Comp1804 Report: Predicting the severity of road accidents in the UK

001245654

17th March 2023

Word count: 2915

Executive summary

We have structured a model that effectively predicts severity of road accidents in the UK and compared it to a neural networks model to check which one responds better. We have a large dataset that contains categorical and numerical features that helps categorize accident severity. To begin with we loaded necessary libraries. The dataset has undergone pre-processing, visualization and building prediction models. Our problem is to perform multi-class classification and type of learning would be supervised machine learning as we have already been given labeled data. We have used Random Forest Classifier, Gradient Boosting Classifier, Stochastic Gradient Descent Classifier, KNeighborsClassifier and Multinomial Naive Bayes classifier to see which one performs most accurately. Random Forest Classifier turned out to be best and We have performed Feature Selection, Hyperparameter tuning and Random Under Sampling to see if we can improve its accuracy to a higher level.

At the end a neural network is set up to compare our traditional algorithm's performance. Both models comparatively achieve the almost same accuracy. As our dataset is large and does not have complex and subtle relationships Random Forest Classifier would be the best fit for our dataset. This is to say that the Neural Network model requires more computational resources and is flexible to understand the very complex and deeper relationship between target variable and feature columns.

1. Exploratory data analysis

We have performed several data visualizations to gain a wide understanding of our data. First we have plotted correlation heatmap to understand correlation between feature columns and target columns. As it can be seen in FIgure 1 'speed_limit' , 'hit_object_off_carriageway', 'skidding_and_overturning' and 'vehicle_leaving_carriageway' have stronger relation with accident_seveirty than others. Other columns with lower correlation will be set apart further. For dimensionality reduction we will select columns with higher correlation and importance. It can reduce computational cost and improve model accuracy. This will be double checked when visualizing the best features selection method. Alos, correlation heatmap is computed as a feature importance.

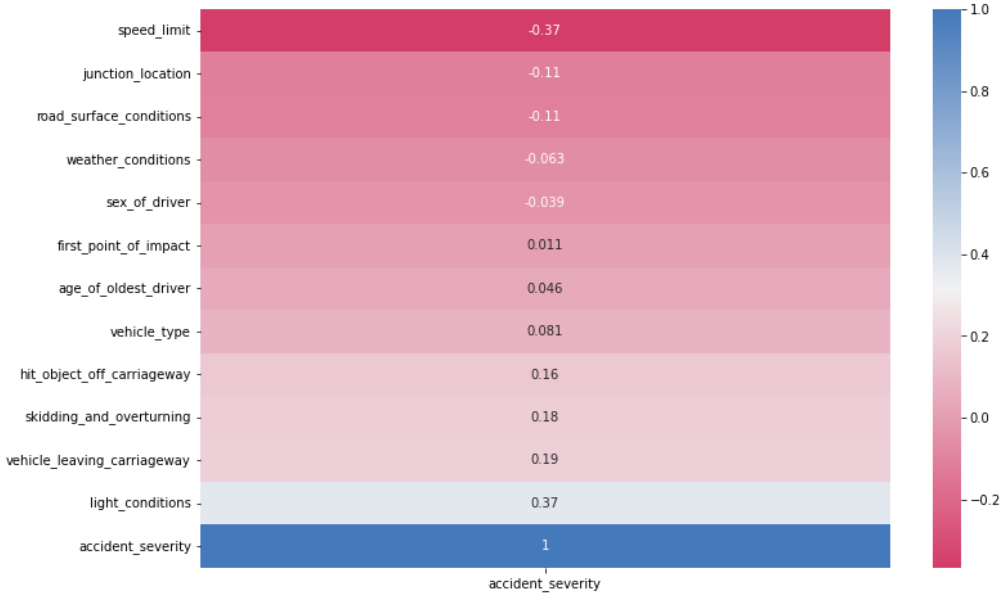

Figure 1. Correlation Heatmap

After that, We have plotted distribution plots to check if categories in categorical columns are well distributed and need balancing or sampling. As we have used the Random Forest Classifier model it is efficient enough to learn noisy features from the dataset. However, we would validate it during model evaluation.

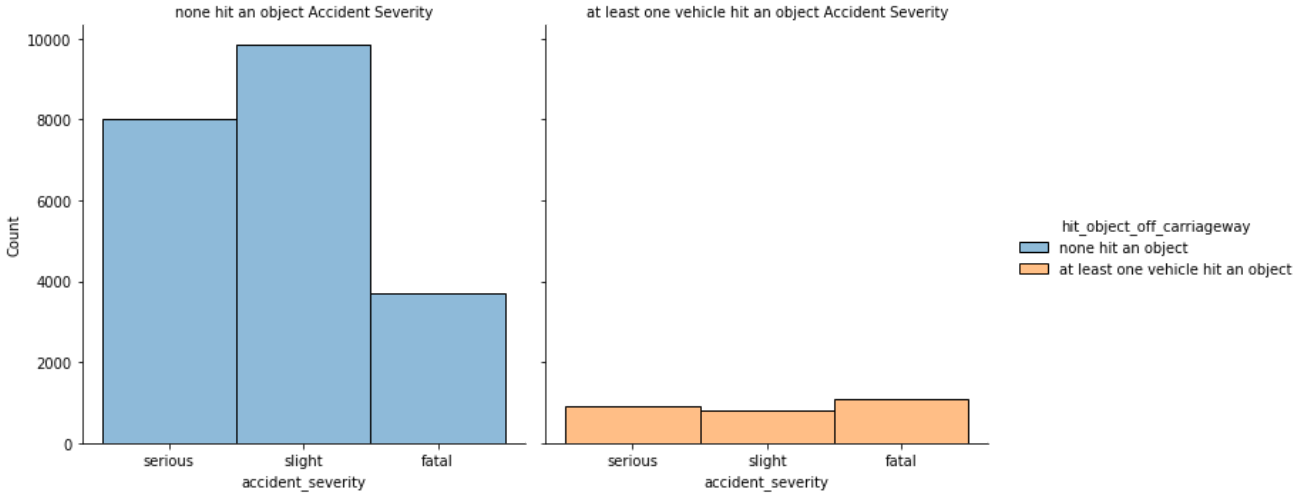

Figure 2. Distribution plot for 'hit_object_off_carriageway'

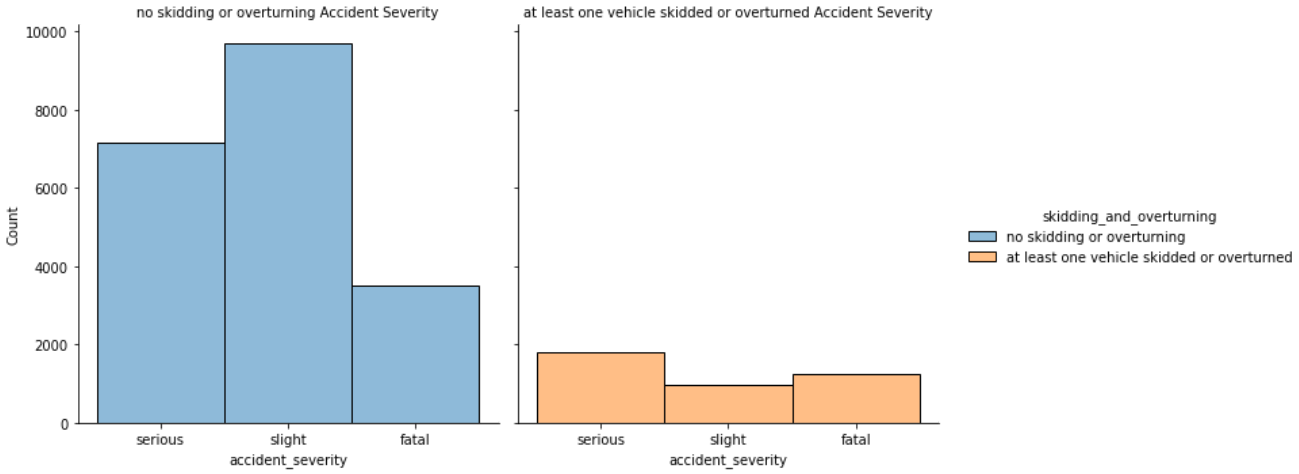

Figure 3. Distribution plot for 'skidding_and_overtuning'

From our visualization of distribution, categories in feature columns does not seem to be biased towards one target category.

Subsequently, we have plotted a feature importance chart to understand dimensionality reduction and model evaluation. It is somehow related to correlation heatmap.

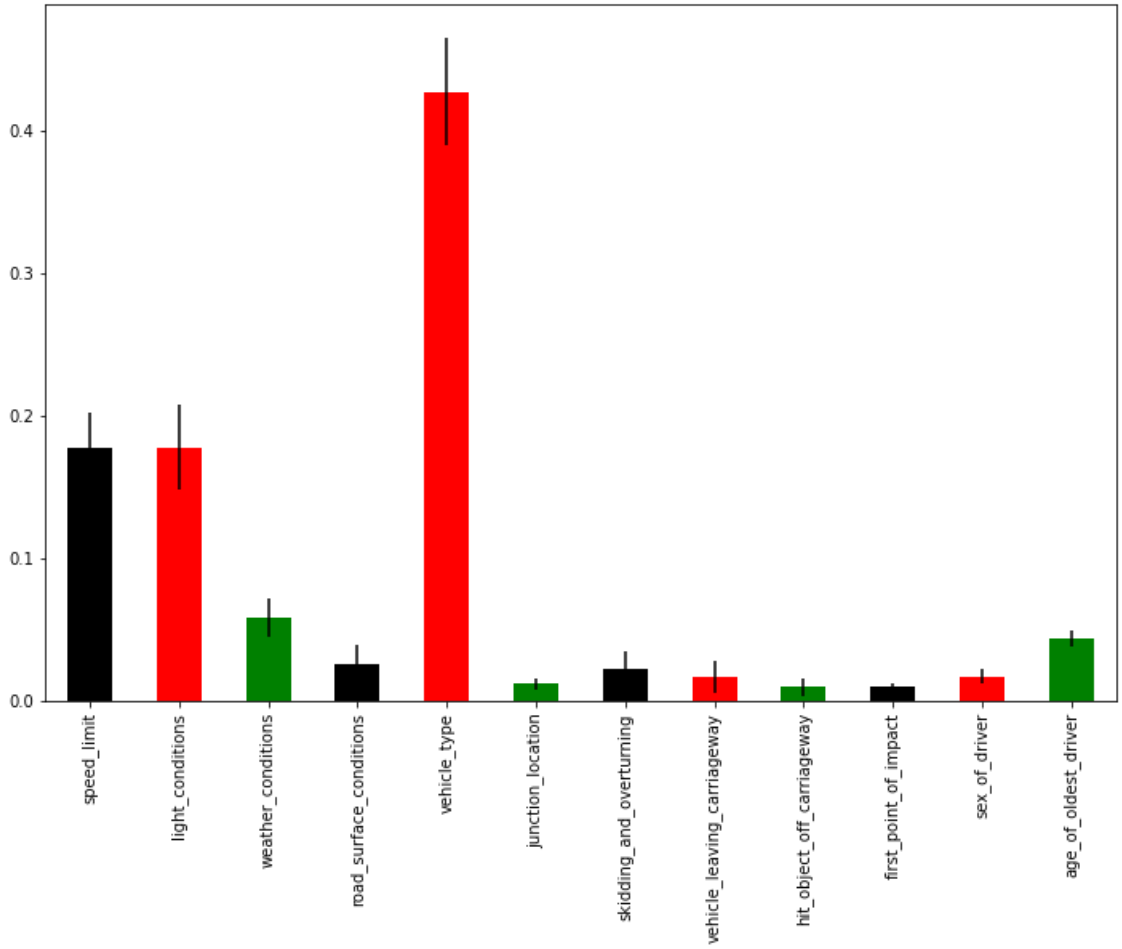

Figure 4. Bar chart for feature importance

From the feature importance bar chart we can easily choose which feature columns are important and have a strong relationship with our target variable.

2. Data preprocessing

At the initial stage, we have observed our dataset and gathered some tasks that need to be done within data preprocessing. We had seen that our dataset contains 31647 rows and 14 columns. We have also taken a look at the datatype of columns in our dataset. After that, we have found that data has 1172 duplicated rows. For multi class classification problems duplicated rows can fabricate feature importance and can lead to weak model performance. Dropping duplicate rows can enhance feature and target relationship and can help models to generate accurate predictions.

Subsequently, we have checked the datatype of each column and verified that there are three categories in our target column - fatal, slight and serious. We have overseen the uniqueness of values in each column and noticed that 'Slight', 'Fatal' and 'Serious' needed to be converted into their lower case form. Also, it would be logical to think that speed_limit can never be -1. So, -1 is replaced with 0. The string 'data missing or out of range' has been replaced with null.

After that, we split our data into training and test in 80%:20% proportion. As our dataset is large and our aim is to get higher accuracy so I would want our model to learn on a large training set to learn underlying relationships in an effective way. The lack of larger sample sizes makes the validation technique even more critical in these studies in terms of deriving insights and making comparisons using machine learning models. (Singh, 2021).

After training/test split we see the proportion of null values in each column. For categorical columns with less than 15% null values proportion are filled up by most frequent imputation technique. For categorical columns with more than 15% null values proportion is filled up by constant imputation method in which constant has the lowest count among other categories in the column. For numerical column ('age_of_oldest_driver') with more than 15% null values have been filled up by k-Nearest Neighbors method. This method can reduce bias results and make relationships easier to understand for our model. Another solution was to use a regression model to fill missing values in 'age_of_oldest_driver' but it would make our dataset complex and computationally intensive for classification tasks. Filling missing values operation should be performed on training and test both dataset.

After that, we performed encoding by ordincalEnc (Ordinal Encoding Technique) . Practically speaking, it would be logical to use ordincalEnc from sklearn.preprocessing as our feature columns and target variables are discrete categorical values. Also, encoding should be fit on training and test both to ensure they are trained and tested on the same set of features. The reason behind the label encoding is that machine learning requires numerical values as an input, it can not understand categorical strings.

Removal of duplicate values, Filling missing values, training and test splitting and label encoding operations have readied our dataset to begin with classification models.

3. Classification using traditional machine learning

Our goal is to predict the categories for the 'accident_severity' column from the given dataset. After data preprocessing we have learned that the column contains three categories: serious, fatal and slight. Practically speaking when it comes to predict more than two categories from given the labeled data the problem is

called multi class classification and type of machine learning should be supervised machine learning. We have chosen Random Forest Classifier, Gradient Boosting Classifier, Stochastic Gradient Descent Classifier, k-nearest Neighbors Classifier and Naive Bayes Classifier for the problem. Ultimately, Random Forest and Gradient Boosting Classifiers give the same accuracy approximately 74% and it is higher then rest. Compared to Random Forest the Gradient Boosting may struggle with a large number of iterations and can be computationally intensive. After comparing Prediction Accuracy, Precision Score, Recall Ration and R1 Score for each model I decided to move forward with Random Forest Classifier as it is robust to massive data and computationally efficient. Also, it learns better on high-dimensional data than the Gradient Boosting Classifier method. As it uses multiple trees to get trained it is less likely to be prone to overfitting.

Random Forest Classification model

Code - `RandomForestClassifier(n_estimators=1000, max_depth=5, max_features="sqrt", min_samples_split=2, min_samples_leaf=1)`

Parameters explanation -

- n_estimators=1000 - Higher number of trees can improve our model accuracy but there is a chance that higher numbers can make our model computationally intensive. So, 1000 trees are sufficient.
- max_depth=5 - Deeper number of depth can understand complex relationships but as our dataset is not too complex to understand it would be logical to have less number of maximum depth.
- max_features=sqrt - Helps to decide the number of features to contemplate during split and avoid overfitting. Square root of the total number of feature columns will be used at node splitting.
- min_samples_split=2 - Specifies number of samples needed for splitting internal nodes. Higher value can lead to underfitting and lower value can lead to overfitting. For our model 2 works as best.
- min_samples_leaf=1 - Each leaf node in our model will at least contain 1 sample.

Parameter tuning depends on the type of problem, size of our dataset and complexity of non linear relationships. Above parameters are best practice for our model performance and its accuracy as it minimizes overfitting by limiting the maximum depth of the tree to 5 and requiring a minimum of 2 samples to split a node.

Feature Selection method

Code - `SelectKBest(score_func=f_classif,k=6)`

Parameters explanation -

- `score_func=f_classif` – f_classif parameter is default value used as scoring parameter. It computes the F value used in ANOVA (analysis of variance) for feature and target value (www.ibm.com, n.d.).
- `k=6` – From figure 3 it is obvious to see that using at least 6 features may improve the model's accuracy.

The feature selection method has listed out most important columns which would again fit into a model for training and prediction. It has slightly increased the accuracy and performance of our model by removing redundant features. Figure 4 has a clear and detailed visualization of each feature's importance.

Random Under Sampling method

Code - `RandomUnderSampler(random_state=42)`

Parameters explanation -

- `random_state=43` – This parameter ensures that random undersampling runs the same way every time function is run. With value 43 it produces a little increment in accuracy.

Both feature selection and random under sampling does not make any significant and higher improvements in the Random Forest Classifier model. However, hyperparameter tuning has improved the accuracy a little bit by almost 1%.

Evaluation of model performance for Random Forest Classification -

1. Confusion Matrix -

The Confusion Matrix model provides insight into predicted and actual classes from our model. It has generated a matrix for our Random Forest Classifier which shows counts for false negatives and true positives.

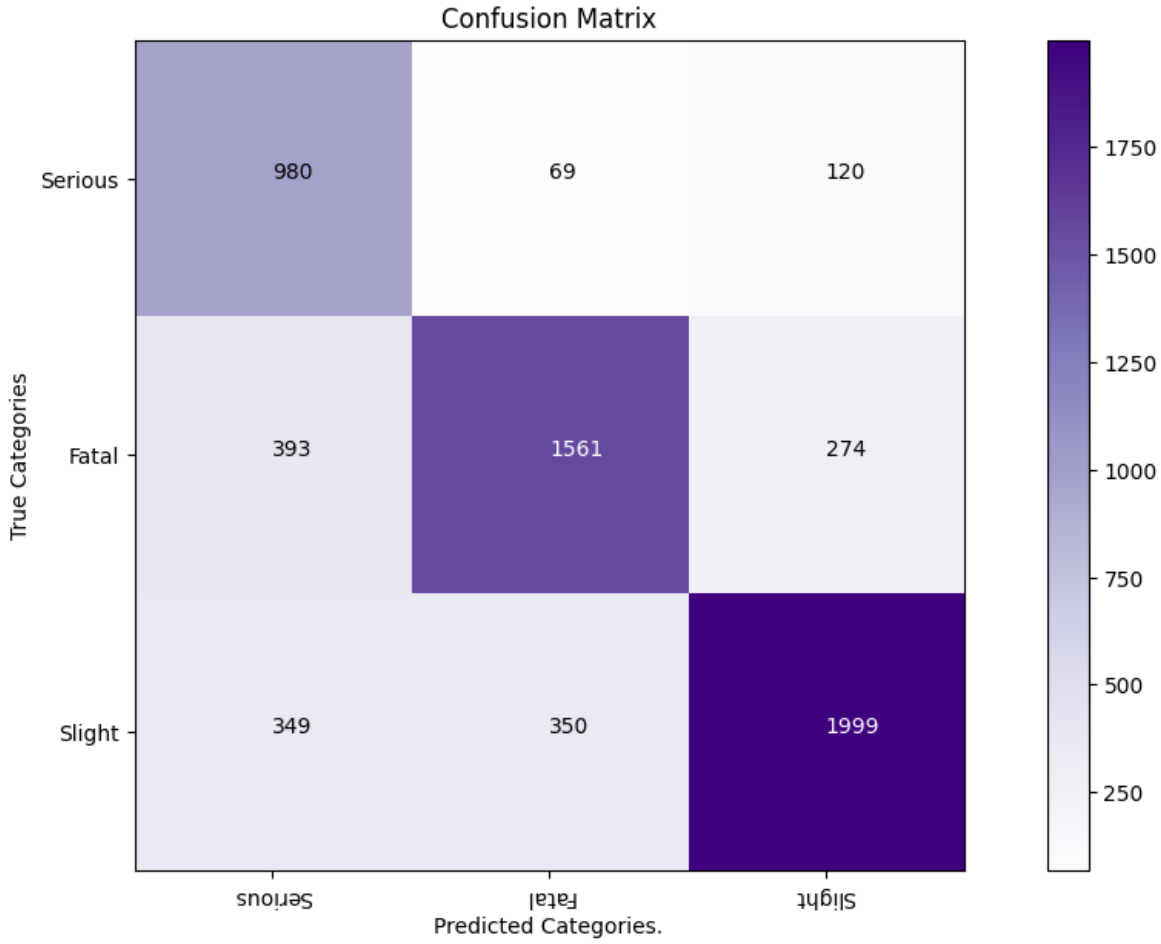

Figure 5. Confusion matrix for Random Forest Classification

The x-axis and y-axis of the matrix correspond to the predicted categories and true categories, respectively. On the one hand, from Figure 5 the diagonal elements of the matrix show the number of categories that are correctly classified. On the other hand, off-diagonal elements show the number of categories that are misclassified. The text inside each cell of our confusion matrix specifies the count of categories in that cell. As we have detailed breakdown of model's performance for accident_severity categories we can identify which part of the model needs rework.

From Figure 5 we can see that the 'Slight' category has 349 and 350 misclassified categories in 'Serious' and 'Fatal' respectively. Taking these false negatives to lower counts can notably increase a model's accuracy. Same procedure should be followed for the 'Fatal' category.

2. Performance Metrics -

At this stage we have checked Prediction Accuracy, Precision, Recall and F1 Score to evaluate the working of our model. As each metric provides a different perspective of the model's performance, all of them are quite appropriate to use.

Performance Metrics	Score
Prediction Accuracy	0.7447087776866284
Precision Score	0.7688725000600715
Recall Score	0.7447087776866284
F1 Score	0.7488058418840438

Table 1. Performance Metrics

From table 1 we have around 74% accuracy, 76%, 74% and 74% score for Prediction, Precision, Recall and F1 respectively. Prediction accuracy tells the proportion of correctly predicted samples. Precision and Recall score shows the amount of true positive predicted categories. F1 Score shows weighted harmonic mean of precision and recall score. In cases where data is not properly balanced and categories are not well distributed, Recall score may not be sufficient. In terms of our dataset precision score and recall score is informative as it is around 74% which can be considered as a good performance.

3. Comparison with trivial baseline -

In our dataset the majority of accident severity is 'Slight' and accounts for the majority of accidents in our dataset. We have achieved 44% accuracy with 'Slight' severity. Theoretically speaking, high accuracy with a majority class classifier does not determine or assess performance of our model. As accuracy of Random Forest Classifier model is significantly higher than majority class classifier, the model is likely capturing some relationships between feature columns and target and providing useful information to predict 'accident_severity'.

Trivial baseline is not the only factor that judges the performance of our model. We have achieved considerably higher accuracy in Performance Metrics around 75%.

4. Classification using neural networks

To achieve our objective of predicting categories for the accident_severity column we have defined two neural networks:- a. Multilayer Perceptron Neural Network b. Convolutional Neural Network. As our dataset is large and has multiple non-linear relationships these two models are the best choice to achieve our goal of predicting categories.

On the one hand, Multilayer Perceptron is a type of feedforward neural network and recommended to perform well on a multi class and learning complex non-linear relationships between the input features and output classes. But, our dataset does not contain complex relationships.

On the other hand, Convolutional Neural Network is well-known for image classification tasks which is not our aim at all. Fortunately, Convolutional Neural Network has performed better on our dataset than Multilayer Perceptron Neural Network. The Convolutional Network is time efficient than the former one as it reduces the number of parameters in the model and improves its ability to generalize to new, unseen data.

Convolutional Neural Network -

```
convolutionalNeuralNetwork = Sequential()
convolutionalNeuralNetwork.add(
      Conv1D(
            filters=32,
            kernel_size=3,
            activation='relu',
            input_shape=(x_train[selected_feature_names].shape[1], 1)
      )
)
convolutionalNeuralNetwork.add(MaxPooling1D(pool_size=2))
convolutionalNeuralNetwork.add(Flatten())
convolutionalNeuralNetwork.add(Dense(50, activation='relu'))
convolutionalNeuralNetwork.add(Dense(3, activation='softmax'))
convolutionalNeuralNetwork.compile(loss='categorical_crossentropy',
optimizer='adam', metrics=['accuracy'])
convolutionalNeuralNetwork.fit(x_train[selected_feature_names],
keras.utils.to_categorical(y_train,3), epochs=30, batch_size=32)
```

The convolutional model has one convolutional layer with 32 filters and a kernel size of 3, followed by a max pooling layer, a flatten layer, and two dense layers with 50 and 3 neurons respectively. The model is trained to minimize the categorical cross-entropy loss using the Adam optimizer, and accuracy is tracked during training as a metric. The model is trained for 30 epochs with a batch size of 32.

`convolutionalNeuralNetwork.add(MaxPooling1D(pool_size=2))` – This adds a max pooling layer to the CNN with a pool size of 2. Max pooling is a downsampling operation that reduces the dimensionality of the feature maps.

`convolutionalNeuralNetwork.add(Flatten())` - This adds a flatten layer to the CNN, which converts the 2D output from the previous layer to a 1D array.

Evaluation of model performance for Convolutional Network -

1. Confusion Matrix -

Figure 6. Confusion matrix for Convolutional Neural Network

From Figure 6 we can demonstrate that the 'Slight' category has 357 and 404 misclassified categories in 'Serious' and 'Fatal' respectively. The 'Fatal' category has 339 and 240 misclassified categories in 'Serious' and 'Slight'. Tackling false predicted categories can improve the model's performance significantly.

2. Performance Metrics -

At this point we have checked Prediction Accuracy, Precision, Recall and F1 Score to evaluate the performance of our model.

Performance Metrics	Score
Prediction Accuracy	0.7509433962264151
Precision Score	0.7380810869578234
Recall Score	0.7685993309001504
F1 Score	0.742809568224597

Table 2. Performance Metrics for Convolutional Network

Table 2 depicts that the convolutional network is achieving reasonably good accuracy, precision, recall, and F1 score. The accuracy score of 0.75 suggests that your model is correctly predicting the class of roughly three out of every four instances in your dataset.

3. Comparison with trivial baseline -

Our convolutional model is compared to the majority classifier. Majority classifier gives around 44.2658% of accuracy while the network performs at 74.2084%. Based on comparison our model is significantly outperforming the majority classifier. This means that our Convolutional Neural Network is correctly predicting the class of a much larger proportion of instances in your dataset than the majority classifier.

If the Convolutional Network is not able to outperform the baseline model, then it may suggest that the model is not effective or needs further improvement.

5. Ethical discussion

1. Privacy and Security: The dataset contains sensitive information about factors involved in road accidents. It's important to ensure that the models are trained and deployed in a way that respects the privacy and security of this information. The data should be protected from unauthorized access or disclosure and the models should not be used to infringe upon privacy.

2. Accountability: Ensuring accountability for the predictions made by the models is crucial. This involves establishing clear lines of responsibility and oversight for the use of the models. When the models are used to make decisions that have significant implications for people's lives, it's essential to have systems in place that guarantee the reliability and fairness of the models' predictions.

6. Recommendations

- Random Forest Classifier is the best candidate for the task as it is more time efficient than others and achieves relative accuracy of 74.4708%. The Convolutional Neural Network has a slightly higher accuracy, it is a more complex and resource-intensive model compared to the Random Forest Classifier. However, the Random Forest Classifier is more time-efficient and is easier to interpret and understand.

- Random Forest Classifier performs the best as our dataset does not have complex relationships and it is massive. Efficiency and interpretability are more important as different models do not make any significant improvements in accuracy. Convolutional Networks would only be the choice if it had achieved more than 85% accuracy.

- Potential areas of the future improvements can be some external factors such as time of day, road construction, vehicle insurance. Taking these additional details into account can have an impact on accuracy.

7. Retrospective

If I were to start the whole coursework again I would broadly investigate the factor that would take accuracy to 90% and ensure efficiency, interpretability, and the model is capturing complex relationships.

8. References

1. Singh, V., Pencina, M., Einstein, A.J. et al. Impact of train/test sample regimen on performance estimate stability of machine learning in cardiovascular imaging. Sci Rep 11, 14490 (2021). https://doi.org/10.1038/s41598-021-93651-5

2. www.ibm.com. (n.d.). F value. [online] Available at: https://www.ibm.com/docs/no/cognos-analytics/11.1.0?topic=terms-f-value