292207792207
RandomForestClassifier  is : 0.8445616883116883
AdaBoostClassifier  is : 0.8425324675324676
GradientBoostingClassifier  is : 0.8486201298701299
```

Yes, the Gradient boosting model achieved the best balance of precision and recall.

Precision/Recall Analysis: The model consistently flags severe cases while minimizing false positives.

9. How will you deploy your model?

Deployment will involve:

Backend: Use Flask or FastAPI to serve the model.

Frontend: Interactive dashboard built with Streamlit.

Deployment Environment: Host the application on cloud platforms like AWS or Heroku.

The Final project code was uploaded in github (link)