Executive summary

It is important to identify the causes to accidents and make arrangements to minimize them since accidents cause a significant number of injuries and deaths every year. This report contains a detailed analysis of implementation and validation of machine learning models developed using traditional classification methods and deep learning to predict the likelihood of accidents occurring in UK. The data obtained from Emergency services of UK about accidents that occurred in UK 2019 were taken to train, validate and test the models.

The report contains detail description on each step followed in analyzing the above problem including exploratory data analysis, data preprocessing, handling class imbalance, feature engineering, model training using algorithms like decision trees, gradient decent classifier, random forest and neural networks along with the obtained accuracy metric values. In obtaining the best hyperparameters, Grid search was used and several iterations of searching was done in finetuning the parameters. The best model is selected after optimizing the hyperparameters and comparing with other models.

Results show that the best traditional classifier is Random forest with an accuracy of 80%. Additionally the report includes an ethical discussion about the applicability of the model in real world and areas to be improved and recommendations.

# 1. Exploratory data analysis

## 1.1 Reading the dataset

The given dataset includes data about **31647** accidents which took place in UK within year 2019. In accordance with the information provided, it can be stated formally that Table 1 lists 14 variable values for each accident, and the target variable of interest is the accident severity, which comprises three sub-categories, namely, 'Slight', 'Serious', and 'Fatal'.

Table 1: Variables in the dataset

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Index | Variable | type | Relevant to the final model | Specification |
| 1 | accident\_index | Categorical | No | None |
| 2 | speed\_limit | Numerical | Yes | Feature |
| 3 | light\_conditions | Categorical | Yes | Feature |
| 4 | weather\_conditions | Categorical | Yes | Feature |
| 5 | road\_surface\_conditions | Categorical | Yes | Feature |
| 6 | vehicle\_type | Categorical | Yes | Feature |
| 7 | junction\_location | Categorical | Yes | Feature |
| 8 | skidding\_and\_overturning | Categorical | Yes | Feature |
| 9 | vehicle\_leaving\_carriageway | Categorical | Yes | Feature |
| 10 | hit\_object\_off\_carriageway | Categorical | Yes | Feature |
| 11 | first\_point\_of\_impact | Categorical | Yes | Feature |
| 12 | sex\_of\_driver | Categorical | Yes | Feature |
| 13 | age\_of\_oldest\_driver | Numerical | Yes | Feature |
| 14 | accident\_severity | Categorical | Yes | Target |

## 1.2 Exploring the variables

**Categorical variables**

Chart, bar chart

Description automatically generatedBelow Figure 1 shows the percentage of categories/classes in the target variable ‘accident\_severity’

Figure 1: Distribution of the categories in the target variable

It can be observed that the

* majority of the accidents fall under the 'Slight' severity category.
* The next prevalent category is 'Severe'.
* number of samples in the 'Fatal' category is significantly lower.

Therefore, to avoid biased performance, poor generalization, and overfitting to the dominant class, it is imperative to address the class imbalance before feeding the data into the model.

Chart, bar chart

Description automatically generated

Figure 2 : Category distribution in the feature variables

Figure 2 above shows the sample distribution of each category in some of the selected feature variables.

1. In almost every variable there is a dominant category.
   * Example : Most of the accidents are happened during the daylight and fine weather.
2. There are considerable amount of samples with ‘data missing or out of range’ which needs to be addressed.

**Numerical variables**

There are 2 numerical variables

Table 2: Numerical column analysis

|  |  |  |
| --- | --- | --- |
| index | speed\_limit | age\_of\_oldest\_driver |
| count | 31647 | 25197 |
| max | 70 | 101 |
| 75% | 40 | 59 |
| mean | 36.57203 | 47.25403818 |
| 25% | 30 | 34 |
| 50% | 30 | 47 |
| std | 13.83736 | 16.77965644 |
| min | -1 | 6 |

Data cleansing should be done since some values are not acceptable

* Minimum speed recorded is -1
* Minimum age of a driver mentioned to be 6 years which is doubtful

Chart, histogram

Description automatically generatedChart, box and whisker chart

Description automatically generated

Figure 3: Accident count and severity variation with age and speed limit.

As per Figure 3,

* numerical features are ranging in a considerable range and need to normalize or standardize the data before feeding to the model.
* There are some bins in the ‘Accident count variation with age’ figure where age is very low. After analyzing in detail there 194 entries in which age of the driver <18.
* **Majority of the ‘Slight’ category accidents are happened at a vehicle speed of 30kmph.**

## 1.3 Checking for missing or duplicate values

* There are missing values in label dataset as well as in ‘age\_of\_oldest\_driver’.

Table 3: Missing values in variables

|  |  |
| --- | --- |
| Variable | Missing count |
| speed\_limit | 0 |
| light\_conditions | 0 |
| weather\_conditions | 0 |
| road\_surface\_conditions | 0 |
| vehicle\_type | 0 |
| junction\_location | 0 |
| skidding\_and\_overturning | 0 |
| vehicle\_leaving\_carriageway | 0 |
| hit\_object\_off\_carriageway | 0 |
| first\_point\_of\_impact | 0 |
| sex\_of\_driver | 0 |
| age\_of\_oldest\_driver | 6450 |
| Missing values in the label dataset | 1172 |

* Also there are 1172 duplicate entries in the total dataset

## Chart, scatter chart Description automatically generated1.4 2D visualization of the dataset using principal components

Figure 4. 2D plot of the dataset

Above Figure 4 shows how the data points are distributed in a 2D space using Principal Component Analysis. This technique aid to simplify the representation of the data while retaining the most critical information.

**Summary and key take-away to data preprocessing**

1. Target classes are imbalanced.
2. Many missing data points can be observed. Need to replace them or drop these records through imputation methods
3. Many of the features in the dataset are categorical. These categorical features will need to be encoded before they can be used in machine learning models.
4. There are null values and duplicates entries in the dataset which needs to be address

# 2. Data preprocessing

## 2.1 Data cleaning

The observed data impurities in the section 1, should be removed before proceeding in to the data splitting.

1. Replacing ‘**serious’, ‘slight’** and **‘fatal’** categories by **‘Serious’, ‘Slight’** and **‘Fatal’** respectively assuming they all convey the same category
2. Remove all the duplicates from the dataset. After removing all, the current dataset has **30475 entries**
3. Dropping the entries with invalid data types.
   * Here the 194 entries with age of the driver below 18 are dropped without further processing hence these records are doubtful
   * Entry with Minimum speed recorded -1 is dropped

After all above data cleaning the current dataset includes 30280 entries.

## 2.2. splitting (training/validation/test)

Before proceeding in to data pre-processing dataset should be split to train, test and validation to minimize the data leakage.

Target variable contains 1172 missing values and has class imbalance.

Considering above, since **stratified sampling should be applied to maintain the same class percentages in the train, validation and test datasets,** the 1172 entire rows with missing values in target variable should be removed.

Considering the percentage of missing values in the target variable which is 3% (=1172/31647) it seems ok to drop those rows.

Now the total dataset has 29148 entries and then split the dataset with below percentages.

**Splitting**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Train | Test | Validation |
| % of records | 80% | 20% | 10% from train data |

## 2.3 Handling missing values

**Numerical columns**

Replace the missing values of column “age\_of\_oldest\_driver”

* By the **mean”** of available values

**Categorical columns**

* Replace records with **“data missing or out of range”**
* By the “most frequent” category of the available values
  + The missing data percentage <5% for each column *(Makwana, 2021)*
  + Data missing is completely random
  + Easy and fast

In both above cases SimpleImputer will be **fit only to the training dataset**, calculate the mean/ mode and **then transform validation and test set to avoid data leakage.**

## 2.4 Feature scaling for the numerical values

Values in numerical columns 'speed\_limit', 'age\_of\_oldest\_driver' are scaled to have zero mean and unit variance using **StandardScaler.**

Sample

|  |  |  |
| --- | --- | --- |
| **Index** | **speed\_limit** | **age\_of\_oldest\_driver** |
| 12708 | -0.474343644 | 1.38E-16 |
| 26435 | -0.474343644 | 0.391025162 |

## 2.5 Columns transform

The ML models need data in numerical forms and hence need to transform categorical variables to numerical.

There are 2 options

1. One-hot encoding🡪 for feature dataset.
   * expands the columns in the dataset **to have one column per category.**
   * categories are not ordinal but random (like high, medium, low)
   * Number of categories are less
2. Label encoding 🡪 for label dataset
   * Categories are ordinal (fatal, serious, slight)

Note:

Encoders are only fit to training set and used for the transformation of training, testing and validation sets.

Now there are **30 columns** in the X dataset after the encoding.

## 2.6 Random over sampling

In section 1, observed a class imbalance that is needed to be addressed during the data pre-processing stage.

For that **“Random over sampling”** is used where sample count in all the 3 classes matched to that of category **“Slight” (largest class)**  by randomly duplicating samples from the minority classes.

|  |  |  |
| --- | --- | --- |
| **accident\_severity** | **Class percentage before oversampling** | **Class percentage after oversampling** |
| **Slight** | 42% | 33% |
| **Serious** | 38% | 33% |
| **Fatal** | 20% | 33% |

-

# Classification using traditional machine learning

## 3.1 Pipelines

Pipeline can be used to chain multiple estimators into one *(scikit-learn, n.d.).* In this case the pipeline (imbalance pipeline to address class imbalance) is used with below steps

1. RandomOverSampler
2. Classifier

## 3.2 Final selected model

After analysing and comparing the performance (**balanced accuracy metric**) among few classification models, the final selected model is “**RandomForest Classifier”.**

Below are the finalized model hyper parameters

Table 4: Final hyperparameters of random forest classifier

|  |  |  |
| --- | --- | --- |
| ***Hyperparameter*** | ***Value*** | ***Description*** |
| n\_estimators | 775 | Trees count in the forest.   * Helps in reducing the variance and improve the generalization of the model. If * Too high 🡪 overfitting and time consuming * Low 🡪 underfitting |
| Criterion / impurity | ***Gini*** | Quantify the impurity of a given subset of data at a node based on the proportion of different classes or labels.   * The impurity is high when a node contains mix of instances from different classes or labels. * Used to split the data at each node so that the impurity is reduced after the split * ***Gini*** entropy calculated as below     Pmk= proportion of class k observations in node m |
| max\_depth | 11 | The number of nodes from root node to the farthest leaf node in a single decision tree   * If too high, the model tends to overfit * If too low, model performance is degraded |
| oob\_score | False | use out-of-bag samples or observations which have not beenused to grow the tree (Niladri Syam and Kaul, 2021) to estimate the generalization score |
| min\_samples\_split | 9 | If a node has fewer samples than this, it will not be split further, and the splitting process will stop for that branch.   * If too high, underfitting with low model complexity * Low, overfitting |
| max\_features | sqrt | The features count to consider when looking for the best split   * **Sqrt:** number of features considered for splitting at each node is the square root of the total number of features |
| class\_weight | 'balanced' | Weights are adjusted so that they are inversely proportional to class frequencies |

Random Forest is an ensemble learning method (combines multiple models to improve the predictive performance of the overall system ) that makes use of **bagging** to improve the performance and accuracy of the classification. The Random Forest algorithm works by creating multiple decision trees on different subsets of the dataset using a technique called **bootstrap aggregation (bagging).** The algorithm randomly samples the dataset and feeds these samples separately to each decision tree. The **out-of-bag** (non-selected) **samples** are then used to generalize the model.

At each node of the decision tree, the algorithm selects the best feature to split the data by looking at the impurity (**like Gini impurity and entropy)** and other criteria such as **minimum samples required to create a split**. The samples used to split a node are randomly selected from the dataset, and this process is repeated until the tree reaches its maximum depth.

Once all the decision trees are created, classification happens by voting. Each decision tree classifies the data point, and the class with the most votes across all the trees is the final classification.

Random Forest also integrates class weight to balance the distribution of samples in the dataset. This is important when the dataset is imbalanced, and one class dominates the dataset. By assigning weights to each class, the algorithm gives more importance to underrepresented classes, which improves the accuracy of the classification.

## 3.3 Experimental analysis

1. **Comparison among models to find the most appropriate model**

* 5 models were developed including random forest classifier to compare the performance and select the best.
* The hyperparameters for all the 5 models were selected using **Randomized Parameter search**.
* **Balanced accuracy** metric used to assess the performance of the model during hyper-parameter optimization and while comparing with other models.
  + Average recall across the classes
  + Provide good accuracy for imbalanced classes.

Below is the comparison in final test balanced accuracy.

Chart, bar chart

Description automatically generated

Figure 5: Accuracy comparison among the models

As per Figure 5, the highest balanced accuracy is given by the Random forest models and hence selected it for the classification task.

1. **Hyper-parameter optimization**
   * 1. Defining search space

|  |  |  |
| --- | --- | --- |
| hyperparameter | space | Reason for range specification |
| n\_estimators / Number of trees | 1 to 1000 | \*\* higher number 🡪 better performance  increases computational cost  can overfit  \*\* 1-1000 is reasonable |
| min\_samples\_split | 2 to 10 | \*\* if low 🡪 overfitting  \*\* if high 🡪 underfitting.  \*\* 1-20 is reasonable |
| max\_depth | 1 to 20 |
| criterion | 'gini', 'entropy', 'log\_loss' | \*\* available options to select the optimum |
| Max\_features | 'sqrt','log2',None |
| Oob\_score | True or False | \*\* whether including OOB samples can be computationally efficient and can help estimate performance on new data |

1. The hyperparameter tuning was done using **“Randomized Parameter Optimization**” to increase the efficiency over exhaustive **Grid Search** which exhaustively generates candidates from a parameter grid.
2. **5 fold Stratified Cross validation** was used to increase the accuracy since the dataset is imbalanced.
   * + - To ensure that **each fold has a representative proportion of each class**,
       - can avoid the model training on **biased data to majority class**, which could lead to poorer performance.
3. Hyperparameter prioritization:

Table 5: Hyperparameter tuning stages random forest classifier

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Model | Hyperparameters included in the search space | | | | | | | Balanced Accuracy (test) |
| Criterion | Tree count | Max depth | min\_samples\_  split | oob\_score | max\_features | min\_samples  leaf |
| RF\_1 | X | X | X |  |  |  |  | 79.6% |
| RF\_2 | X | X | X | X |  |  |  | 79.89% |
| RF\_3 | X | X | X | X | X | X |  | 80.00% |
| RF\_4 | X | X | X | X |  |  | X | 79.79% |

Table 3 shows the 4 stages of hyperparameter tuning in the random forest model. At each phase different set of hyperparameters are fed to the random seach space to reduce overfitting and finding the optimum

**if all parameters are included in the search space there is a high chance of model getting overfit.**

**Chart, line chart

Description automatically generated**

Figure 6: Accuracy change in each phase of tuning

Above Figure 5 shows

* how training, validation and test balance accuracy value variation for the 4 stages
* **RF\_3** has the highest balanced accuracy on test data with least overfitting
* In model 4 (RF\_4) model gets highly overfit.

**Hence taken model 3 (RF\_3) as the finalized one and best estimators were finalized as per the table 2.**

## 3.4 Model evaluation

### 3.4.1. Confusion matrix

As per the confusion matrix in Figure 7, majority of the samples are predicted correctly but some of the predictions are inaccurate mainly in classes **Severe** and **Fatal**.

Chart, treemap chart

Description automatically generated

Figure 7: Confusion matrix for final selected model

### 3.4.2. Performance metrics

Below metrics are used to assess the accuracy of the model other than the balanced accuracy.

1. **Precision**

* Indication to show if the model predicts a certain class, how certain that the prediction is accurate.
* Since the dataset is having class-imbalance, **Precision**  is a good metric compared to accuracy here, where a high precision score indicates that the classifier has accurately identified the classes.
* Characterized by the equation
  + TP / (TP + FP)
    - TP – True Positive (correct classification)
    - FP – False Positive (incorrect classification)

1. **Recall**

* Indication to show what fraction of class data points, the algorithm has able to recover correctly.
* measures the proportion of true positives among all positives.
* In a class-imbalanced dataset, a high recall score indicates that the classifier is able to identify most of the positive cases.
* Characterized by the equation
  + TP / (TP + FN)
    - FN – False Negative

The model has below precision and recall values for the 3 classes

|  |  |  |
| --- | --- | --- |
|  | Precision | recall |
| Slight | 0.62 | 0.94 |
| Serious | 0.82 | 0.71 |
| Fatal | 0.86 | 0.75 |

**Discussion**

The recall for class ‘Slight’ is very low and precision is high compared to others meaning, models is bit biased to predict the majority class Slight.

### 3.4.2. comparison with one “trivial” baseline

Trivial baseline:

**always predict the majority class, which is "slight"**

Below table depicts what are the accuracy, precision and recall values per each class according to the trivial baseline.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **accident\_severity** | **percentage** | **Accuracy** | **Recall** | **Precision** |
| **Slight** | 42% | 0.42 | 1.00 | 0.42 |
| **Serious** | 38% | 0.42 | 0 | 0 |
| **Fatal** | 20% | 0.42 | 0 | 0 |

The balanced accuracy according to the above table = 0.33%

**Conclusion :**

* Final model performs better than the trivial baseline
* Model has learned meaningful insights and patterns from the data and it's providing accurate predictions.

# 4. Classification using neural networks

## 4.1 Final selected model

The final selected model from is Sequential Recurrent Neural networks. The finalized model parameters as per below and the **final model balance accuracy on test data is 79.45%**

Table 6: Final hyperparameters of sequential NN

|  |  |  |
| --- | --- | --- |
| ***Hyperparameter*** | ***Value*** | ***Description*** |
| Neurons count | Input layer- 30  Dense layer 1- 20  Dense layer 2- 10  Output layer - 3 | Number of nodes in the hidden layer.   * Basic computational unit which process information * If too high more complex the model and overfitting can be happened * If too low, model may failed to learn patterns from data and hence underfit |
| Layers | 7 | collection of neurons   * Building block of a NN and extract representations out of the data fed into them * Each layer is connected with next and previous layers * Last layer is a 3 way softmax layer which outputs an array of 3 probabilities which sum up to 1. *(Chollet, 2018)* |
| Loss | SparseCategoricalCrossentropy | Used by the network to measure the performance on training data and adjust the hyperparameters on reducing the loss function   * Sparse categorical entropy computes the cross-entropy between the predicted probability distribution and the true probability distribution |
| Optimizer | SGD | The mechanism through which the network update itself depending on the loss function and data. |
| Optimizer learning rate | 0.01 | step size at which the optimizer updates the weights of the model during training |
| Activation function | Relu | A mathematical function which is applied to the output of every layer to introduce non-linearity so that the network can model complex patterns in data.   * ReLU function returns the input if it is positive, and 0 otherwise * Simple and effective. |
| Batch size | 48 | determines how many samples are processed at a time before the model updates its weights.   * If large, faster training and high computation resource requirement |
| Epochs count | 50 | number of times the entire training dataset is passed through the neural network during the training process. Each epoch consists of multiple batches of data, |
| Drop out | 0.3 | Regularization technique   * “Dropout, applied to a layer, consists of randomly dropping out(setting to zero) a number of output features of the layer during training.” *(Chollet, 2018)* |

In above neural network model the input layer of the model has 30 nodes that receive the input features. The dense layers with ReLU activation function that applies a linear transformation to the input data and introduces non-linearity to the model. The output layer has 3 nodes with the softmax activation function that outputs the predicted probabilities for each class.

During training, the model uses the stochastic gradient descent (SGD) optimizer with a learning rate of 0.01 to adjust the weights and biases of the model in the direction that minimizes the loss function. The loss function used is the sparse categorical cross-entropy, which compares the predicted probabilities with the actual class labels and penalizes the model for making incorrect predictions.

To prevent overfitting, 3 dropout layers with dropout rates of 0.3 are added which randomly drops out some of the nodes during training and helps the model to generalize better to new data. Model will iterate over the entire training dataset 50 times (epochs). During each epoch, the model will divide the training data into batches of size 48 and then update the model parameters based on the gradients computed from those 48 samples.

After training, the model is evaluated on a separate validation or test dataset to assess its performance. The model makes predictions on the test data, and the balanced accuracy metrics are calculated to determine how well the model generalizes to new data since there is a class imbalance in the dataset. Finally, the model can be used to make predictions on new, unseen data to classify the severity of accidents based on their features.

## 4.2 Experimental analysis

In selecting the optimum model 2 steps were done

1. Comparison with other neural networks.

The sequential model’s performance is compared with **1 dimensional convolution neural network** to assess the performance.

* Initially defined the Keras neural network model, wrapped it using **SciKeras wrapper and used it within a RandomizedSearchCV** to optimize the hyper-parameters in both models since it is not effective to use manual methods.
* Since the dataset suffers class imbalance **3 fold stratified cross-validation** was used while searching for the best hyperparameters to increase the accuracy
* The 1D CNN model showed a balance accuracy of **79.16%** which is lower than that of **sequential model which is 79.45%**
* Selected sequential model

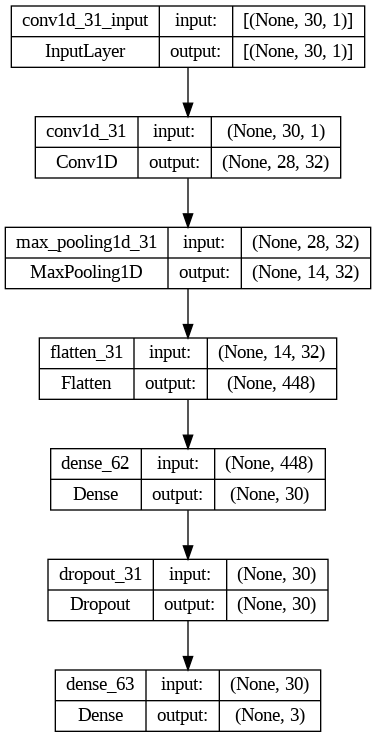
****

Figure 8: Architecture of the used CNN 1D model

1. Tune the hyperparameters of sequential model

During this approach, Hyperparameter tuning is done in 3 stages by creating 3 different models to be fed to scikeras wrapper.

* + a simple sequential model built using **random hyper-parameter selection and evaluating on the unseen data**
  + using the above model as the **base line** optimizing the hyperparameters in the next 2 models.
    - **In the simple model could observe overfitting with training accuracy> test accuracy**
    - In model 2, added dropout layers while creating the model
    - In model 3 another dense layer with a dropout layer was introduced to increase model complexity.
    - All hyper parameters like nodes count, batch size, epoch count, were taken from randomized search CV
  + Another change done here is the incorporation of ‘**Early stopping’** which is a callback option in Keras. The EarlyStopping callback allows to stop training early if the validation loss stops improving for a specified number of epochs (called the **patience**) . This can help **prevent overfitting** and **save time by avoiding unnecessary training**

Table 7: stages of hyperparameter tuning

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Neurons count | | | | Drop  Out layers | Activation | Optimizer | Learning  rate | Epochs | Batch size | Balance Accuracy  (Test) |
| Input layer | Dense layer 1 | Dense layer 2 | Output layer |
| Simple model | 30 | 10 | - | 3 | No | Relu | Relu | Relu | 25 | 64 | 78.47% |
| Model 1 | 50 | 10 |  | 3 | 2 | Relu | Relu | Relu | 50 | 32 | 79.31% |
| Model 2 | 20 | 20 | 10 | 3 | 3 | Relu | Relu | Relu | 50 | 48 | 79.45% |

The best model was chosen to be Model 3 with highest balance accuracy and its hyperparameters given in details in table 6.

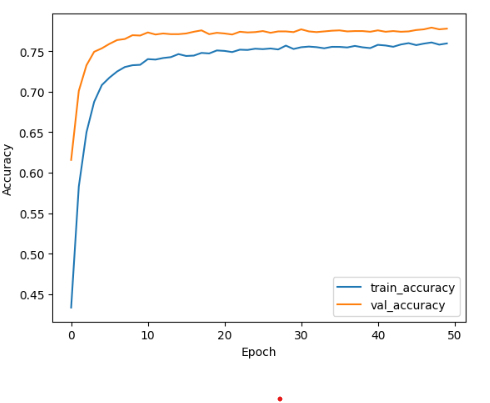
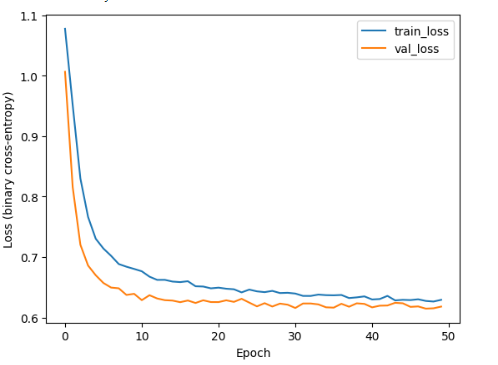


Figure 9: Training, validation accuracy and loss variation with the epochs in final sequential model

Figure 9 shows how the validation error reduce and accuracy improves with the epoch count.

Chart, bar chart

Description automatically generated

Figure 10: Accuracy comparison among the built and trained NN models

## Model evaluation

### Confusion matrix

Figure 11: Confusion matrix for the selected sequential model

As per the confusion matrix in Figure 9, majority of the samples are predicted correctly but some of the predictions are inaccurate mainly in classes **Severe** and **Fatal**.

### Performance metrics

The model has below precision and recall values for the 3 classes and comparison with the RF classifier

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Sequential model Neural networks | | Random forest classifier | |
| precision | recall | Precision | recall |
| Slight | 0.61 | 0.94 | 0.62 | 0.94 |
| Serious | 0.82 | 0.7 | 0.82 | 0.71 |
| Fatal | 0.86 | 0.75 | 0.86 | 0.75 |

**Discussion**

In both the models, recall for class ‘Slight’ is very low and precision is high compared to others meaning, models are bit biased to predict the majority class Slight.

All the scores are almost same for both models but there is a slight increment in recall for class ‘Serious’ in Random forest classifier hence the best model. But overall both models are performing at same level.

### 4.3.3. comparison with one “trivial” baseline

Trivial baseline:

**always predict the majority class, which is "slight"**

As discussed in section 3.4.2 the trivial balanced accuracy = 0.33% and since the NN model accuracy is 79.45%

* Final model performs better than the trivial baseline
* Model has learned meaningful insights and patterns from the data and it's providing accurate predictions.

# 5. Ethical discussion

Below mentioned are some of the social and ethical implications which can take place using the above finalized ML model in predicting the severity of road accidents organized using the Data Hazard Labels framework.

1. Data collection phase

* Hazard: Reinforces Existing Biases
  + It is important to ensure that the data used for training the model is representative of the population being studied and does not introduce any biases. For example, if certain communities are underrepresented in the data, the model may not accurately capture their experiences and could perpetuate existing inequalities.
* Hazard: Risk To Privacy
  + Additionally, it is important to consider the privacy of individuals whose data is being used. Sensitive information such as age and sex could be used to perpetuate discrimination, and so it may be necessary to ensure that data is anonymized.

1. Data processing

* Hazard: Reinforces Existing Biases
  + Oversampling the minority class using random sampling and imputing missing values using the mean or most frequent value may also introduce biases into the model. For example, if older drivers are more likely to have missing age data, the model may not accurately capture their experiences.

1. ML Prediction:
   * Hazard: Difficult To Understand
   * Used evaluation metrics that are easily interpretable and explainable, such as accuracy, precision, and recall. This will allow stakeholders to understand the model's performance and limitations, and to identify areas for improvement
   * Use machine learning models that are transparent and explainable, random forest and neural networks with this documentation which can be used as guide to understand how the model is working and model’s prediction for the stakeholders.

# 6. Recommendations

## 6.1 Machine learning model which is the best candidate for the task and reasons

After all the hyper-parameter optimizations and comparing with most of the classifications methods, “Random Forest” is the model which gave the highest balance accuracy and hence it is taken as the best candidate.

Random.

1. Random Forest is known for its robustness to noise and overfitting. Since the a dataset may contain noise or outliers, Random Forest may perform better in such situations as compared to other models that are sensitive to noise and outliers.
2. Interpretability: Random Forest is also relatively easy to interpret as compared to other models such as neural networks..
3. Computational efficiency: Random Forest is also computationally efficient and can handle large datasets with high dimensionality.

## 6.2 Whether the final model is good enough to be used in practice and why.

Since the accuracy of the model is 80% which implies that every 100 predictions the model makes there is an 20 erroneous predictions. An 80% accuracy may not be sufficient for a critical scenario like predicting accident severity, as it could lead to erroneous predictions and result in the wrong decisions being made by the authorities. Therefore, a higher accuracy would be desirable to ensure the model's predictions are more reliable, and it could be necessary to explore further model tuning or collecting more data to achieve this

## 6.3 Top suggestion for future improvements

**Incorporating more contextual information**

The dataset can be enriched with contextual information like traffic patterns, and driver behaviour, which can help improve the model's accuracy. Collecting more data can help to improve the model's performance. More data can provide the model with additional insights that could enhance the accuracy of the predictions.

# 7. Retrospective

# Model interpretation: Understanding how each of the model arrives at its predictions is important for building trust in the model and ensuring its fairness. Will invest more time in investigating different model interpretation techniques and explore how they can be used to gain insights into the model's decision-making process.

# 8. References

1. Makwana, K. (2021). Frequent Category Imputation (Missing Data Imputation Technique). [online] Geek Culture. Available at: <https://medium.com/geekculture/frequent-category-imputation-missing-data-imputation-technique-4d7e2b33daf7>. (Accessed on : 31-03-2023)
2. Mirjalili, S. (2019). PYTHON MACHINE LEARNING - THIRD EDITION : machine learning and deep learning with python, scikit... -learn, and tensorflow 2. S.L.: Packt Publishing Limited
3. scikit-learn. (n.d.). 6.1. Pipelines and composite estimators. [online] Available at: <https://scikit-learn.org/stable/modules/compose.html#pipeline>. (Accessed on : 31-03-2023)
4. scikit-learn.org. (n.d.). 3.2. Tuning the hyper-parameters of an estimator — scikit-learn 0.23.2 documentation. [online] Available at: <https://scikit-learn.org/stable/modules/grid_search.html#grid-search>. (Accessed on : 01-04-2023)
5. <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

(Accessed on : 01-04-2023)

1. Chollet, F. (2018). Deep Learning with Python. Shelter Island (New York, Estados Unidos): Manning, Cop.