Machine Learning Approaches for Analyzing Road Traffic Accidents in Pakistan

Sibgat Ul Islam
ID: 2111920642
ECE, North South University
Bashundhara, Dhaka
sibgat.islam@northsouth.edu

Imroz Rahim
ID: 2011292042
ECE, North South University
Bashundhara, Dhaka
imroz.rahim@northsouth.edu

Iffat Ara Mehnaz
ID: 2031340042
ECE, North South University
Bashundhara, Dhaka
iffat.mehnaz@northsouth.edu

***Abstract*—Road traffic accidents (RTAs) are a leading cause of injury and death worldwide, particularly in developing countries like Pakistan, where road infrastructure and traffic management face significant challenges. Despite their prevalence, limited research exists on systematically analyzing RTA data to deduce intervention and prevention strategies, especially in developing countries where such incidents are rarely reported, if not ignored. This study analyzes patterns of RTA injury types and patient status reported in the city of Rawalpindi-Punjab using standard machine learning techniques along with a custom Neural Network. Our results indicate critical risk factors and determine patient status along with the types of injuries that occurred. The results indicate that XGBoost (accuracy: 0.7299, F1 score: 0.7268 for patient status, and accuracy: 0.7598, F1 score: 0.7061 for injury type) and Multi-class ANN (accuracy: 0.7197, F1 score: 0.7170 for patient status, and accuracy: 0.7690, F1 score: 0.6992 for injury type) consistently outperformed other models, highlighting their suitability for analyzing RTA data.**

I. Introduction

Injuries related to RTA incidents are a global public health problem. Despite advancements in road safety, they remain a leading cause of mortality and morbidity, especially in low-income regions of the world. RTAs account for 90 percent of road fatalities [1]. Identification of primary factors of road accident severity is required to minimize the severity of accidents.

RTAs not only result in immediate loss of life but also cause permanent disabilities along with long-term consequences. These consequences include but are not limited to disabilities A better analysis of non-fatal RTA cases is instrumental for injury prevention and control measures.

In this study we applied various ML algorithm to analyze and predict the fatality of these incidents with the goal to get the most accurate predictions. Our aim was to tune the algorithms in such a way that these algorithms may be used as a real-world applicable tool which may help to reduce the time it takes for the hospitals to respond and hospitals may be ready for the incoming patients, caught in the incident.

II. Methodology

In this study, we have trained various machine learning models and also implemented an Artificial Neural Network, in order to predict the type of injury and the status of the patient, if a similar kind of accident is to occur again, which might help the emergency teams to response more appropriately to treat the patient.

This study also shows the effect on the performance of different hyper-parameters applied on the machine learning algorithms and the ANN model.

A. RTA Data 2020 to July 2023 Dataset

The dataset[2] curated by the author *M Shujaat Abid* was of based on accidents that occurred in Pakistan over three years.

1) Description: The RTA Dataset[2] which we were dealing with had 23 columns, covering reason of the accident occurrence, the place of the occurrence, the patient's status after the accident also the kind of injury the victim was affected by. The dataset[2] also had information about the time of the occurrence, the type of vehicles (if there were any) involved on the accident, call time and the response time of the hospital which were called after the accident. The dataset contains 46189 of samples and 25 columns.

2) Exploratory Data Analysis (EDA): Before applying pre-processing, we performed EDA on the dataset[2], to see what kind of data we were dealing with, to get more insights of the data and to plan the we were going to handle the dataset[2].

- **Data types**: The dataset[2] had primarily integer and floating point data with some textual or object data. Among the textual data, there was reason column, which described the reason of accident.
- **Pattern of the data**: On our EDA, we discovered that the data was extremely unbalanced for our target variables. The dataset
- **Anomalies**: The dataset had few samples mixed with numerical and textual data also unrelated information. Also we noticed that there were some samples which had "same" as reason, in "Reason" column. Our investigation further told us that, the dataset had mixed language (Urdu and English). Also there were 8 missing values which were later handled by us.
- **Target Distribution**: The dataset is largely unbalanced. The table of the distribution shows us that:

TABLE I
DISTRIBUTION OF PATIENT STATUS AND INJURY TYPES

Category	Proportion
Patient Status	
Alive & unstable	51%
Alive & stable	47%
Dead	1.27%
Injury Type	
Minor	73.15%
Single Fracture	14.58%
Head Injury	7.58%
Multiple Fractures	1.68%
Spinal Injury	0.82%

*3) **Pre-processing**:* Before applying the machine learning models, the dataset was preprocessed to ensure data quality and compatibility with the models. The pre-processing steps included:

- **Handling Missing Values**: Missing values were addressed by [describe the method used, e.g., imputation with mean/median values, removal of missing entries, etc.].
- **Feature Scaling**: Feature scaling was applied to ensure uniformity across all features and to avoid bias in the algorithms. Features were scaled using several scaling methods.
  - **Standardization**: Standardization was used to scale continuous features to have a mean of 0 and a standard deviation of 1.
  - **Min-Max Normalization**: Min-Max Normalization was applied for certain features, to scale values within the range of [0,1].
- **Categorical Encoding**: Categorical features were encoded to numerical format in order to make them compatible with our models.
  - **Label Encoding**: Label Encoding was applied to categorical columns such as InjuryType, Cause, EducationTitle, PatientStatus, CallHour, and CallMonth.
  - **One Hot Encoding**: One Hot Encoding was applied to the Gender column to create binary columns.
- **Data Splitting**: The dataset was split into training and testing sets with a ratio of [mention the ratio, e.g., 80:20] to evaluate the model performance.
- **Text Processing**: The Reason column, which contained textual data, underwent several preprocessing steps using the NLTK library:
  - **Tokenization**: Text was tokenized into individual words.
  - **Lowercasing**: All text was converted to lowercase to ensure uniformity.
  - **Removing Non-Alphabetic Characters**: Non-alphabetic characters were removed using regular expressions.
  - **Stopword Removal**: Common stopwords (e.g., ”and”, ”the”) were removed to focus on the meaningful words.
  - **Lemmatization**: Words were lemmatized using the WordNet Lemmatizer to reduce words to their base form (e.g., ”running” to ”run”). The cleaned and lemmatized tokens were stored in a new column *reason_tokenized*, and a deduplicated, comma-separated string version of the tokens was saved in the Reason column.
- **Word Embedding Vectorization**: Word embedding vectorization was done using Word2Vec[3] embeddings to convert tokenized text into numerical representations.
  - The pre-trained Word2Vec[3] model was loaded with *gensim*[4] to vectorize each tokenized list of words in the *reason_tokenized* column by averaging Word2Vec vectors of individual words.

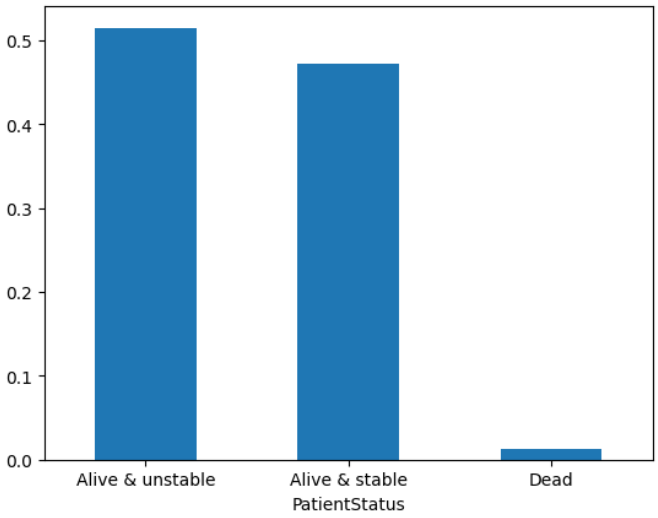

Fig. 1. Distribution of Patient Status

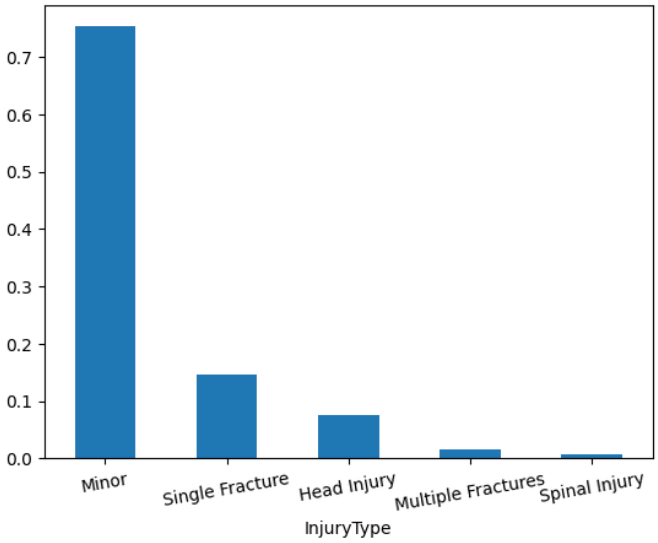

Fig. 2. Distribution of Injury Type

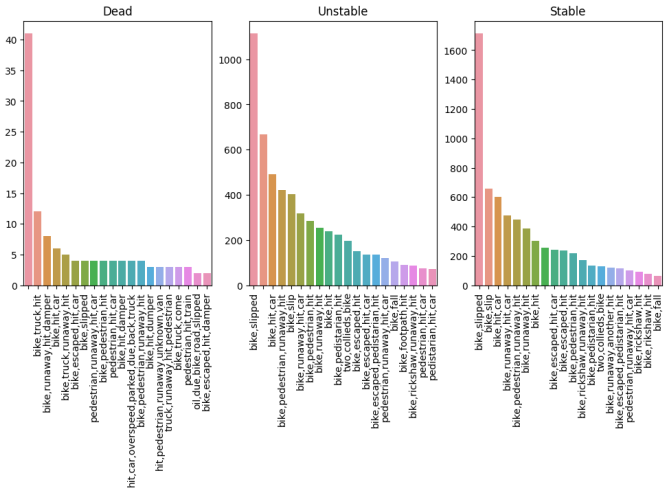

Fig. 3. Patient Status by Vehicle Involvement

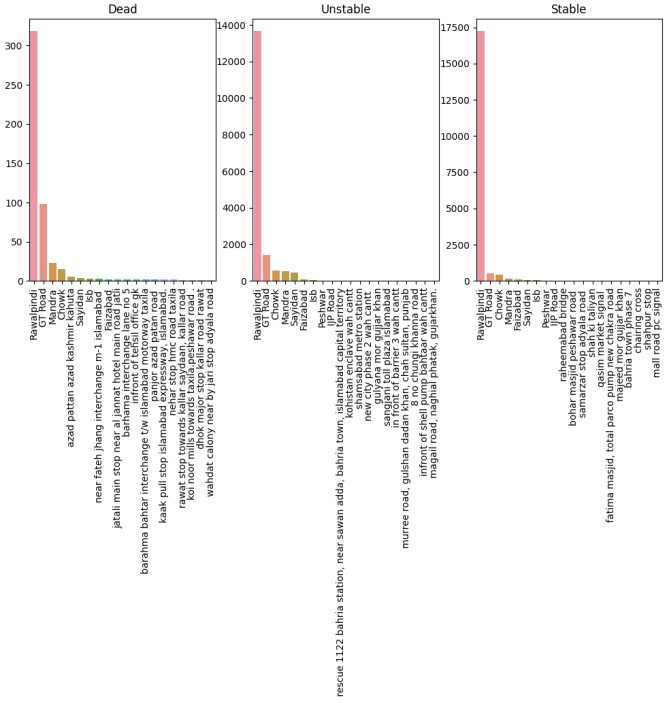

Fig. 4. Patient Status by Area

  - After vectorization, rows where the vector length was not equal to 300 were dropped. This ensured the consistency of vector dimensions. The index of the dataset was reset after filtering to maintain a clean structure.
- **Feature Reduction**: To optimize the dataframe for the selected machine learning models, we performed feature reduction to focus on the most relevant columns. This involved dropping several columns that were unnecessary for analysis and to reduce dimensionality.
  - **TotalPatientsInEmergency**: This column was removed as it did not provide any significant context to the predictive modeling process. The total number of patients in the emergency room did not reflect any direct understanding of individual accident cases.
  - **Reason**: The original reason column was transformed into a numerical format using tokenization and vectorization. Once transformed, the original column became redundant thus it was dropped to avoid unnecessary complexity.
  - **Response Time**: Similar to TotalPatientsInEmergency, the ResponseTime column did not provide any significant insights and was deemed unnecessary to our analysis.
  - ***reason_tokenized***: This column contained raw tokens of the Reason text data. After vectorizing the word list into 300-dimensional numerical format and expanding it into separate columns, the `reason_tokenized` itself was redundant and was dropped.
- **Final Dataset**: After completing the pre-processing steps, the dataframe was ready for machine learning. The final dataframe included:
  - Encoded categorical features.
  - Cleaned and vectorized text data.
  - Scaled and normalized numerical features.
  - A reduced set of columns for efficient modeling.

B. Models

We used 5 different predefined classifier models from Scikit learn[5] and One our defined ANN model. Those are:

- Decision Tree Classifier
- Support Vector Machine Classifier
- Random Forest Classifier,
- XGBoost Classifier
- Logistic Regression
- Our-defined Artificial Neural Network Classifier model.

1) Overview: Here we present the a simplified overview of the models that we used. We Have also included the mathematical equations where possible, in below:

- **Decision Tree Classifier**: The Decision Tree algorithm recursively splits the data into subsets based on the value of the input features. The decision tree model is defined by:

$$h(x) = \sum_{i=1}^{N} w_i \cdot I(x \in R_i)$$

  where R_i are the regions defined by the decision nodes, and w_i are the prediction values.
- **Support Vector Machine Classifier**: SVM aims to find a hyperplane that best separates the data into different classes. The decision function is given by:

$$f(x) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i \langle x_i, x \rangle + b\right)$$

  where α_i are the Lagrange multipliers, y_i are the class labels, $\langle x_i, x \rangle$ represents the kernel function, and b is the bias term.

- **Random Forest Classifier**: Random Forests Classifier are an ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes (classification) or mean prediction (regression) of the individual trees. The model is defined as:

$$h(x) = \frac{1}{T}\sum_{t=1}^{T} h_t(x)$$

where T is the number of trees, and $h_t(x)$ is the prediction of the t-th tree.

- **XGBoost Classifier**: XGBoost Classifier is an optimized gradient boosting algorithm that builds additive models in a forward stage-wise manner. The model can be expressed as:

$$y_i = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in \mathcal{F}$$

where K is the number of trees, f_k is a function in the functional space $\mathcal{F}$ defined by the decision trees.

- **Logistic Regression**: Logistic Regression models the probability of a binary outcome using a logistic function. The model is defined by the equation:

$$P(y = 1|x) = \sigma(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p)$$

where $\sigma(z) = \frac{1}{1+e^{-z}}$ is the logistic function, β_0 is the intercept, and $\beta_1, \beta_2, \ldots, \beta_p$ are the coefficients for the predictors.

- **Artificial Neural Network**: The Multi-Target Neural Network (MultiTargetNN) is a fully connected feedforward neural network designed for classification tasks with multiple target outputs. The architecture consists of several layers with the following structure:
  - **Input and Output Layer**: An input layer with input_dim features and an output layer with out_class neurons for the final classification.
  - **Hidden Dimensions**: Three hidden layers, each with hidden_dim neurons.
  - **Activations**: ReLU[6] activation functions were used in between the layers.
  - **Dropout**: A dropout[7] layer with a dropout rate of 0.15 to prevent overfitting.
  - **Loss Function**: The cross-entropy loss, also known as log loss, is used as a performance measurement which basically gives out probabilities (values between 0 to 1) of a classification model's output. The logits (output of the last layer) of the model is in range from -infinity to +infinity and the cross entropy loss scales it in between 0 to 1. The formula for cross-entropy loss L is given by:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\log(\hat{y}_{i,c})$$

where:

* N is the number of samples.
* C is the number of classes.
* $y_{i,c}$ is a binary indicator (0 or 1) if class label c is the correct classification for sample i.
* $\hat{y}_{i,c}$ is the predicted probability that sample i belongs to class c.

*2) **Hyper-parameters**:* In this study, we evaluated the performance of five different pre-defined machine learning models[5] and one ANN model. Each model was tested with a range of different hyper-parameters to optimize their performance. The models and their corresponding hyper-parameters are as follows:

- **Decision Tree**: We used the Decision Tree Classifier with varying maximum depths of 4, 5, 6, 7, 8, and 9 to control the complexity of the model.
- **Support Vector Machine (SVM)**: The SVM model was tested with different kernel functions such as linear, polynomial (poly), radial basis function (rbf), and sigmoid kernels.
- **Random Forest**: The RandomForestClassifier was evaluated using different numbers of estimators, specifically 10, 50, 100, and 200, to see the impact of the number of trees in the forest.
- **XGBoost**: The XGBoost model was tuned with various learning rates of 0.001, 0.01, 0.1, and 1 to investigate the effect of the learning rate on model performance.
- **Logistic Regression**: The LogisticRegression model was tested with different solvers, including lbfgs, liblinear, newton-cg, newton-cholesky, sag, and saga, as well as varying maximum iterations (max-iter) of 50, 100, 150, 200, 250, and 300 to ensure convergence.
- **Multi-Target Neural Network**: The MultiTargetNN was trained using various hyperparameters to optimize its performance:
  - **Epochs**: The number of training epochs was set to 50, 100, and 150 to evaluate the whether the longer or shorter training helps more.
  - **Learning Rate (lr)**: Different learning rates of 0.1, 0.01, 0.15, and 0.001 were adjusted to see if different learning rates had any effect.
  - **Batch Size**: The model was trained with batch sizes of 32, 64, and 128 to understand the effect of batching on model's performance
  - **Hidden Dimensions**: The hidden layers were configured with 64, 128, and 256 neurons to analyze the impact of model complexity on the ability to capture data patterns.

III. Training and Evaluation

Our training pipeline involves splitting data in most efficient and effective way possible. Because our dataset[2] was hugely imbalanced (Fig. 1, Fig. 2), we made sure to *stratify* the target columns and split them in 75% as training and 25% as validation set using splitter from sklearn[5]. The purpose of this process was to make sure that our training set had followed the existing distribution for different classes of the target; Otherwise, during the split there is a chance that classes with lower distribution may never make it to the training set, thus the model will never learn anything about it.

We wrote our pipeline so that, every model gets to train on the same train split, for the same classes, thus avoiding split-time discrimination, to get fair results on every model.

Also, our ANN model was especially modified so that, it follow the sk-learn model's [5] convention and trained all of the model on the same loop, to ensure fair results.

During the evaluation stage, our test-split was also stratified, so it had all of the classes including the lower distribution classes, so the model had all of the classes in it's prediction pipeline and our model's scores on IV shows that.

IV. Results

All of the results we show in the table is the probability between 0 to 1, Before converting it in 0%-100% scale

TABLE II
Model Performance with Default Parameters on Patient Status

Model	Target	Accuracy	Precision	Recall	F1 Score
MulticlassANN	PatientStatus	0.72	0.72	0.72	0.72
Decision Tree	PatientStatus	0.63	0.63	0.63	0.63
SVM	PatientStatus	0.53	0.54	0.53	0.43
Random Forest	PatientStatus	0.7	0.7	0.7	0.7
XGBoost	PatientStatus	0.73	0.73	0.73	0.73
LogisticRegression	PatientStatus	0.7	0.71	0.7	0.68

TABLE III
Model Performance with Default Parameters on Injury Type

Model	Target	Accuracy	Precision	Recall	F1 Score
MulticlassANN	InjuryType	0.77	0.70	0.77	0.68
Decision Tree	InjuryType	0.67	0.68	0.67	0.68
SVM	InjuryType	0.76	0.58	0.76	0.66
Random Forest	InjuryType	0.75	0.66	0.75	0.69
XGBoost	InjuryType	0.76	0.70	0.76	0.71
LogisticRegression	InjuryType	0.76	0.58	0.76	0.66

TABLE IV
Parameters for Tuned Our MulticlassANN on Patient Status

Model	Target	Parameters	Accuracy	Precision	Recall	F1 Score
MulticlassANN	PatientStatus	{'epoch': 50}	0.71	0.71	0.71	0.71
MulticlassANN	PatientStatus	{'epoch': 100}	0.71	0.73	0.71	0.70
MulticlassANN	PatientStatus	{'epoch': 150}	0.72	0.73	0.72	0.72
MulticlassANN	PatientStatus	{'lr': 0.1}	0.51	0.26	0.51	0.34
MulticlassANN	PatientStatus	{'lr': 0.01}	0.70	0.70	0.70	0.69
MulticlassANN	PatientStatus	{'lr': 0.15}	0.51	0.26	0.51	0.34
MulticlassANN	PatientStatus	{'lr': 0.001}	0.71	0.75	0.71	0.69
MulticlassANN	PatientStatus	{'batch_size': 32}	0.72	0.72	0.72	0.71
MulticlassANN	PatientStatus	{'batch_size': 64}	0.71	0.71	0.71	0.71
MulticlassANN	PatientStatus	{'batch_size': 128}	0.72	0.72	0.72	0.72
MulticlassANN	PatientStatus	{'hidden_dim': 64}	0.72	0.73	0.72	0.72
MulticlassANN	PatientStatus	{'hidden_dim': 128}	0.72	0.72	0.72	0.72
MulticlassANN	PatientStatus	{'hidden_dim': 256}	0.72	0.72	0.72	0.72

TABLE V
PARAMETERS FOR TUNED OUR MULTICLASSANN ON INJURY TYPE

Model	Target	Parameters	Accuracy	Precision	Recall	F1 Score
MulticlassANN	InjuryType	{'epoch': 50}	0.77	0.65	0.77	0.68
MulticlassANN	InjuryType	{'epoch': 100}	0.77	0.64	0.77	0.68
MulticlassANN	InjuryType	{'epoch': 150}	0.77	0.71	0.77	0.68
MulticlassANN	InjuryType	{'lr': 0.1}	0.76	0.58	0.76	0.66
MulticlassANN	InjuryType	{'lr': 0.01}	0.76	0.58	0.76	0.66
MulticlassANN	InjuryType	{'lr': 0.15}	0.76	0.58	0.76	0.66
MulticlassANN	InjuryType	{'lr': 0.001}	0.77	0.64	0.77	0.68
MulticlassANN	InjuryType	{'batch_size': 32}	0.77	0.64	0.77	0.68
MulticlassANN	InjuryType	{'batch_size': 64}	0.77	0.64	0.77	0.68
MulticlassANN	InjuryType	{'batch_size': 128}	0.77	0.67	0.77	0.68
MulticlassANN	InjuryType	{'hidden_dim': 64}	0.77	0.64	0.77	0.68
MulticlassANN	InjuryType	{'hidden_dim': 128}	0.77	0.65	0.77	0.68
MulticlassANN	InjuryType	{'hidden_dim': 256}	0.77	0.64	0.77	0.68

TABLE VI
PARAMETERS FOR TUNED DECISION TREE ON PATIENT STATUS

Model	Target	Parameters	Accuracy	Precision	Recall	F1 Score
Decision Tree	PatientStatus	{'max_depth': 4}	0.72	0.74	0.72	0.71
Decision Tree	PatientStatus	{'max_depth': 5}	0.72	0.73	0.72	0.72
Decision Tree	PatientStatus	{'max_depth': 6}	0.72	0.73	0.72	0.72
Decision Tree	PatientStatus	{'max_depth': 7}	0.73	0.73	0.73	0.72
Decision Tree	PatientStatus	{'max_depth': 8}	0.72	0.73	0.72	0.71
Decision Tree	PatientStatus	{'max_depth': 9}	0.72	0.73	0.72	0.71

TABLE VII
PARAMETERS FOR TUNED DECISION TREE ON INJURY TYPE

Model	Target	Parameters	Accuracy	Precision	Recall	F1 Score
Decision Tree	InjuryType	{'max_depth': 4}	0.77	0.72	0.77	0.68
Decision Tree	InjuryType	{'max_depth': 5}	0.77	0.71	0.77	0.68
Decision Tree	InjuryType	{'max_depth': 6}	0.77	0.65	0.77	0.68
Decision Tree	InjuryType	{'max_depth': 7}	0.76	0.68	0.76	0.69
Decision Tree	InjuryType	{'max_depth': 8}	0.76	0.69	0.76	0.69
Decision Tree	InjuryType	{'max_depth': 9}	0.76	0.68	0.76	0.69

TABLE VIII
PARAMETERS FOR TUNED SVM ON PATIENT STATUS

Model	Target	Parameters	Accuracy	Precision	Recall	F1 Score
SVM	PatientStatus	{'kernel': 'linear'}	0.66	0.66	0.66	0.66
SVM	PatientStatus	{'kernel': 'poly'}	0.52	0.54	0.52	0.38
SVM	PatientStatus	{'kernel': 'rbf'}	0.53	0.54	0.53	0.43
SVM	PatientStatus	{'kernel': 'sigmoid'}	0.53	0.54	0.53	0.44

TABLE IX
PARAMETERS FOR TUNED SVM ON INJURY TYPE

Model	Target	Parameters	Accuracy	Precision	Recall	F1 Score
SVM	InjuryType	{'kernel': 'linear'}	0.74	0.65	0.74	0.68
SVM	InjuryType	{'kernel': 'poly'}	0.76	0.58	0.76	0.66
SVM	InjuryType	{'kernel': 'rbf'}	0.76	0.58	0.76	0.66
SVM	InjuryType	{'kernel': 'sigmoid'}	0.76	0.6	0.76	0.66

TABLE X
PARAMETERS FOR TUNED RANDOM FOREST ON PATIENT STATUS

Model	Target	Parameters	Accuracy	Precision	Recall	F1 Score
Random Forest	PatientStatus	{'n_estimators': 10}	0.67	0.67	0.67	0.67
Random Forest	PatientStatus	{'n_estimators': 50}	0.701	0.705	0.70	0.70
Random Forest	PatientStatus	{'n_estimators': 100}	0.70	0.701	0.70	0.709
Random Forest	PatientStatus	{'n_estimators': 200}	0.70	0.70	0.705	0.70

TABLE XI
PARAMETERS FOR TUNED RANDOM FOREST ON INJURY TYPE

Model	Target	Parameters	Accuracy	Precision	Recall	F1 Score
Random Forest	InjuryType	{'n_estimators': 10}	0.74	0.66	0.74	0.69
Random Forest	InjuryType	{'n_estimators': 50}	0.74	0.66	0.74	0.69
Random Forest	InjuryType	{'n_estimators': 100}	0.75	0.66	0.75	0.68
Random Forest	InjuryType	{'n_estimators': 200}	0.74	0.66	0.74	0.68

TABLE XII
PARAMETERS FOR TUNED XGBOOST ON PATIENT STATUS

Model	Target	Parameters	Accuracy	Precision	Recall	F1 Score
XGBoost	PatientStatus	{'learning_rate': 0.001}	0.72	0.73	0.72	0.72
XGBoost	PatientStatus	{'learning_rate': 0.01}	0.73	0.74	0.73	0.72
XGBoost	PatientStatus	{'learning_rate': 0.1}	0.73	0.74	0.73	0.73
XGBoost	PatientStatus	{'learning_rate': 1}	0.71	0.7	0.71	0.7

TABLE XIII
PARAMETERS FOR TUNED XGBOOST ON INJURY TYPE

Model	Target	Parameters	Accuracy	Precision	Recall	F1 Score
XGBoost	InjuryType	{'learning_rate': 0.001}	0.77	0.67	0.77	0.68
XGBoost	InjuryType	{'learning_rate': 0.01}	0.77	0.69	0.77	0.68
XGBoost	InjuryType	{'learning_rate': 0.1}	0.77	0.69	0.77	0.69
XGBoost	InjuryType	{'learning_rate': 1}	0.74	0.69	0.74	0.71

TABLE XIV
PARAMETERS FOR TUNED LOGISTIC REGRESSION ON PATIENT STATUS

Model	Target	Parameters	Accuracy	Precision	Recall	F1 Score
LogisticRegression	PatientStatus	{'solver': 'lbfgs'}	0.7	0.71	0.7	0.68
LogisticRegression	PatientStatus	{'solver': 'liblinear'}	0.71	0.72	0.71	0.7
LogisticRegression	PatientStatus	{'solver': 'newton-cg'}	0.71	0.72	0.71	0.7
LogisticRegression	PatientStatus	{'solver': 'newton-cholesky'}	0.71	0.72	0.71	0.7
LogisticRegression	PatientStatus	{'solver': 'sag'}	0.67	0.69	0.67	0.65
LogisticRegression	PatientStatus	{'solver': 'saga'}	0.66	0.68	0.66	0.63
LogisticRegression	PatientStatus	{'max_iter': 50}	0.69	0.7	0.69	0.68
LogisticRegression	PatientStatus	{'max_iter': 100}	0.7	0.71	0.7	0.68
LogisticRegression	PatientStatus	{'max_iter': 150}	0.7	0.74	0.7	0.68
LogisticRegression	PatientStatus	{'max_iter': 200}	0.7	0.72	0.7	0.68
LogisticRegression	PatientStatus	{'max_iter': 250}	0.7	0.72	0.7	0.68
LogisticRegression	PatientStatus	{'max_iter': 300}	0.7	0.71	0.7	0.68

TABLE XV
PARAMETERS FOR TUNED LOGISTIC REGRESSION ON INJURY TYPE

Model	Target	Parameters	Accuracy	Precision	Recall	F1 Score
LogisticRegression	InjuryType	{'solver': 'lbfgs'}	0.76	0.58	0.76	0.66
LogisticRegression	InjuryType	{'solver': 'liblinear'}	0.76	0.64	0.76	0.66
LogisticRegression	InjuryType	{'solver': 'newton-cg'}	0.76	0.65	0.76	0.66
LogisticRegression	InjuryType	{'solver': 'newton-cholesky'}	0.76	0.64	0.76	0.66
LogisticRegression	InjuryType	{'solver': 'sag'}	0.76	0.58	0.76	0.66
LogisticRegression	InjuryType	{'solver': 'saga'}	0.76	0.58	0.76	0.66
LogisticRegression	InjuryType	{'max_iter': 50}	0.76	0.58	0.76	0.66
LogisticRegression	InjuryType	{'max_iter': 100}	0.76	0.58	0.76	0.66
LogisticRegression	InjuryType	{'max_iter': 150}	0.76	0.62	0.76	0.66
LogisticRegression	InjuryType	{'max_iter': 200}	0.76	0.62	0.76	0.66
LogisticRegression	InjuryType	{'max_iter': 250}	0.76	0.62	0.76	0.66
LogisticRegression	InjuryType	{'max_iter': 300}	0.76	0.69	0.76	0.66

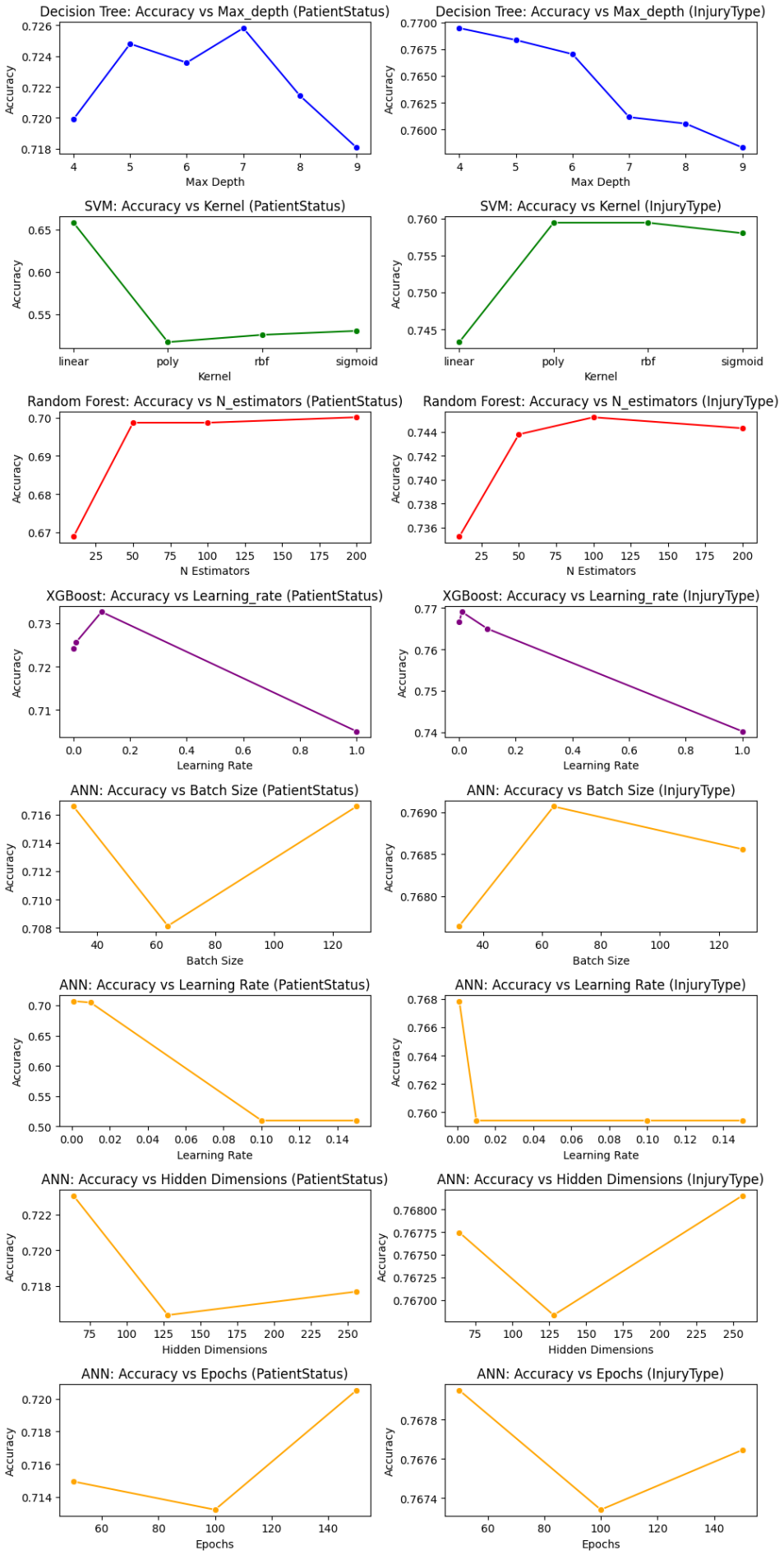

Fig. 5. Training comparison

V. Difficulties And Work Around

The most difficult part of this study was to deal with the dataset[2] because of the imbalanced data distribution on both targets (Injury Type and Patient Status), (fig 1, 2 and table 1, 2). These imbalances were evident in both the distribution of injury types and the outcomes related to patient status. In 4 and 1 as well as in Tables 1 and 2. This disparity is clearly reveals how certain injury types or patient statuses were overrepresented, while others were significantly underrepresented. For example, the number of alive patients greatly outnumber the number of dead patients. Such imbalances can lead to biased model performance, under-performing for rarer categories. To Deal better with this situation we stratified our splits and retained the lower distribution class as much as possible.

Our results on the IV section shows that our work greatly retained good accuracy, recall, precision as well as F1 score.

VI. Future Work

We plan on to extend the dataset collected from other countries in other countries and make a practical model which can actually predict real time information more reliably. This may help to reduce accidents on more accident prone areas by taking appropriate measures and also may help the hospitals be *more ready* or *"prepared"* for any incoming, accident-struck patients in emergency.

VII. Conclusion

In conclusion, this study provides a comprehensive analysis of the Road Traffic Accident [2] dataset, with particular focus on addressing the challenge due to dataset imbalance in both distributions of the target class, our careful pre-processing and the various other strategies, such as tokenizing and vectorizing, tuning model parameters and stratifying columns, we successfully achieved more than 70% accuracy on most of the model's evaluation.

In summary, while the study provides valuable insights into the challenges of working with imbalanced road traffic accident datasets, continued efforts in model refinement and data handling are necessary to advance predictive accuracy and support decision-making in traffic safety.

References

[1] U. Farooq, J. Bhatti, M. Siddiq, *et al.*, "Road traffic injuries in rawalpindi city, pakistan," *Eastern Mediterranean health journal = La revue de santé de la Méditerranée orientale = al-Majallah al-ṣiḥḥīyah li-sharq al-mutawassiṭ*, vol. 17, pp. 647–53, Sep. 2011. DOI: 10.26719/2011.17.9.647.

[2] M. S. Abid, *Road traffic accident dataset, rawalpindi-punjab, pakistan*, The dataset contains collisions happened during the year 2020 to July 2023. This can be used for analyzing traffic collisions happening in Pakistan., Chishti, Shujaat, 2024. DOI: 10.7910/DVN/4VGTDR.

[3] T. Mikolov, K. Chen, G. Corrado, and J. Dean, *Efficient estimation of word representations in vector space*, 2013. arXiv: 1301.3781 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1301.3781.

[4] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," English, in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, Valletta, Malta: ELRA, May 2010, pp. 45–50.

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[6] K. Fukushima, "Visual feature extraction by a multilayered network of analog threshold elements," *IEEE Transactions on Systems Science and Cybernetics*, vol. 5, no. 4, pp. 322–333, 1969. DOI: 10.1109/TSSC.1969.300225.

[7] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014, ISSN: 1532-4435.