INDUSTRIAL TRAINING - FINAL EVALUATION

ACCIDENT PREDICTION USING XAI AND SPN(SUM PRODUCT NETWORKS)

SUPERVISOR/S:
DR. BHAVESH N. GOHIL
ASSOCIATE PROFESSOR, PH. D. DOCSE - SVNIT
NISHTHA SRIVASTAVA
B.TECH., M.TECH., PH. D(PURSUING) DOCSE - SVNIT

PRESENTED BY :
RISHABH RAMESH MALOO
ADM. NO.: U20CS006

AGENDA

- **Problem statement**
  - Understanding Road Safety Challenges
  - Challenges and Limitations of Traditional Approaches
  - XAI (Explainable Artificial Intelligence)
  - SPN (Sum-Product Network)
- **Solution Proposed**
  - Revolutionizing Accident Severity Classification
  - Architecture diagram
  - About datasets
- **About Program**
  - Experiment Setup
  - Algorithm Steps and Libraries used
- **Results / Outcome**
- **My Learning / Problems Faced**
- **Future work**

PROBLEM STATEMENT - UNDERSTANDING ROAD SAFETY CHALLENGES

- **Overview:**
  - Road accidents pose a significant global problem, prompting nations to implement extensive measures to reduce fatalities and injuries.
  - Despite efforts, creating safer roads hinges on comprehending the complex nature of traffic incidents.
- **Challenges:**
  - Road safety models face difficulties due to the intricate interplay of variables.
  - Accidents exhibit non-linear relationships with various factors, complicating modeling efforts.

PROBLEM STATEMENT - UNDERSTANDING ROAD SAFETY CHALLENGES

- **Importance:**
  - Accurate understanding of accident causation crucial for effective safety management and policy implementation.
  - Machine learning (ML) offers promising avenues for analyzing accident data and identifying contributing factors.
- **Utilization of XAI and SPN:**
  - Explainable artificial intelligence (XAI) methods enhance interpretability of ML models, aiding in understanding prediction outcomes.
  - Sum-Product Networks (SPNs) are employed for developing predictive models to assess road accident severity.

CHALLENGES AND LIMITATIONS OF TRADITIONAL APPROACHES

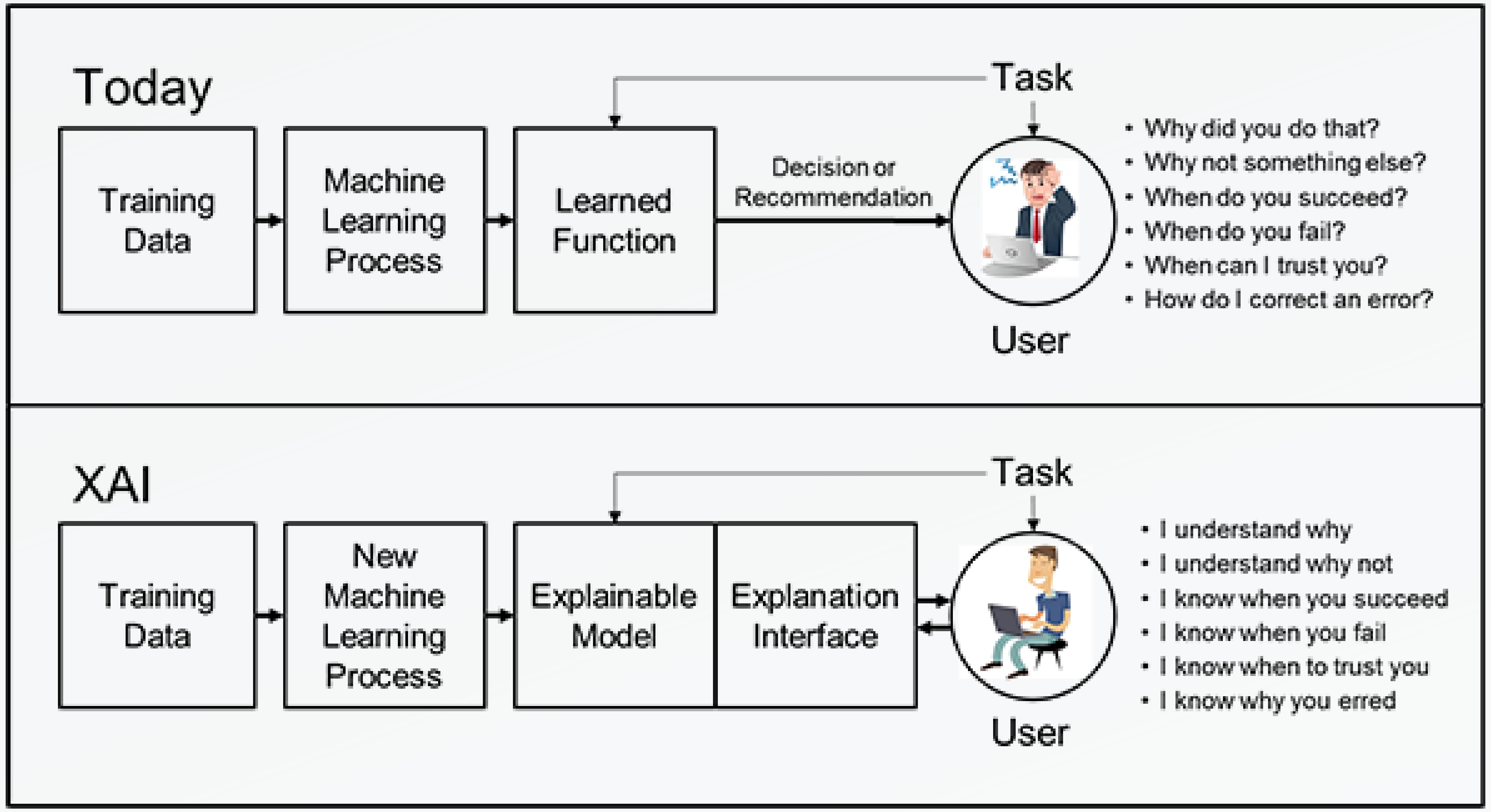
Assumption of Linearity: Traditional methods assume straight-line relationships between variables, which may not accurately capture the complexities of road safety factors.

Interpretability vs. Predictive Accuracy Trade-off: The balance between being easily understandable & accurately predicting accident outcomes, often sacrificing one for the other.

Generalization and Transferability: Traditional approaches may struggle to apply findings across different contexts, limiting their usefulness in real-world scenarios.

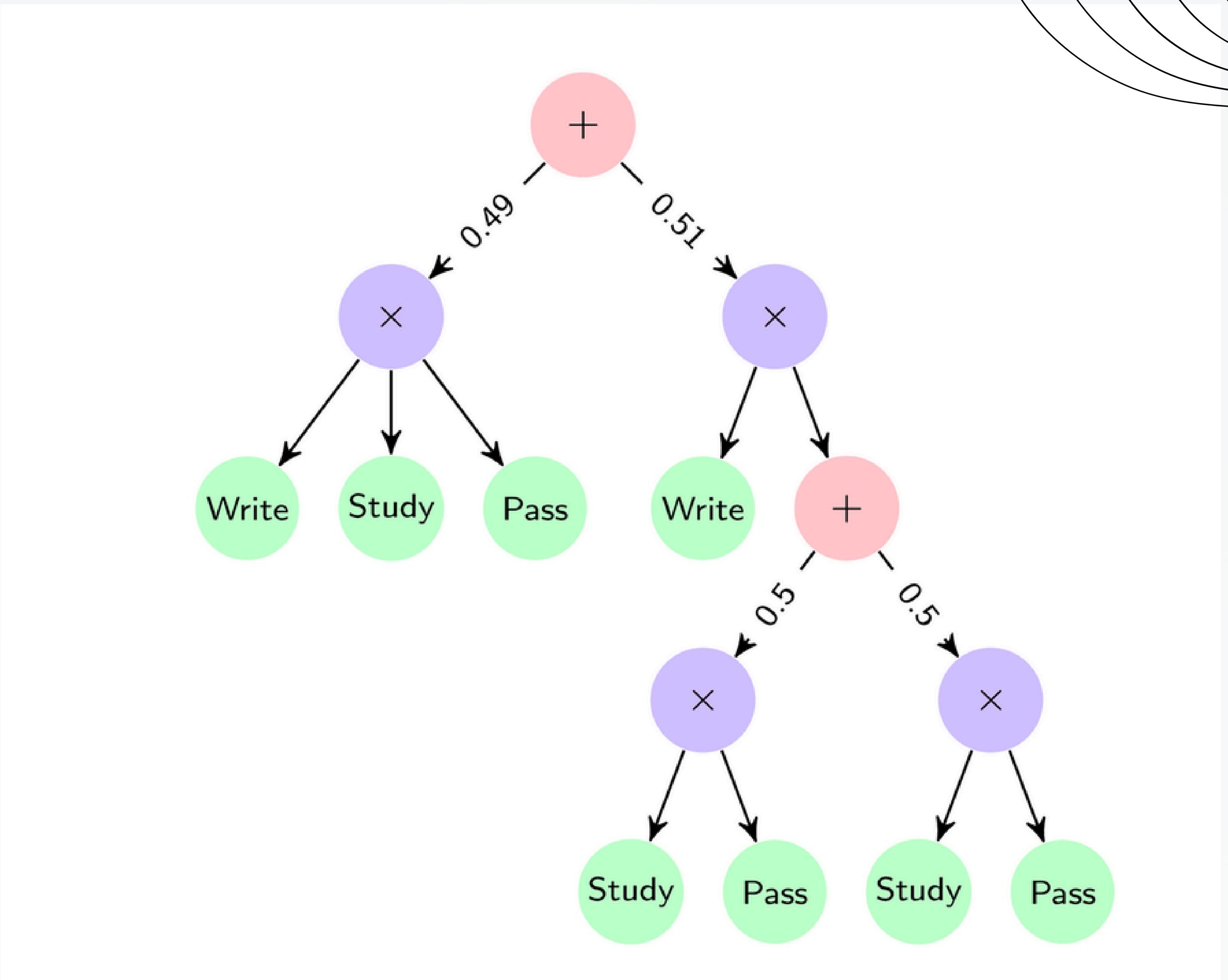
XAI (EXPLAINABLE ARTIFICIAL INTELLIGENCE)

- XAI methods enhance model interpretability, crucial for understanding ML decisions.
- Unveils black-box models, providing insights into feature importance and decision-making processes.
- Enhances trust, aids in compliance, and facilitates error diagnosis in complex ML systems.



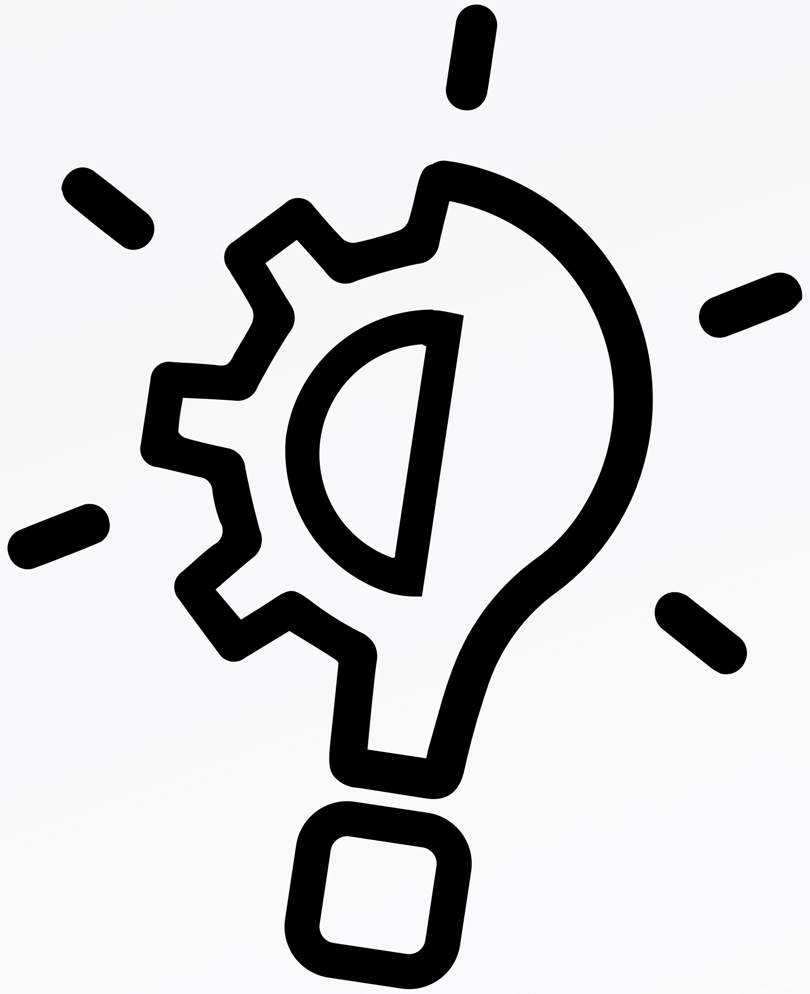
SPN (SUM-PRODUCT NETWORK)

- SPNs are probabilistic graphical models adept at capturing complex data dependencies.
- Provides a comprehensive representation of variable interdependencies, aiding in understanding complex datasets.
- Enhances classification accuracy, facilitates interpretation of model predictions, and enables efficient data analysis.

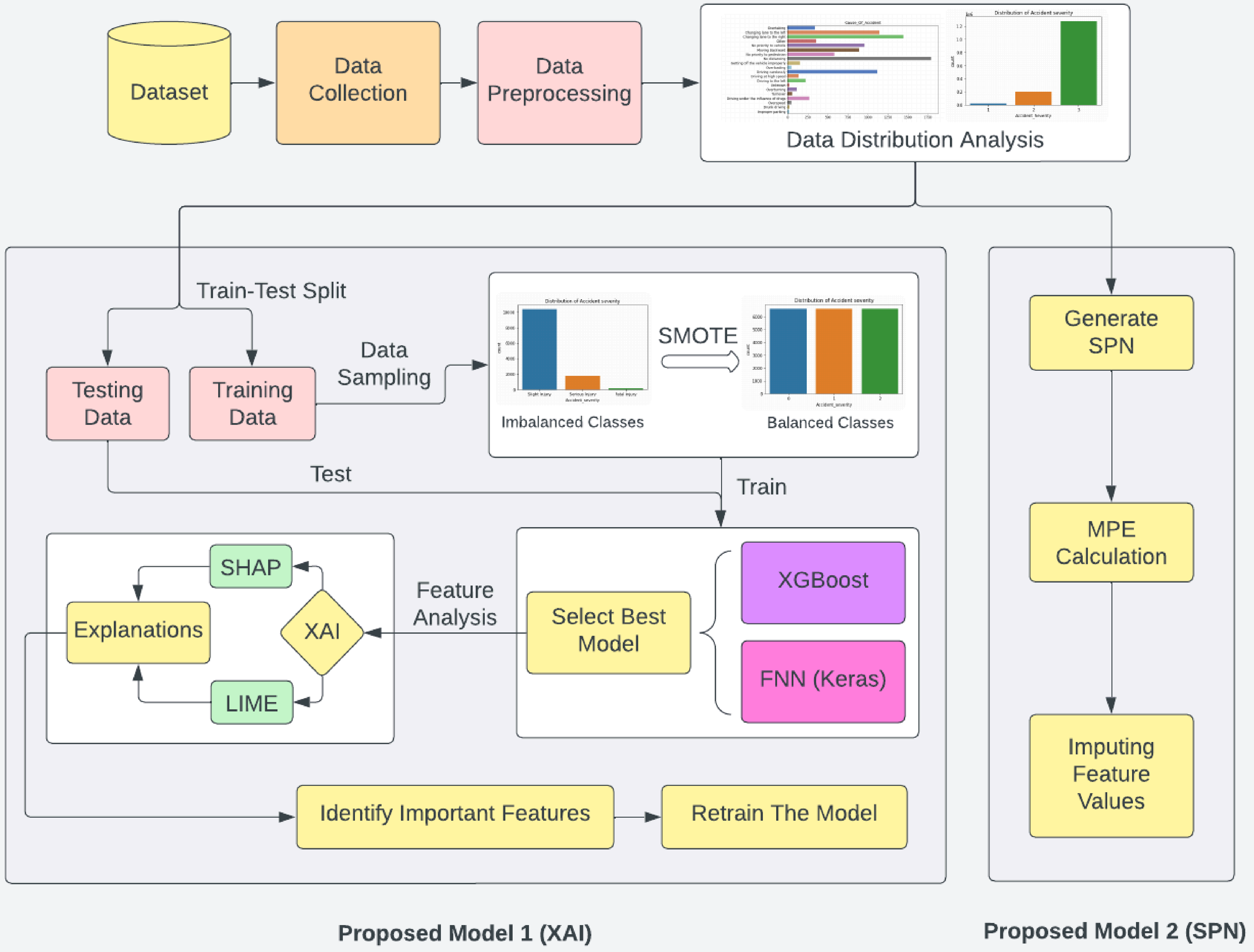


SOLUTION PROPOSED - REVOLUTIONIZING ACCIDENT SEVERITY CLASSIFICATION

- Dataset Import and Preprocessing
- Data Resampling with SMOTE
- Model Training, Evaluation and Selection
- Interpretability Analysis with SHAP and LIME
- SPN Model Development and Assessment



ARCHITECTURE DIAGRAM



RISHABH RAMESH MALOO

ABOUT DATASETS

US Traffic Accident Dataset (2016 - 2023)
- Real-time data from all US states, covering approximately 7.7 million accidents.
- High accuracy, diverse attributes for analysis.

Road Traffic Accident Dataset - Addis Ababa City (2017-20)
- Manual records from 2017 to 2020, comprising 12,316 accidents.
- Addresses public safety concerns, guides policy interventions.

The Road Accident (United Kingdom 2000-2018) Dataset
- Documenting over 1.8 million accidents.
- Comprehensive dataset with minimal inaccuracies, valuable for severity analysis.

ABOUT PROGRAM

- **Experiment Setup:**
  - Implemented on a machine with AMD A6-9220 RADEON R4, 5 COMPUTE CORES 2C+3G 2.50 GHz processor and 4GB RAM running Windows 10.

  - Programming Environment
    - Python
    - Jupyter Notebook
    - Google Colab

ABOUT PROGRAM

- **Algorithm Steps Libraries used:**

Data preprocessing / EDA

Applying SMOTE

Testing ML Models

XAI - Computing SHAP & LIME Values

Generating SPN

Most Probable Explainations

Calculating MPE

RESULTS / OUTCOME

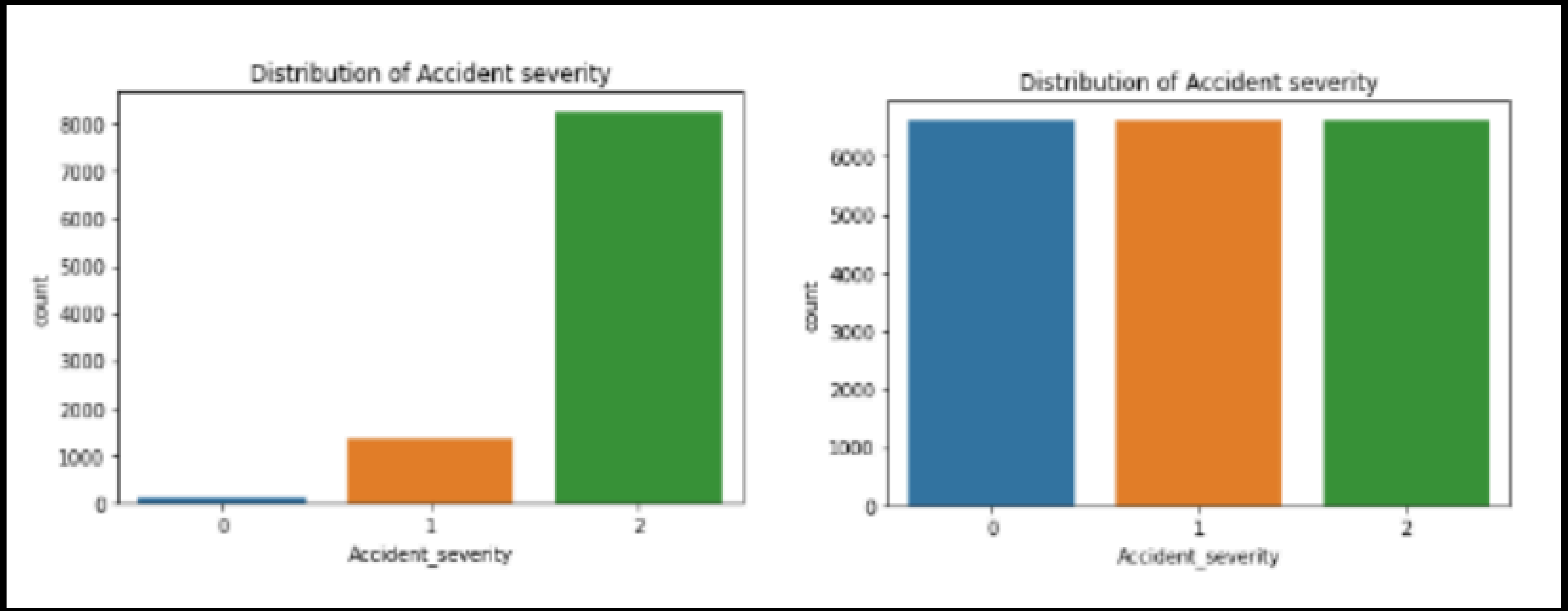
SMOTE

In [102]:
```
from imblearn.over_sampling import SMOTE
```

In [103]:
```
smote = SMOTE(random_state=42)
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)
```

In [104]:
```
print("X_train_smote: ", X_train_smote.shape)
print("y_train_smote: ", y_train_smote.shape)
```

```
X_train_smote:  (19875, 14)
y_train_smote:  (19875,)
```



CLASS DISTRIBUTION - BEFORE V/S AFTER SMOTE

RISHABH RAMESH MALOO

RESULTS / OUTCOME

ACCURACY SCORES OF SEVERAL MODELS CONSIDERED

Model	Accuracy
XGBoost (normal)	85.08%
XGBoost (SMOTE-resampled data)	91.59%
Keras/Tensorflow (normal)	79.45%
Keras/Tensorflow (SMOTE-resampled data)	84.62%

RESULTS / OUTCOME

```
# Create a SHAP explainer object for the trained XGBoost model
explainer = shap.TreeExplainer(model)

X_test_sample = X_test[:1000]

start_time = time.time()
shap_values = explainer(X_test_sample)
end_time = time.time()

elapsed_time = end_time - start_time
print("Time taken for calculating SHAP values:", elapsed_time, "seconds")
```

```
Time taken for calculating SHAP values: 5.733771800994873 seconds
```

CALCULATING SHAP VALUES

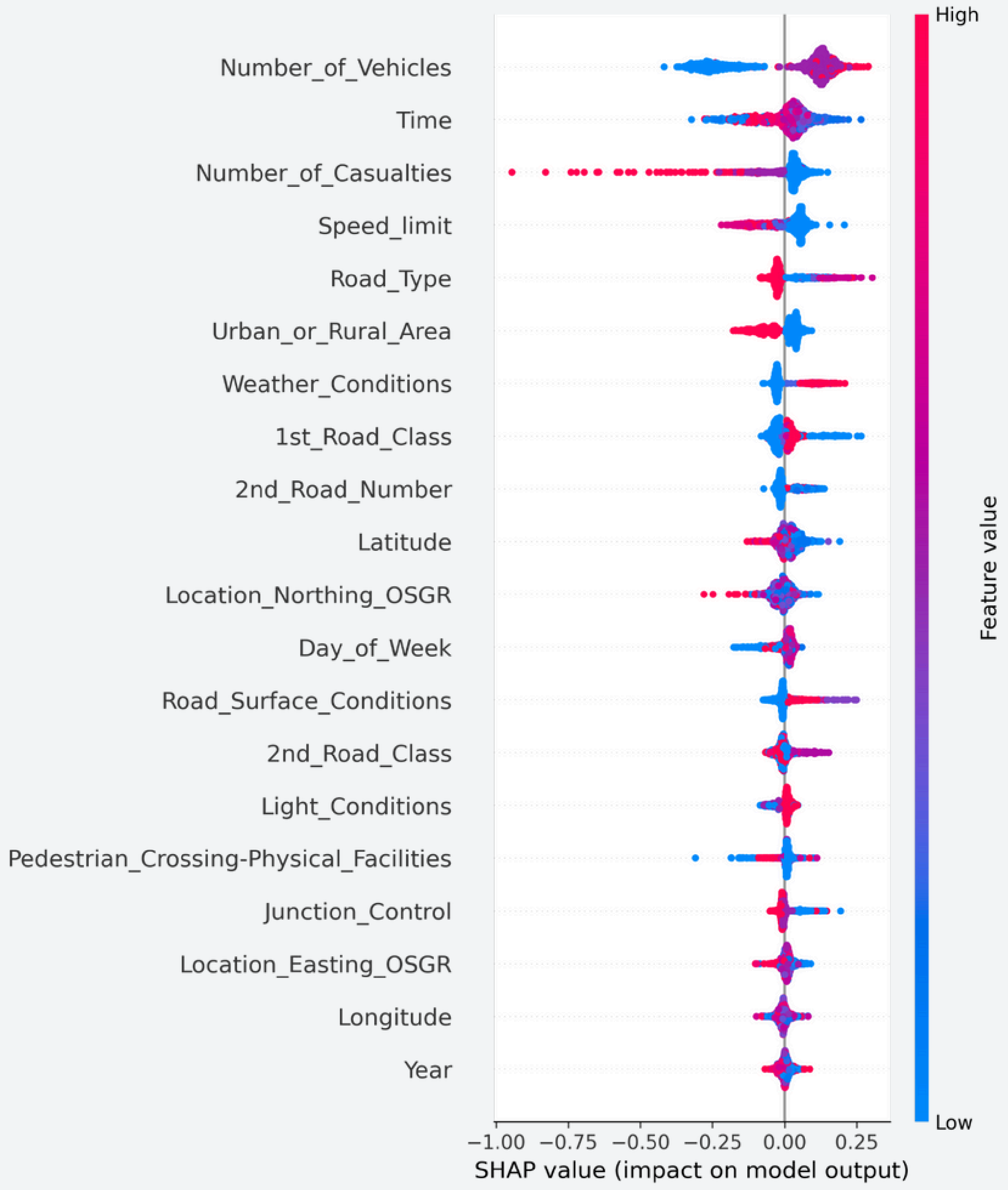
```
from lime import lime_tabular

explainer_lime = lime_tabular.LimeTabularExplainer(X_train.values, feature_names=column_names, class_names=['0', '1', '2'], verbose=True, mode='classification')
```

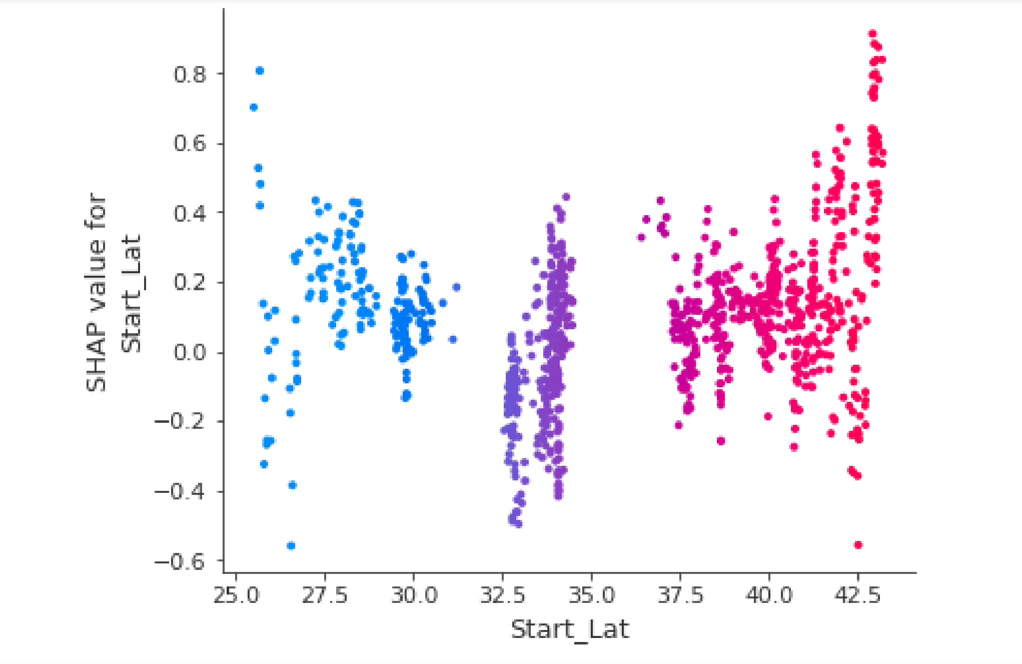
CALCULATING LIME VALUES

RESULTS / OUTCOME

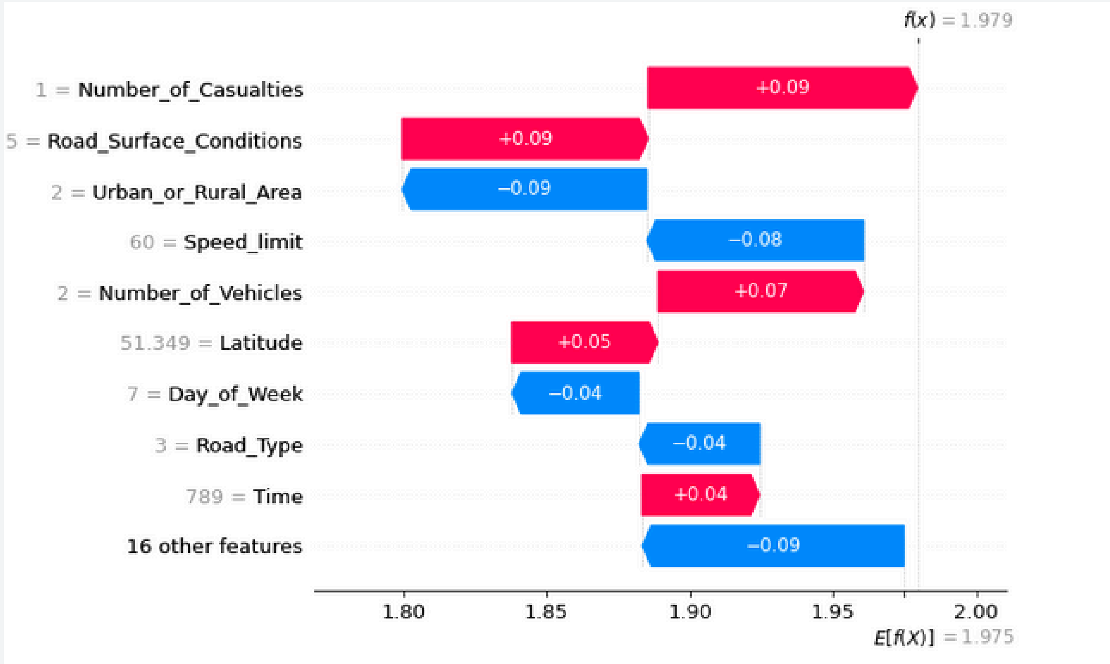
SHAP PLOTS



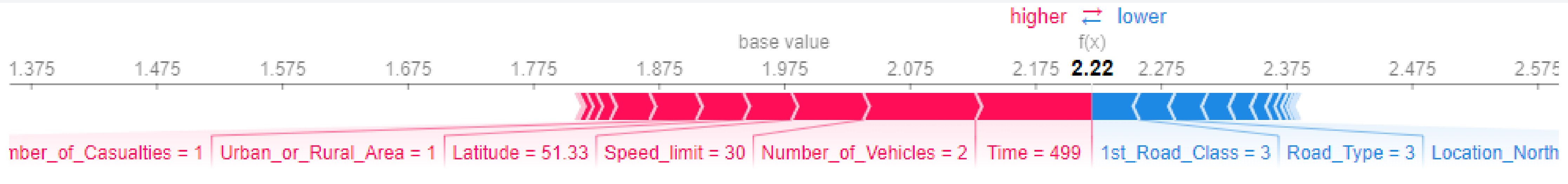
THE SUMMARY PLOT



THE DEPENDENCE PLOT



THE WATERFALL PLOT

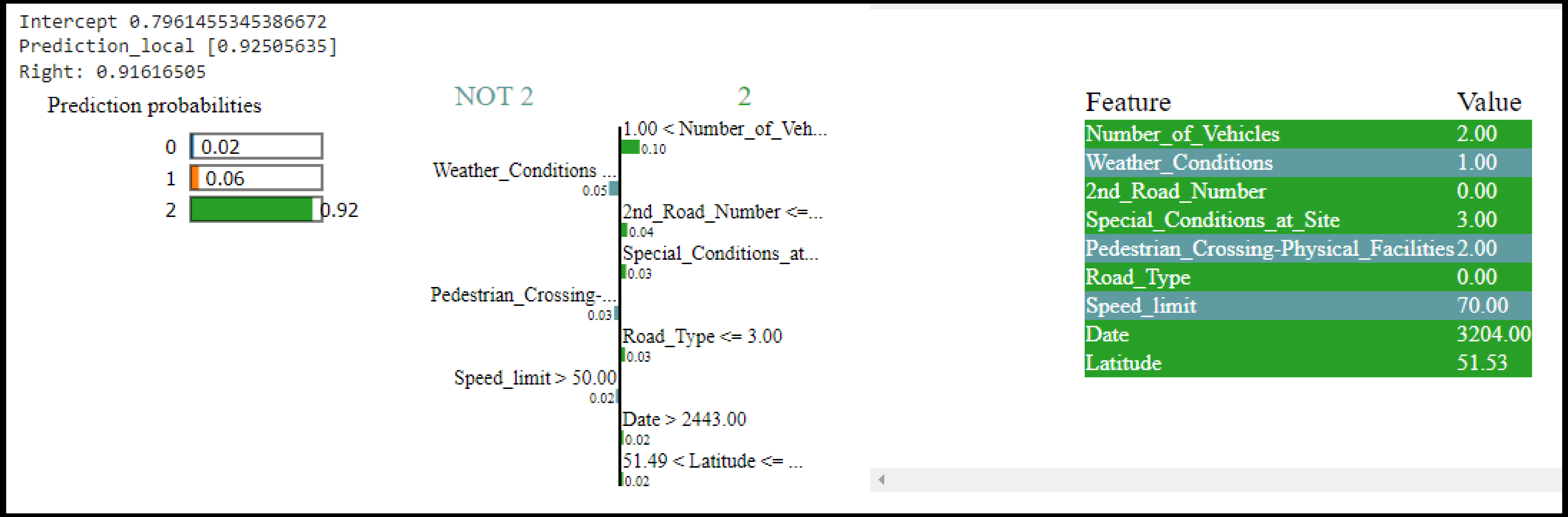


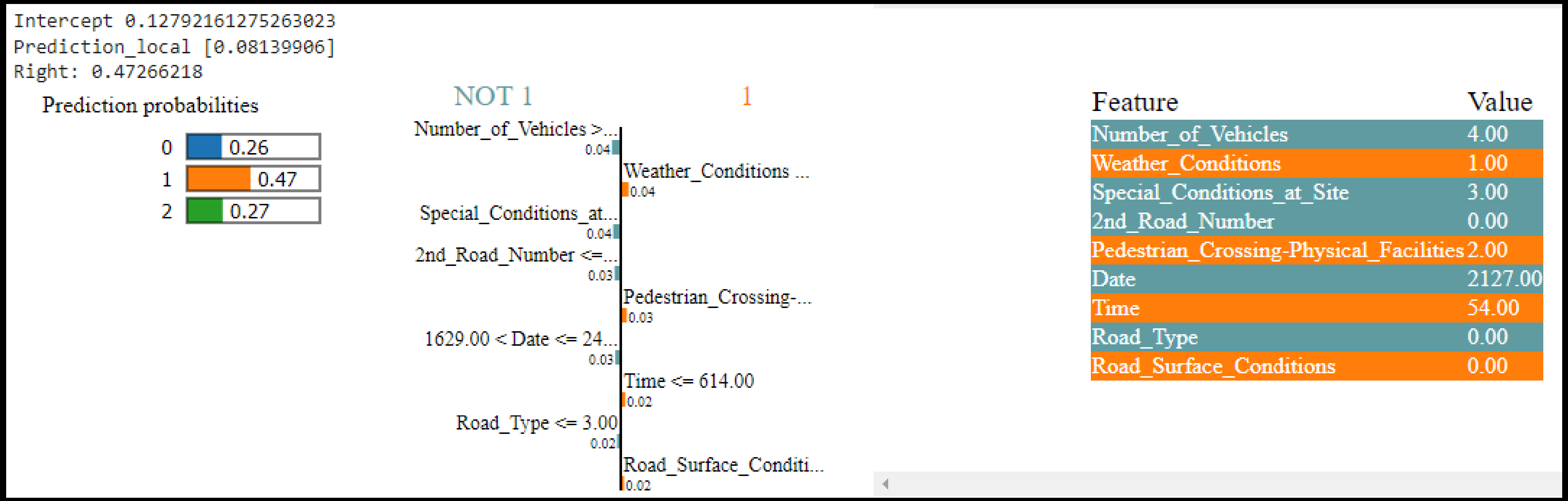
THE FORCE PLOT

RISHABH RAMESH MALOO

RESULTS / OUTCOME

LIME PLOTS





RISHABH RAMESH MALOO

RESULTS / OUTCOME

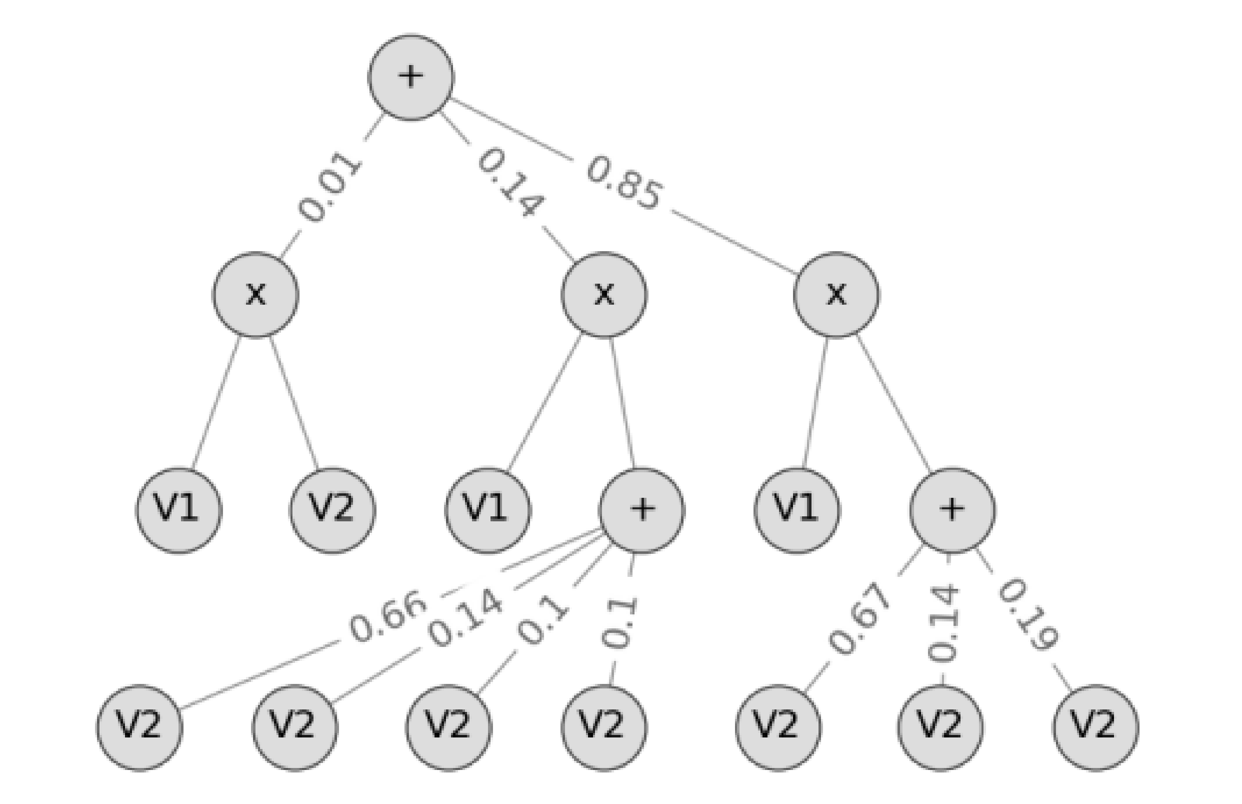
```
[67] start_time = time.time()

     # Create SPN
     spn_classification = learn_classifier(data=df.values, ds_context=spn_context, label_idx=label_idx, spn_learn_wrapper=learn_parametric)

     end_time = time.time()
     # Calculate the elapsed time
     elapsed_time = end_time - start_time
     print("Elapsed time:", elapsed_time, "seconds")

     print(spn_classification)

     Elapsed time: 30.384788513183594 seconds
     SumNode_0
```



SUM-PRODUCT NETWORK (SPN) CONSTRUCTION

RESULTS / OUTCOME

```
# Generate the test_classification array from the DataFrame df
test_classification = df.values.astype(float)  # Convert to float to support NaN values

# Find the index of the column named "Accident_severity"
accident_severity_index = df.columns.get_loc("Accident_severity")

# Add NaN values to the test instances at the index corresponding to "Accident_severity"
test_classification[:, accident_severity_index] = np.nan

test_classification
```

```
array([[407.,   1.,   1., ...,   5.,  16.,  nan],
       [575.,   3.,   0., ...,   5.,   1.,  nan],
       [425.,   0.,   0., ...,   5.,  12.,  nan],
       ...,
       [242.,   3.,   2., ...,   5.,   1.,  nan],
       [242.,   3.,   0., ...,   5.,   5.,  nan],
       [242.,   3.,   0., ...,   1.,   1.,  nan]])
```

```
# Compute MPE
mpe_result = mpe(spn_classification, test_classification)

print(mpe_result)
```

```
[[407.   1.   1. ...   5.  16.   2.]
 [575.   3.   0. ...   5.   1.   2.]
 [425.   0.   0. ...   5.  12.   2.]
 ...
 [242.   3.   2. ...   5.   1.   2.]
 [242.   3.   0. ...   5.   5.   2.]
 [242.   3.   0. ...   1.   1.   2.]]
```

Feature	Accuracy of calculated MPE
Road surface type	0.9322
Casualty severity	0.9320
Sex of driver	0.9308
Pedestrian movement	0.9244
Accident severity	0.8794

ACCURACY SCORES OF SEVERAL FEATURES' MPES

MY LEARNING / PROBLEMS FACED

- **Learning:**
  - Acquired proficiency in applying machine learning and explainable AI techniques to road safety analysis.
  - Gained insights into the importance of data preprocessing and feature engineering for model performance.
  - Developed skills in interpreting and visualizing model predictions using XAI methods.

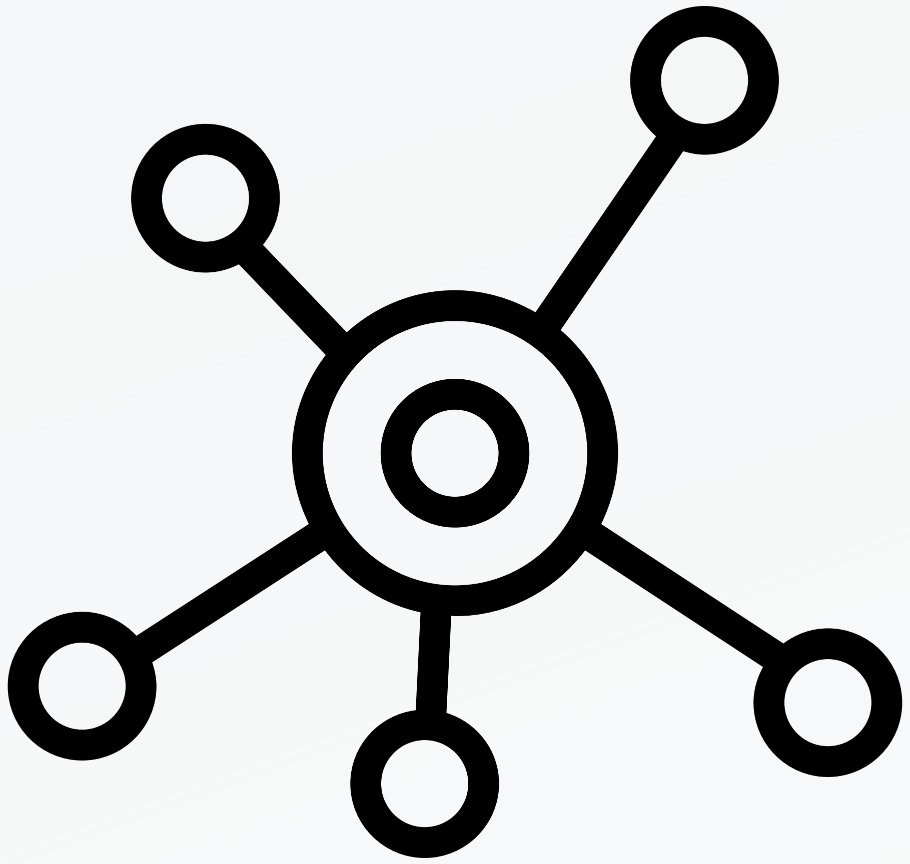
- **Challenges Faced:**
  - Data preprocessing complexities: Dealing with missing values, outliers, and imbalanced datasets.
  - Model selection dilemma: Evaluating various ML algorithms and choosing the most suitable for the task.

MY LEARNING / PROBLEMS FACED

- **Lessons Learned:**
  - Importance of domain knowledge: Understanding traffic safety principles and factors influencing accident severity.
  - Iterative approach to model development: Experimenting with different techniques and refining strategies based on feedback.
  - Collaboration and communication: Engaging with experts in the field to address challenges and validate findings.

FUTURE WORK

- Refining Algorithm Accuracy
- Incorporating SHAP and LIME plots for deeper interpretability and refinement of model predictions.
- Further refining SPNs to capture complex variable dependencies more accurately.
- Investigating causal inference methods to better understand the relationships between features and accident severity.



THANK YOU