HiLCoE
School of Computer Science & Technology

Title - Neural Network Assignment

Course - Artificial Intelligence

Section - DRB2202C

Group members

1. **Tsedalu Ashenafi** (MI1194)
2. **Yahwenessi Elias** (JS5191)
3. **Semir Sultan** (Kf8256)
4. **Wakene Tamiru** (ST9884)
5. **Imran Getu** (EH6031)

Introduction

The main goal of this project is to build a reliable neural network classification model. The model will uses Crash Data data set to predict 'Accident Type' based on different factors.

Data set: The data set used for this project contains information about vehicle crashes. It includes a wide range of features such as driver details, vehicle information, road conditions, and environmental factors.

Methodology: The project follows a structured methodology:

1. **Data Understanding:** Initial exploration of the dataset to understand its structure, identify data quality issues, and uncover insights.
2. **Data Preparation:** Cleaning and preprocessing the data to make it suitable for a neural network model. This includes handling missing values, encoding categorical features, and scaling numeric features.
3. **Model Design:** Architecting a neural network with appropriate layers, neurons, and activation functions.
4. **Training:** Training the model on the preprocessed data, using techniques to monitor performance and prevent overfitting.
5. **Evaluation:** Assessing the model's performance on a separate test set using various classification metrics.
6. **Interpretation:** Analyzing the results to understand the model's strengths and weaknesses and suggest areas for improvement.

Initial Findings: A preliminary analysis of the data set was conducted to understand its basic characteristics.

- **Structure:** The dataset originally contains **60,004 rows** and **54 columns**. The features are a mix of numeric (22) and categorical (31) data types, providing a comprehensive view of each incident.
- **Missing Values:** The data set contains a significant number of missing values. Several columns, such as 'Zone', 'Driving License', 'Exiting/entering', 'Contributory Action', and 'Region', are **100% empty** and were dropped. Other columns like 'Victim-3 Movement' (99.8% missing) and 'Victim-2 Movement' (99.5% missing) also have very high percentages of missing data. A strategy of dropping columns with more than 30% missing data was adopted. For the rest, imputation was performed using the median for numerical features and the mode for categorical features.
- **Duplicates:** The data set was checked for duplicate rows, and **no duplicate entries** were found. This indicates good data integrity in terms of redundant records.

Visualisations:

➢ All visualizations are found in the project folder in the "outputs/visualizations " folder.

- **Class Distribution:** The distribution of the target variable, 'Accident Type', is imbalanced. This is a crucial observation as it can bias the model towards the majority class. The visualization class_distribution.png illustrates this imbalance, which needs to be addressed during the modeling phase, possibly by using techniques like stratified sampling or class weights.
- **Feature Distributions:** The distributions of various numeric and categorical features were examined using histograms and bar charts, available in feature_distributions.png. This helped in understanding the range and frequency of different values for each feature.
- **Correlation Analysis:** A correlation heatmap correlation_heatmap.png was generated to understand the relationships between numeric features. The top correlations are listed in reports/top_correlations.csv. A notable correlation of **0.63** was found between 'Driver age' and 'Driver experiance(years)', which is expected.
- **Outliers:** Boxplots boxplots_outliers.png were used to identify outliers in the numeric features. Outliers can skew the model's learning process, and a capping strategy using the Interquartile Range (IQR) was applied to handle them.

Preprocessing Requirements::

- **Handling Missing Values:** Dropping columns with high percentages of missing data and imputing the rest.
- **Encoding Categorical Variables:** Converting categorical features into a numeric format using one-hot encoding, label encoding, or frequency encoding based on the cardinality of the feature.

- **Feature Scaling:** Normalizing the numeric features to a standard scale to ensure that features with larger ranges do not dominate the learning process.
- **Handling Imbalance:** Using stratified splitting to maintain the class distribution in the training, validation, and test sets.
- **Outlier Treatment:** Capping outliers to reduce their influence on the model.

Preprocessing Steps:

Missing Value Handling:

- Columns with more than 30% missing data were dropped. This included columns like 'Zone', 'Driving License', and several 'Victim' related features.
- For the remaining columns, missing numeric values were imputed using the **median**, while missing categorical values were filled with the **mode**. This approach is robust to outliers and prevents data loss.

Categorical Variable Encoding:

- A mixed-encoding strategy was used to handle the categorical features:
- **Binary features** (2 unique values) were encoded using **Label Encoding**.
- **Low cardinality features** (up to 10 unique values) were transformed using **One-Hot Encoding**. This creates new binary columns for each category, avoiding an arbitrary order.
- **High cardinality features** (more than 10 unique values) were encoded using **Frequency Encoding**. This replaces each category with its frequency of occurrence, which can be a useful signal for the model.
- **Feature Scaling :** All numeric features were scaled using **StandardScaler**. This standardizes the features to have a mean of 0 and a standard deviation of 1, ensuring that all features contribute equally to the model's training process.
- **Outlier Handling :** Outliers in numeric features were handled by **capping** them at the 1.5 * IQR (Interquartile Range) bounds. This reduces the skewing effect of extreme values without removing the data points entirely.

Data Split:

The preprocessed data set was split into three sets to ensure a robust evaluation of the model:

- **Training Set :** 70% of the data, used to train the model.
- **Validation Set :** 15% of the data, used to tune hyper parameters and monitor for overfitting during training.
- **Test Set :** 15% of the data, kept separate and used for the final evaluation of the model's performance.

The split was performed with stratification on the target variable ('Accident Type') to ensure that the class distribution was preserved across all three sets. The processed data files are stored in the outputs/processed_data/ directory.

Model Architecture

The neural network was designed to be a deep, regularized model suitable for this classification task.

Network Design:

The model consists of the following layers:

- **Input Layer:** The input layer accepts a vector of size corresponding to the number of features after preprocessing.
- **Hidden Layers:** The network has **three hidden layers** with a decreasing number of neurons: **64, 32, and 16**. This "funnel" structure is designed to learn hierarchical representations of the data, from general features in the first layer to more specific patterns in the deeper layers.
- **Activation Function:** The **ReLU (Rectified Linear Unit)** activation function is used in all hidden layers. ReLU is computationally efficient and helps mitigate the vanishing gradient problem.
- **Output Layer:**
- The output layer has a number of neurons equal to the number of classes in the target variable.
- The **Softmax** activation function is used, which outputs a probability distribution over the classes, making it ideal for multi-class classification.

Justification for Design Choices:

- **Depth and Width:** The choice of three hidden layers provides a good balance between model capacity and the risk of overfitting. The decreasing number of neurons helps in compressing the information and extracting the most salient features.
- **Regularization:** To prevent overfitting, several regularization techniques were employed:
- **L2 Regularization (λ=0.001):** Applied to the weights of the dense layers to penalize large weights and encourage the model to learn simpler patterns.
- **Dropout (Rate=0.3):** Applied after each hidden layer to randomly set a fraction of neuron activations to zero during training. This forces the network to learn more robust features.
- **Batch Normalization:** Used after each dense layer (except the last one) to stabilize the learning process and speed up convergence by normalizing the inputs to each layer.
- **Optimizer and Loss Function :** The **Adam optimizer** was chosen for its adaptive learning rate capabilities, which performs well in most scenarios.
- The **sparse categorical cross entropy** loss function was used, as it is designed for multi-class classification problems where the labels are provided as integers.

Training Process

The model was trained using a comprehensive strategy to ensure robust learning and to mitigate overfitting. The training process was performed on the training set (70% of the data), with performance monitored on the validation set (15%).

Training Parameters:

- **Loss Function:** sparse categorical cross entropy was used, which is appropriate for multi-class classification with integer-encoded labels.
- **Optimizer:** The **Adam optimizer** was employed with an initial learning rate of **0.001**.
- **Batch Size:** A batch size of **32** was used, which provides a good balance between computational efficiency and stable gradient estimation.
- **Number of Epochs:** The model was set to train for a maximum of **100 epochs**.

Overfitting Mitigation:

Several techniques were used to prevent the model from overfitting to the training data:

- **Early Stopping:** An early stopping callback was configured with a patience of **15 epochs**. This callback monitors the validation loss and stops the training process if no improvement is seen for 15 consecutive epochs, restoring the weights from the best-performing epoch.
- **Reduce Learning Rate on Plateau:** The learning rate was dynamically adjusted during training. If the validation loss did not improve for 7 epochs (half the early stopping patience), the learning rate was reduced by a factor of 0.5. This allows the model to make finer adjustments to the weights as it approaches a minimum.
- **Dropout and L2 Regularization:** As described in the model architecture, dropout and L2 regularization were key components in preventing overfitting.

Training Performance:

The training process was monitored by tracking the loss and accuracy on both the training and validation sets. The training_history.png plot shows these curves.

The training and validation accuracy curves track each other closely, indicating that the model is generalizing well. The loss curves show a steady decrease, and the gap between the training and validation loss is small, which further suggests that overfitting was successfully controlled. Early stopping was triggered, which prevented the model from continuing to train unnecessarily.

Evaluation

After training, the model's performance was evaluated on the unseen test set (15% of the data). This provides an unbiased assessment of the model's ability to generalize to new data.

Performance Metrics:

The model achieved a high level of performance on the test set. The key metrics are as follows:

- **Accuracy: 96.7%**
- **Weighted Precision: 96.6%**
- **Weighted Recall: 96.7%**
- **Weighted F1-Score: 96.6%**

Confusion Matrix :

The confusion matrix, visualized in confusion_matrix.png , provides a detailed breakdown of the model's performance for each class. It shows the number of true positive, true negative, false positive, and false negative predictions. The strong diagonal in the matrix indicates a high number of correct predictions for all classes.

Per-Class Metrics:

The performance for each individual class is detailed in per_class_metrics.png. This is particularly important for an imbalanced data set. The model performs very well on the majority classes like 'Minor' and 'PDO'. For some of the minority classes, the performance is lower, which is expected. For example, the 'Fatal' class has a lower F1-score compared to the majority classes, which suggests that the model has more difficulty correctly identifying this class.

Cross-Validation :

The file cross_validation_results.png shows the results of a 5-fold cross-validation. The consistent performance across the folds demonstrates that the model is robust and not overly sensitive to the specific train-test split. The average validation accuracy across the folds is high, which further validates the model's performance.

Model Strengths and Weaknesses:

Strengths:

- The model demonstrates excellent performance on the majority classes, such as 'Minor' and 'PDO' accidents, achieving high precision and recall. This is likely due to the abundance of data for these classes.
- The overall accuracy of 96.7% on the test set is a strong indicator of the model's predictive power.
- The use of regularization techniques successfully prevented overfitting, leading to good generalization on unseen data.

Weaknesses:

- The model struggles with the minority classes. For instance, the 'Fatal' accident class, while critically important, has a lower F1-score. This is a common issue with imbalanced datasets.
- The confusion matrix reveals that some 'Fatal' accidents are misclassified as 'Serious'. This is a critical error that needs to be addressed.
- The analysis.png highlights that the model's errors are concentrated in the minority classes, where the decision boundary is harder to learn due to the limited number of examples.

Prediction Distribution:

The prediction_distribution.png shows that the model's predictions are heavily skewed towards the majority classes, mirroring the class distribution in the training data. This is a direct consequence of the class imbalance. While the model is accurate overall, it is less sensitive to the rare but often more critical event types.

Summary of Findings:

This project successfully demonstrated the development of a neural network model for classifying accident types from the data set. The final model achieved a high accuracy of **96.7%** on the test set, proving its effectiveness. The project covered all the essential stages of a machine learning pipeline, including data preprocessing, model design, training, and evaluation. The chosen architecture, combined with robust regularization techniques, resulted in a model that generalizes well.

However, the analysis also highlighted the challenge of working with imbalanced data, as the model's performance is weaker on the minority classes.

Recommendations for Improvement:

- **Advanced Imbalance Handling:** Implement more advanced techniques to handle the imbalanced data set
- **Feature Engineering:** Explore the creation of new features from the existing data.
- **Hyper parameter Tuning :** Conduct a more systematic hyper parameter search using techniques like **Grid Search** or **Random Search** to find the optimal combination of learning rate, batch size, number of layers, and neurons.
- **Alternative Architectures :** Experiment with different model architectures, such as a wider network or a deeper network.

Overall Workflow Reflection:

The end-to-end workflow from data loading to evaluation was a valuable learning experience. It highlighted the importance of a systematic approach, starting with a thorough understanding of the data, followed by careful preprocessing and a well-justified model design. The iterative process of training, evaluating, and analyzing the results was crucial for building a high-performing model. This project serves as a strong foundation for tackling similar classification problems in the future.