ESCUELA POLITÉCNICA NACIONAL

DEPARTAMENTO DE INFORMÁTICA Y CIENCIAS DE LA COMPUTACIÓN

PREDICTION OF ACCIDENT RISK LEVELS IN TRAFFIC ACCIDENTS USING DL AND RBF NEURAL NETWORKS APPLIED TO A DATASET WITH INFORMATION ON DRIVING EVENTS

THESIS WORK PRIOR TO OBTAINING THE MASTER'S DEGREE IN COMPUTER SCIENCE.

ARCINIEGAS AYALA CRISTIAN VINICIO
cristian.arciniegas@epn.edu.ec

DIRECTOR: Ph.D. Myriam Hernández Álvarez
myriam.hernandez@epn.edu.ec

CODIRECTOR: M.Sc. Pablo Marcillo Lara
pablo.marcillo@epn.edu.ec

Quito, April 2024

Approval

We certify that this thesis work *Prediction of accident risk levels in traffic accidents using DL and RBF neural networks applied to a dataset with information on driving events* was developed by Arciniegas Ayala Cristian Vinicio, student of the Master's Degree in Computer Science with mention in Intelligent Systems, under our supervision.

Ph.D. Myriam Hernández Álvarez

DIRECTOR

M.Sc. Pablo Marcillo Lara

CODIRECTOR

Declaration of authorship

I, Arciniegas Ayala Cristian Vinicio, declare under oath that the work presented here is my own, that it has not been previously presented to obtain any degree or professional title, and that all bibliographical references included in this document were reviewed.

The Escuela Politécnica Nacional may use the respective rights to this work according to the provisions of the Intellectual Property Law, its Regulations, and the Institutional Rules in force.

Arciniegas Ayala Cristian Vinicio

This work is dedicated first to God for allowing me to study and face this challenge and to my wife and daughter, who have always supported me, even in the most difficult moments.

Acknowledgements

I am grateful to the Escuela Politécnica Nacional for accepting me into this master's program. I am grateful to my professors, classmates, and those who guided and supported me throughout this thesis work, enabling me to complete it successfully. Thank you all very much!

Contents

List of Acronyms

ACC	Accuracy
ANN	Artificial Neural Network
AUC	Area Under the Curve
BN	Bayesian Network
CNN	Convolutional Neural Networks
DBN	Deep Belief Network
DL	Deep Learning
DNN	Deep Neural Networks
FFN	Feed-Forward Network
GBRT	Gradient Boosted Regression Trees
GPC	Gaussian Process Classifier
LR	Logistic Regression
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MI	Mutual Information
MLPNN	Multilayer Perceptron Neural Network
MLP	Multilayer Perceptron
ML	Machine Learning
NB	Naive Bayes
RBFNN	Radial Basic Functions Neural Networks
RBF	Radial Basis Functions
ReLU	Rectified Linear Unit
RF	Random Forest
RNN	Recurrent Neural Networks
SEN	Sensitivity
SOM	Self-Organizing Map
SPE	Specificity
SVC	Support Vector Classification

List of Figures

List of Tables

List of Appendices

Resumen

El aprendizaje profundo o *deep learning* (DL) debe trabajarse offline porque las fases de entrenamiento y ejecución se procesan por separado. Este proceso suele requerir diferentes computadores debido a los requisitos para obtener los modelos. Una limitación de este enfoque es que los datos de entrenamiento no pueden incorporarse continuamente a la base de conocimientos porque es necesario repetir un proceso de entrenamiento largo y complejo para obtener nuevos modelos. Aunque el entorno no sea estático, es crucial entrenar dinámicamente los modelos integrando nueva información durante la ejecución. Las redes neuronales de función de base radiales (RBFNN) permiten un aprendizaje continuo gracias a su alta velocidad de aprendizaje y a su arquitectura sencilla. Estas redes se alimentan con una capa de entrada que contiene datos transformados mediante funciones de base radial (RBF). Así, las RBFNN permiten una interpretación directa de la función, pues cada nodo sólo tiene una capa oculta. Esta característica puede suponer una ventaja para una mejor comprensión y control de la red.

Este estudio analizó un enfoque de conjunto de datos dinámico añadiendo nuevos datos al entrenamiento sobre la marcha, debido a los constantes cambios en los datos del conductor, la información del vehículo, las condiciones ambientales y los accidentes de tráfico. El conjunto de datos utilizado para predecir los niveles de riesgo de accidentes, se obtuvo a partir de trabajos previos de doctorado y maestría de la Escuela Politécnica Nacional el cual contiene información sobre eventos de conducción.

Finalmente, este trabajo compara el tiempo de procesamiento y el desempeño de Redes Neuronales Convolucionales (CNN) con varios algoritmos de Aprendizaje Automático (ML), incluyendo RBF, Perceptrón Multicapa (MLP) y Random Forest (RF), utilizando métricas de evaluación de precisión, especificidad y sensibilidad-recall. Los resultados ofrecen recomendaciones para nuevos modelos de predicción de accidentes.

Palabras clave: Aprendizaje Profundo, Redes neuronales de Función de Base Radial, Predicción de accidentes de tránsito, Aprendizaje automático.

Abstract

Deep learning (DL) must be worked offline because the training and execution phases are processed separately. This process often requires different computers due to the requirements to obtain the models. A limitation of this approach is that the training data cannot be continuously incorporated into the knowledge base because a long and complex training process needs to be repeated to obtain the new models. Although the environment may not be static, it is crucial to dynamically train models by integrating new information during execution. Radial Basic Functions Neural Networks (RBFNN) allow for continuous learning due to their high learning speed and simple architecture. These networks are fed with an input layer containing data transformed using Radial Basis Functions (RBF). Thus, the RBFNN allows for a direct interpretation of the function, with each node having only one hidden layer. This characteristic can provide an advantage in better understanding and control over the network.

The study analyzed a dynamic dataset approach by adding new data to the training on the fly, given the constant changes in the driver's data, vehicle information, environmental conditions, and traffic accidents. The dataset used to predict accident risk levels was obtained from previous doctoral and master's works at Escuela Politécnica Nacional that contained information on driving events.

Finally, this study compares the processing time and performance of Convolutional Neural Networks (CNN) with several Machine Learning (ML) algorithms, including RBF, Multilayer Perceptron (MLP), and Random Forest (RF), using evaluation metrics of accuracy, specificity, and sensitivity-recall. The results provide recommendations for new accident prediction models.

Keywords: Deep Learning, Radial Basis Function Neural Network, Prediction Traffic Accidents, Dynamic Learning.

Chapter 1

INTRODUCTION

Road traffic deaths and injuries continue to pose a significant challenge to global health and development. According to WHO's Global Status Report on Road Safety 2023 [1], road traffic crashes are the leading cause of death among children and adolescents aged 5 to 29 years. In 2021, an estimated 1.19 million people died due to road traffic accidents, which is a 5% decrease from the 1.25 million deaths recorded in 2010. Even with the global motor vehicle fleet doubling, there has been a slight overall reduction in deaths. Despite this, the cost of mobility remains excessively high. Nine out of ten deaths occur in low- and middle-income countries, while individuals in low-income countries continue to face the highest risk of death per capita.

In 2023, Ecuadorian National Transit Agency (ANT) recorded 20,994 traffic accidents nationwide. Of these, 20.78% involved cars alone, resulting in 18,605 injuries and 2,373 fatalities. Among cities with the largest populations, Guayaquil had the highest traffic accidents, with 4,402, followed by Quito, with 3,816 accidents [2].

These facts motivated the search for practical solutions to prevent more lives from being lost due to traffic accidents. An interesting proposal, mentioned by Ren et al. [3], is to use the large flow of traffic data that can be obtained and, through the use of Deep learning (DL) and Artificial neural networks (ANN), develop predictive models to reduce the risk levels of traffic accidents, which can be implemented in effective risk warning systems for drivers. However, it is important to note that obtaining a good prediction accuracy of the risk of traffic accidents is complicated because it is related to several factors, like weather or road conditions, which affect the effectiveness [4]. Another reason is that various conditions differ from one region to another. Tritat and Lee [5] mentioned that predicting traffic accident risk remains challenging due to many factors contributing to accidents, including the number of vehicles on the road and external conditions like weather, road conditions, ambient lighting, and time of day. They also indicated that recent studies have attempted to combine various factors using complex models to make better and more precise predictions.

ML methods have been extensively utilized in traffic prediction problems, allowing the prediction of multiple crash injuries using data that includes different causes and factors from

events on roads and streets [6]. Several studies [7], [8] have explored and analyzed various types of ANN and concluded that the Multilayer Perceptron Neural Network (MLPNN) is the most commonly used ANN for predicting road accidents. They also found that in some cases, RBFNN has better predictive performance than MLPNN, but this difference could be due to several factors.

However, Ye et al. [9] state that predicting traffic accident risk requires much data. Therefore, many researchers have turned to DL to develop models for accident risk analysis. Modern DL networks usually consist of tens or hundreds of successive layers to discover complex structures in high-dimensional data and to extract hierarchical representations in feature learning [10].

DL has been called *the technology that will change the world* [11]. In addition to breaking records in image recognition and speech recognition competitions, DL has produced many interesting results in a variety of tasks, for example, natural language processing and, in particular, topic classification, sentiment analysis, question-answering systems, and machine translation. Also, Tian and Zhang [11] mentioned that Recurrent Neural Networks (RNN) and CNN are the most widely used deep learning models. The Long Short-Term Memory (LSTM) is also applied to many diverse learning problems that differ significantly in their scale and nature from the initially tested problems [12]. An LSTM model can store previous data and predict future risk trends, making it widely applicable in risk forecasting.

On the other hand, it is crucial to note that RBFNN is a conventional Feed-forward network (FFN) variety [13], which is a universal approximation function. It is worth mentioning that RBF has greater precision in describing the relationships between risk factors and accident frequency. Moreover, the network structure, primarily denoting the number of nodes in hidden and input layers, is a crucial aspect of neural network model development, given its significant impact on generalization performance. RBFNN proved to have a significant advantage in approximating, classifying, and speeding up processes [14].

1.1 Motivation

This study aims to prove that RBFNN learning is faster when only three levels (input, hidden layer, output layer) are applied, allowing a dynamic dataset to be used under changing conditions. Moreover, predictions made using this second method are easily auditable, and the results can be comparable to those achieved with DL. A key objective of this work was to compare and evaluate these ML approaches in traffic accident risk prediction and the implications of using a dynamic dataset. We have used the term *dynamic dataset* [15] to refer to a process that incorporates new data of driving characteristics collected at specific time intervals from vehicle agents, processed using relevant mechanisms and algorithms, and finally added to the main dataset, making it changed. All this is done to acquire information on

driving events in a real-world environment where data flows continuously.

Thus, we aim to demonstrate that using an RBFNN allows a faster validation process to obtain new models, and its prediction performance could be comparable to other more complex ANN, for example, DL networks, when applied to solve problems of predicting accident risk levels using a dynamic dataset of driving event information.

1.2 Objectives

This thesis work proposes the following objectives:

- Predict traffic accident risk levels using different configurations of DL neural networks applied to a driving dataset.
- Predict traffic accident risk levels using the same driving dataset employing RBF models.
- Evaluate the performance of both approaches using quantitative metrics.
- Compare the results obtained and recommend each technique's use.

1.3 Hypothesis

An RBFNN with only three levels allows faster training to obtain models, and its performance is comparable to that of other ANNs that are more complex and use DL when applied to solving problems of predicting accident risk levels using a dataset of driving event information.

1.4 Scientific contributions

The scientific contributions are:

- Application of DL and RBFNN for predicting traffic accident risk levels using *POLIDriving*, a dataset with information on driving events.
- Evaluation of both approaches in performance and feasibility of application in dynamic learning using quantitative metrics.
- Presentation of the results in a scientific paper to be submitted to a high-impact scientific journal.

1.5 Thesis outline

This thesis follows the structure outlined below: Chapter 2 analyzes related works. Chapter 3 presents the theoretical framework. Chapter 4 describes the methods and materials used. Chapter 5 showcases the results of the comparison between neural networks and ML algorithms. Finally, Chapter 6 provides the conclusion and recommendations.

Chapter 2

RELATED WORK

Building an effective traffic accident risk prediction system is important in traffic accident prevention. However, predicting the risk of a traffic accident is difficult because many related factors are involved [3]. For that reason, several types of research have been developed to predict the risk of traffic accidents. This chapter presents related work that provide evidence on this topic.

2.1 Risk prediction in traffic accidents

Agarwal [16] developed an ML approach to predict accident risk by training a CNN with past accident data and Google Maps images of accident-prone road segments. The approach unlocks the precise interactions of small road features that contribute to higher accident risk. The study found that the model achieved 93% for precision, 94% for recall, and 0.86 for F1-score, the highest values observed in this study.

Kumeda *et al.* [17] applied several classification algorithms to a dataset to classify three categories of injuries: fatal, severe, and minor. The results showed that Fuzzy-FARCHD achieved the highest accuracy at 85.94%, followed by RBFNN at 84.14%, Random Forest (RF) at 83.42%, Naive Bayes (NB) at 80.90%, and MLPNN at 79.27%.

Moosavic *et al.* [18] developed a DL network model called Deep Accident Prediction (DAP). This model includes several neural network-based components that use various data attributes to predict accidents, including traffic events, weather data, points of interest, and time information. The maximum F1-score achieved for DAP was 0.65 compared to other baseline models.

Zhao *et al.* [19] proposed a traffic accident risk forecasting algorithm based on DL for the edge-cloud internet of vehicles. Real-time traffic data was collected and entered into a CNN to extract features. The output of the CNN was then classified for features in an RF, enabling the prediction of traffic accident risks. The proposed algorithm achieved an Area Under the Curve (AUC) of 0.9921, better than the CNN-based model with an AUC of 0.9478.

Huang *et al.* [20] investigated the feasibility of utilizing DL models to identify and forecast crash occurrence. They utilized volume, speed, and sensor occupancy data obtained from roadside radar sensors to create a feature set for the DL models. The findings indicate that the deep model outperforms other models in crash detection and performs similarly in crash prediction. This study utilized a CNN and achieved an accuracy of 77.34% and an F1-score of 0.7651 with the best configuration.

Lee *et al.* [21] developed a risk-level accident classifier using a deep-learning method to predict high-risk taxi drivers. The study was composed of several stages. In stage 0, data on healthy taxi drivers were collected. In stage 1, an RF analysis was used to identify the factors that affect the risk level of accident severity experienced by these drivers. In stage 2, an FFNN classifier was optimized to predict the severity of the risk level of the accident. In stage 3, the optimal model for predicting high risk was selected. Thus, the proposed FFNN model achieved an accuracy 86% and an F1-score of 0.77.

Li *et al.* [22] proposed a real-time crash risk prediction model for arterials using LSTM and CNN. The model could learn from various features, including traffic flow characteristics, signal timing, and weather conditions. Experiments suggested that the proposed model outperforms the sensitivity of CNN (64%) and LSTM (80%) when operating separately. The results indicated the promising performance of using LSTM and CNN to predict real-time crash risk on arterial roads. The achieved sensitivity was 88%, while the false alarm rate was 12%. Additionally, the AUC value was 0.93.

Wang *et al.* [23] developed the Model-level Dynamic Fusion Neural Network (Model-DFNN), which combines satellite imagery and spatiotemporal urban big data in an adaptive way to calculate the risk levels of traffic accidents throughout the city, resulting in a risk map that generated a visual guide when creating emergency response plans. They evaluated the performance of different models; the accuracy obtained was RBF kernel 71.8%, RF 73.2%, and Model-DFNN 83%, respectively.

Purkrábková *et al.* [24] conducted a study on the classification of traffic accident risk in urban areas. Their objective was to propose effective traffic management solutions to minimize social losses in cities with high traffic volume. Using an available dataset with several sources of traffic, meteorological data, and other related data, the authors modeled an MLPNN to address the specific traffic problem. The model achieved an accuracy of 89%, compared to 75.4% for the RF model.

Brühwiler *et al.* [25] investigated how the integration of different geographic context sets, like weather conditions, points of interest, and land use, can improve various machine learning risk assessment models for discriminating between crash and accident-free drivers. The authors

evaluated the predictive performance of five machine learning classifiers, including Logistic Regression (LR), RF, XGBoost, FFNN, and LSTM. The better accuracies achieved by these models were 75.2%, 75.7%, 75.5%, 75.4%, and 55.3%, respectively. The LSTM model did not outperform the other approaches, possibly due to insufficient training data.

Lin *et al.* [26] developed a model to predict high-risk intersections for traffic accidents and prioritize areas for improvement. By analyzing the number of crashes and fatalities, this study attempts to estimate the relative risk level of individual intersections, determine the key risk factors, and establish an intersection risk prediction model to predict the probability and severity of future accidents. The study compared and evaluated the following neural networks: the better accuracy obtained was NB 71.84%, Decision Tree C4.5 (C4.5) 72.64%, Bayesian Network (BN) 71.81%, MLP 71.94%, Deep Neural Networks (DNN) 72.92%, Deep Belief Network (DBN) 72.62%, and CNN 72.62% for creating an accident risk prediction model.

Kaffash *et al.* [27] presented a hybrid predictive model for estimating the risk of traffic accidents using a generalized FFNN and RBFNN tuned with a Self-Organizing Map (SOM). This predictive model, which incorporates 22 different predictor features, estimated the risk of road accidents. A quality assessment of the proposed approach for different scenarios showed that the predicted accident risk had a high level of accuracy: an average accuracy of 90.74%.

Park and Hong [28] proposed a risk prediction model that reflects the road's static and dynamic features, including its length, speed limit, traffic volume, altitude, and the azimuth of the sun. The MLP model was tested using relevant data and achieved an accuracy of 75% and a recall of 81%.

Wang *et al.* [29] developed prediction models for traffic crash risk potential using traffic-related data. They applied the model to 7 classifiers and obtained 84 classification models. Finally, they compared the performance of the models and proposed the optimal resampling algorithm and classifier for predicting expressway risk potential. The best models were RF, XGBoost, and Support Vector Machine (SVM), with accuracies of 80%, 80%, and 81%, respectively.

Using several features, Amorim *et al.* [30] conducted experiments with various ML algorithms to determine the best classifier for identifying severe or non-severe accident risks associated with Brazilian federal road hotspots. The used dataset includes the spatial footprint, weekday and time of the accident, road type, route, orientation, weather conditions, and accident type. They tested SVM, RF, and an XGBClassiffer. The accuracies of these approaches were 58.60%, 70.12%, and 71.25%, respectively. The MLPNN model yielded promising results, achieving an accuracy of 83%, a precision of 84%, a recall of 83%, and an F1-score of 0.82.

Jin and Noh [31] proposed a system based on deep learning to predict traffic accidents in urban environments and estimate the associated risk levels using the Accident Risk Index (ARI). They compared the accuracy of the proposed accident model (CNN-DNN) with SVM, LR, and MLP, achieving accuracies of 94%, 90%, 88%, and 90%, respectively.

2.2 Related work analysis

The related work propose approaches that use either a static dataset or a heterogeneous source dataset. However, these studies did not incorporate new data on the fly to train or test new models. Additionally, most of these works do not present the time used for execution and obtaining results.

It should be noted that the number of data sources used varied from 6 to 42. The primary data sources were *traffic accidents*, including the number of fatalities, injuries, and collisions resulting in casualties or fatalities. *Weather conditions*, *road infrastructure*, *driver information*, and *vehicle data* were also considered. Figure 2.1 presents the related work using the most common data sources.

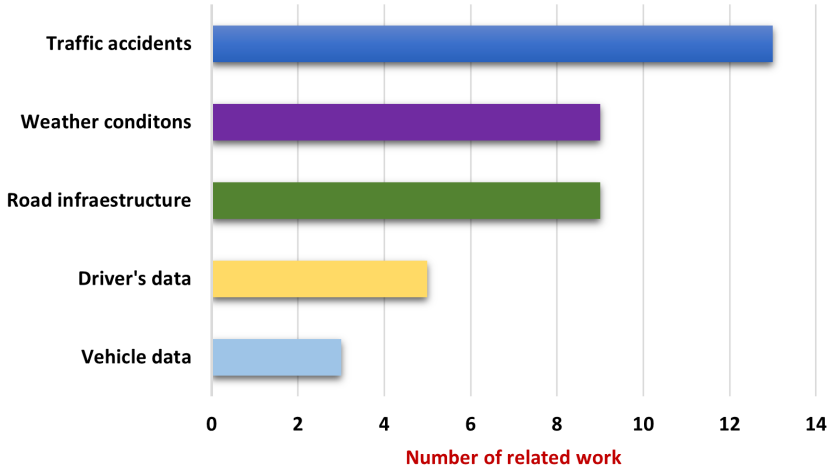


Figure 2.1: Number of related work that uses the most common data sources.

The ANN most frequently used in these studies were RF and MLPNN, followed by CNN, RBFNN, and LSTM. According to this analysis, the RBFNN is not commonly used in related work to predict the risk level of traffic accidents. Figure 2.2 shows the most ANN used in related work.

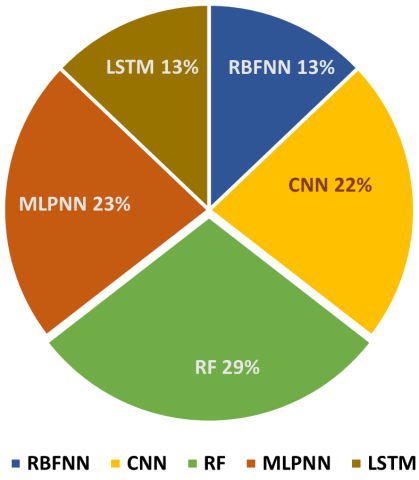


Figure 2.2: ANN and algorithms commonly used in related work.

The analyzed studies also showed that CNN and MLPNN achieved the highest accuracy of 93% and 90%, respectively, while RBFNN, RF, and LSTM achieved an accuracy of 84.14%, 83.42%, and 65%, respectively. Figure 2.3 displays the accuracy of all models used in the related work.

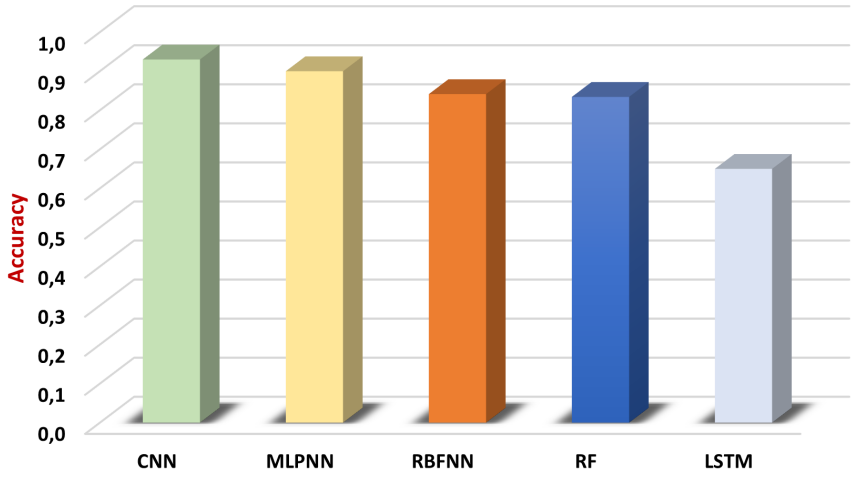


Figure 2.3: Accuracy obtained by ANN and algorithms in related work.

Half of the models in related work used binary classification, while the other half used multiclass classification; it is important to note that binary classification models yielded better results than multiclass classification. The accuracy may decrease when the number of predictor classes increases. Figure 2.4 shows the relationship between the number of predictor classes and accuracy.

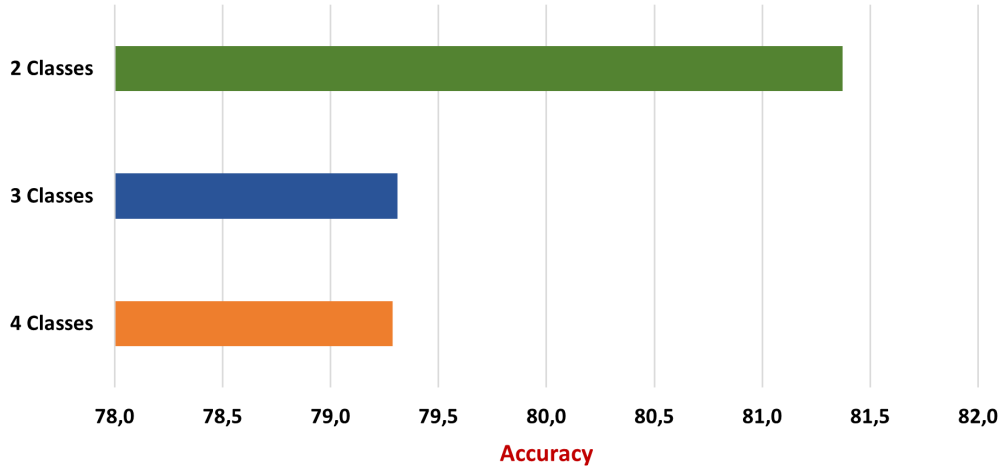


Figure 2.4: Accuracy according to the number of predictor classes.

Finally, Table 2.1 shows the resume of the analyzed related work.

Table 2.1: Resume of related work

Authors	Purpose	Accuracy/Results
Agarwal [16]	Develops a machine learning approach using CNN for predicting accident risk.	Precision: 93%, Recall: 94%, F1-score: 0.86
Kumeda et al. [17]	Compare different classifier algorithms using a dataset of crash injury categories.	Accuracy: Fuzzy-FARCHD 85.94%, RBFNN 84.14%, RF 83.42%, NB 80.90%, MLPNN 79.27%
Moosavic et al. [18]	Propose a deep neural network model called the DAP to predict the risk of a traffic incident.	F1-score: 0.65
Zhao et al. [19]	Propose a traffic accident risk forecasting algorithm based on deep learning for edge-cloud internet of vehicles.	AUC: 0.9921
Huang et al. [20]	Investigate the feasibility of utilizing deep learning models to identify and forecast crash occurrence.	Accuracy: 77.34%, F1-score: 0.7651
Lee *et al.* [21]	Develop a risk-level accident classifier using a deep-learning method to predict high-risk taxi drivers.	Accuracy: 86%, F1-score: 0.77
Li et al. [22]	Propose a real-time crash risk prediction model for arterials using LSTM and CNN.	Sensitivity: LSTM 80%, CNN: 64%, LSTM-CNN 88%

Table 2.1: Resume of related work (cont.)

Authors	Purpose	Accuracy/Results
Wang et al. [23]	Propose a model to predict the levels of risk for traffic accidents.	Accuracy: RBF 71.8%, RF 73.2%, DFNN-model 83%
Purkrábková et al. [24]	Conducted a study on the classification of traffic accident risk in urban areas. Their objective was to propose effective traffic management solutions to minimize social losses in cities with high traffic volume.	Accuracy: MLPNN 89%, RF 75.4%
Brühwiler et al. [25]	Evaluate machine learning risk assessment models: LR, RF, XGBoost, FFNN, and LSTM networks.	Accuracy: LR 75.2%, RF 75.7%, XGBoost 75.5%, FFNN 75.4%, LSTM 55.3%
Lin et al. [26]	Develop a model to predict high-risk intersections for traffic accidents.	Accuracy: DNN 72.62%, DBN 72.62%, MLP 71.94%, CNN 72.62%
Kaffash et al. [27]	Present a hybrid predictive model for estimating the risk of road accidents.	Accuracy: 90.74%
Park and Hong [28]	Propose a risk prediction MLP model that reflects the road's static and dynamic features: length, speed limit, traffic volume, altitude, and azimuth of the sun.	Accuracy: 75%, Precision: 73%, Recall: 81%
Wang et al. [29]	Develop prediction models for traffic crash risk potential using traffic-related data.	Accuracy: RF 80%, XGBoost 80%, SVM 81%
Amorim et al. [30]	Conduct experiments with various machine learning algorithms to determine the best classifier for identifying severe or non-severe accident risks associated with Brazilian federal road hotspots.	Accuracy: SVM 58.60%, RF 70.12%, XGBClassiffer 71.25%, MLPNN 83%
Jin and Noh [31]	Propose a system based on deep learning to predict traffic accidents in urban environments and estimate the associated risk levels.	Accuracy: SVM 90%, LR 88%, MLP 90%, CNN-DNN 94%

Chapter 3

THEORETICAL FRAMEWORK

This chapter presents the most important conceptual components of the thesis work, focusing on describing the ML algorithms, the ANN used, and the applied techniques.

3.1 Deep Learning

ML is a field of computer science that aims to provide machines with intelligence to perform tasks very similar to those performed by humans [32]. ML includes several approaches, such as supervised learning (SL), unsupervised learning (UL), reinforcement learning (RL), and DL. SL allows for identifying relationships and dependencies from input data or features and predicting output values through their predictor class labels. UL involves discovering new information from a dataset without a label or predictor class. RL aims to learn how to perform a sequence of actions in an environment that maximizes an agent's rewards for a given task. DL is primarily based on ANN, designed to simulate the functioning of the human brain. Figure 3.1 shows the taxonomy of ML approaches [32].

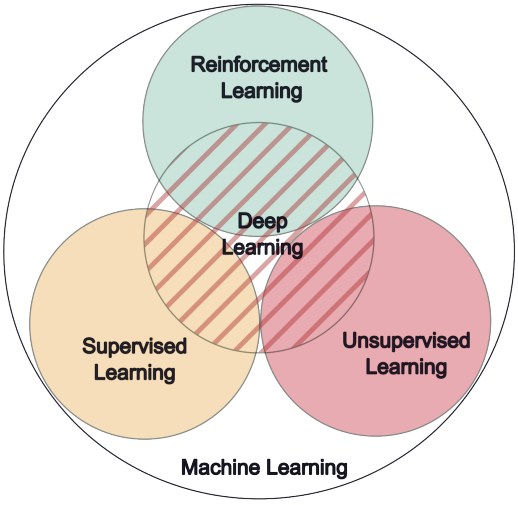


Figure 3.1: Machine Learning approaches.

DL represents a new line of research in the field of ML [33]. It is a complex algorithm that has achieved good speech and image recognition results and solves complex problems in pattern

recognition. DL has enabled machines to imitate various human activities, like seeing, hearing, and even thinking. For this reason, many researchers [9], [5] have chosen this approach to develop risk assessment models.

3.2 Rectified Linear Unit

Rectified Linear Unit (ReLU) is an activation function widely used that adds non-linearity to DL models and resolves the vanishing gradients problem [19]. It ranks among the most commonly used activation functions in DL. The Equation 3.1 defined the ReLU activation function.

$$ReLU(x) = \begin{cases} x(x > 0) \\ 0(x \leq 0) \end{cases}. \tag{3.1}$$

3.3 Convolutional Neural Networks

CNN is utilized for computer vision and classification tasks and is effective for processing data with various dimensionalities based on the convolution operations applied (1D, 2D, or 3D) [7]. The CNN network comprises four main operations: convolution, pooling or subsampling, non-linearity, and classification. These operations form the building blocks of the CNN network. The purpose of the convolution layer is to convolve the input features and include a bias [19]. The calculation of the convolutional layer is shown in Equation 3.2.

$$S(i,j) = (X^*W)(i,j) = \sum_m \sum_n x(i+m, j+n)^* w(m,n) + b, \tag{3.2}$$

where X is the input feature, W is the convolutional kernel, and b is the bias.

One crucial feature of this neural network is its output with fully connected layers. The primary reason for this implementation is its ability to classify the features of the input data into different categories within the training dataset. Figure 3.2 shows a CNN model architecture.

3.4 Random Forest

The RF algorithm is a simple ML classification method that can produce accurate results without complicated hyperparameter tuning [17]. RF builds multiple decision trees (DT) and combines them to achieve high and accurate predictions. Figure 3.3 shows the correlation between a DT and an RF. A DT takes on a tree-like structure that enables data classification

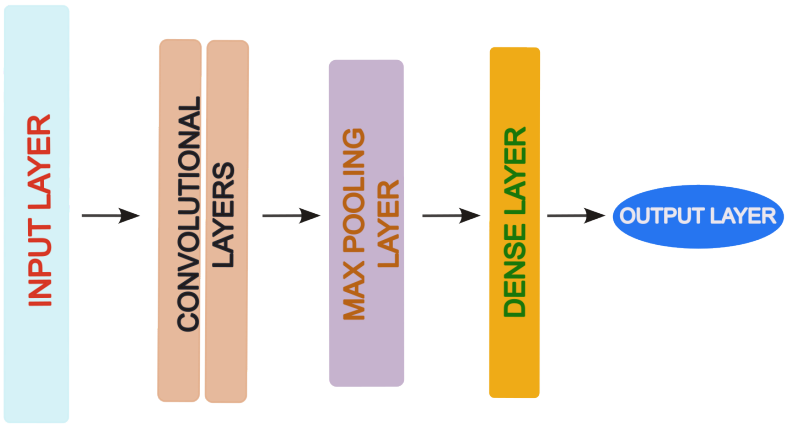


Figure 3.2: CNN Architecture.

[33]. The main goal of a DT is the condensation and organization of classification rules within a training dataset.

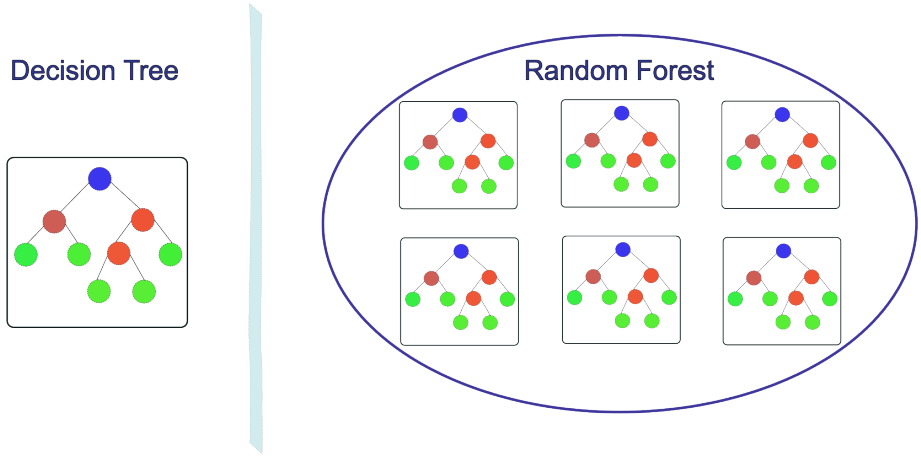


Figure 3.3: Correlation between Decision Tree and Random Forest.

Additionally, RF can be applied to classification and regression problems in current ML systems. It is a tree-based model, and its classification method is non-linear [29].

The RF classifier utilizes the Gini Index (GI) metric for attribute selection [30]. This index measures the impurity of an attribute concerning classes. When selecting a case x randomly from a given training set A and indicating that it belongs to a certain class C_i, the GI is described in Equation 3.3.

$$\sum \sum_{j \neq i} (f(C_i, A)/|A|)(f(C_j, A)/|A|), \tag{3.3}$$

where $f(C_i, A)/|A|$ represents the probability that the selected case x belongs to class C_i.

The RF classifier has an advantage over other decision tree methods because when using new training data, adult trees are not pruned whenever a tree reaches maximum depth.

3.5 Radial Basis Function

The RBFNN is a type of FFNN [34], and it has a simple structure, concise training, and quick convergence, enabling it to approximate any non-linear function [14]. For these reasons, RBFNN has a wide range of applications. The RBFNN structure consists of an FFNN with three layers, including a single hidden layer [35]. The first layer is the input layer, composed of the signal source node. The second layer is the hidden layer, and the type of problem determines the number of hidden units. The transfer function of the hidden units is a non-linear function with radial symmetry and a non-attenuating negative relationship with the center. The third layer is the output layer, which responds to the input mode.

The RBFNN is known for improved prediction efficiency and more stable results [14]. RBFNN assists in quickly finding suitable prediction models that RBFNN needs tuning fewer parameters. The RBF networks frequently train more rapidly than back-propagation networks [8]. These networks are comparatively resilient against non-stationary inputs due to the behavior of their hidden units employing radial basis functions. The Gaussian function is considered the basis function of the RBF network. Therefore, the representation of the general formula for the RBF output is described in Equation 3.4.

$$y(x) = \sum_{i=1}^{M} w_i e^{\left(\frac{-(||x-c_i||)^2}{2\sigma^2}\right)}, \tag{3.4}$$

where, input, output, center, width, and number of basis functions centered at c_i are denoted by x, $y(x)$, c_i, σ, and M, respectively. Similarly, weights are denoted by w_i. Figure 3.4 shows a RBF architecture [35].

3.6 Multilayer Perceptron

An MLPNN is a type of FFNN that functions sets of input data onto a set of fitting outputs. It consists of an input layer, a hidden layer, and an output layer [17]. Each hidden layer comprises a set of neurons that receive data from the previous layer, add weight, and then pass it through a non-linear activation function. The first hidden layer receives data from the

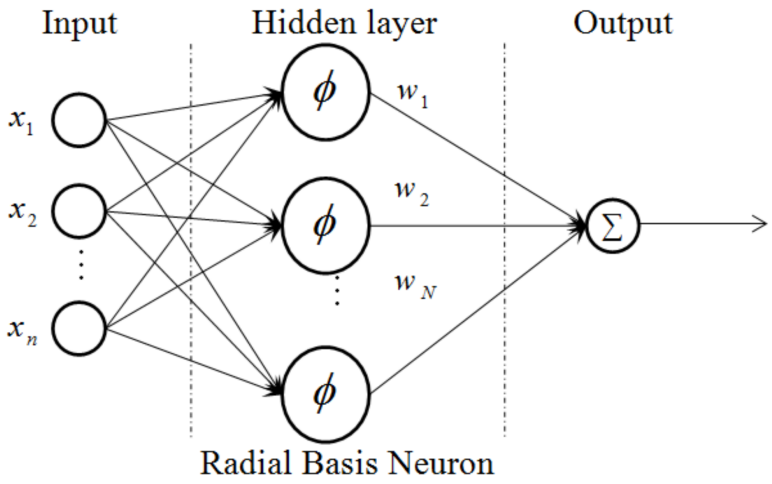


Figure 3.4: RBF Architecture.

input layer, and each successive layer follows suit. The output layer receives the last hidden layer's output, adds weight, and passes it through a non-linear activation function to produce the target output. The most commonly used technique for solving non-linear problems today is MLPNN [8]. Figure 3.5 displays the straightforward architecture of an MLPNN [8], where x_i indicates the input for the network.

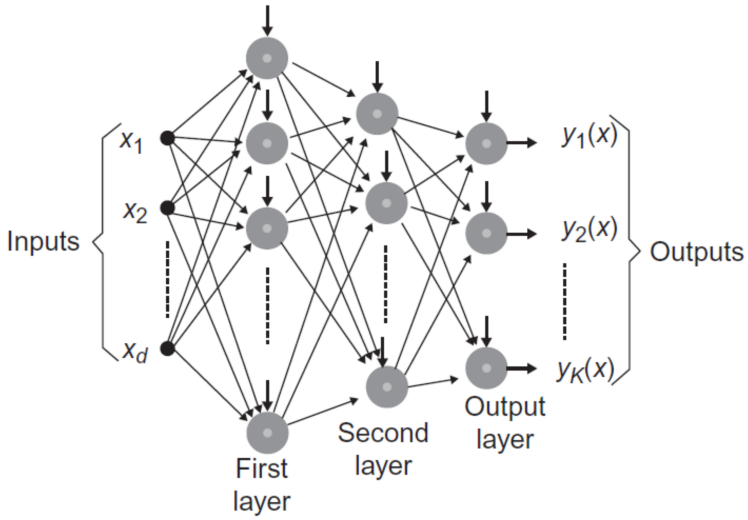


Figure 3.5: MLPNN Architecture.

The Equation 3.5 represents the Multilayer Perceptron (MLP) output.

$$yi\left(x^{(j)}\right) = \varphi\left(\sum_{k=1}^{n} w_{ik}x_k^{(j)} + b_i\right), \tag{3.5}$$

where, $\varphi(x)$ is the non-linear activation function, yi is the output of i-th neuron, $x^{(j)}$ is the

input of j-th layer, $x_k^{(j)}$ is the value of k-th neuron in j-th layer, w_{ik} is the weight between i-th neuron to k-th neuron and b_i is the bias value for i-th neuron.

3.7 Mutual Information matrix

The MI matrix quantifies the non-linear dependence between two random variables [36]. Therefore, given a multivariate time series, an MI matrix can be constructed in a stationary environment by evaluating the MI values between each pair of variables. The MI matrix is formally defined in Equation 3.6. Given a m-dimensional process ξ, denoted by $x_i(i = 1, 2, \ldots, m)$ the i-th dimensional of the process measurement, then the MI matrix over ξ is defined.

$$M = \begin{bmatrix} I(x_1; x_1) & I(x_1; x_2) & \ldots & I(x_1; x_m) \\ I(x_2; x_1) & I(x_2; x_2) & \ldots & I(x_2; x_m) \\ \vdots & \vdots & \ddots & \vdots \\ I(x_m; x_1) & I(x_m; x_2) & \ldots & I(x_m; x_m) \end{bmatrix} \in \mathbb{R}^{m \times m}, \tag{3.6}$$

where $I(x_i; x_j)$ denotes MI between variables x_i and x_j.

According to Shannon information theory, $I(x_i; x_j)$ is defined over the join probability distribution of x_i and x_j and their respective marginal distribution [36]. Equation 3.7 presents the definition of $I(x_i; x_j)$.

$$I(x_i; x_j) = H(x_i) + H(x_j) - H(x_i, x_j), \tag{3.7}$$

where $H(\cdot)$ denote the entropy and $H(\cdot, \cdot)$ denotes the join entropy. In particular, $I(x_i; x_i) = H(x_i)$.

In theory, the MI matrix is symmetric and non-negative. Furthermore, without any dependence on the paired variables, it reduces to a diagonal matrix with the entropy of each variable on the main diagonal.

3.8 Data Balancing

Imbalanced data significantly affects the learning process since most standard machine learning algorithms expect a balanced class distribution or an equal misclassification cost [37]. For this reason, it was necessary to solve the dataset imbalance problem.

Let us consider dt an imbalanced dataset with dt_{min} and dt_{maj} being the subsets of samples belonging to the minority and majority classes, respectively. The balancing ratio of the dataset is defined in Equation 3.8

$$r_{dt} = \frac{|dt_{min}|}{|dt_{maj}|}, \tag{3.8}$$

where $|.|$ represents the cardinality of a set. The process of balancing involves resampling dt into a new dataset dt_{res}, such that $r_{dt} > r_{dt_{res}}$.

3.8.1 Oversampling

With oversampling, the quantity of minority class samples is increased to match that of the majority classes. The dataset size is increased. Oversampling can achieve balance in the data by generating new samples in dt_{min} until the desired balancing ratio $r_{dt_{res}}$ is reached. The process of oversampling is depicted in Figure 3.6.

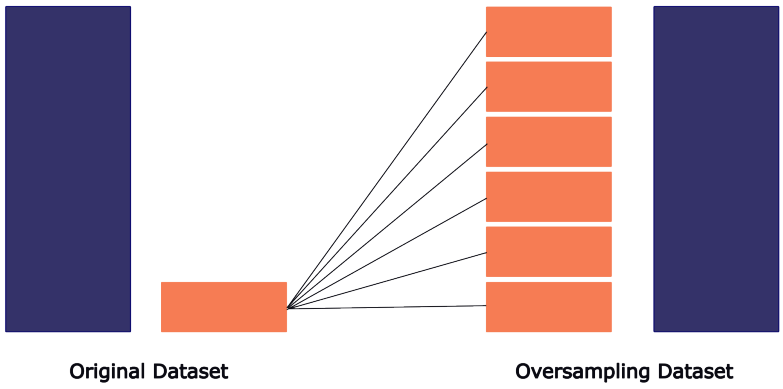


Figure 3.6: Oversampling process.

The Synthetic Minority Oversampling Technique (SMOTE) is essential for oversampling the minority class to generate balanced datasets [38]. It achieves this by oversampling each minority class sample and including synthetic examples along the line segments joining any or all of the k minority class nearest neighbors. The number of neighbors from the k-nearest neighbors is randomly selected based on the amount of oversampling required.

3.8.2 Undersampling

Undersampling reduces the number of samples in the majority class dt_{maj}. The quantity of majority class samples dt_{maj} is reduced to match that of the minority classes, resulting in an overall reduction in dataset size. The undersampling technique is illustrated in Figure 3.7.

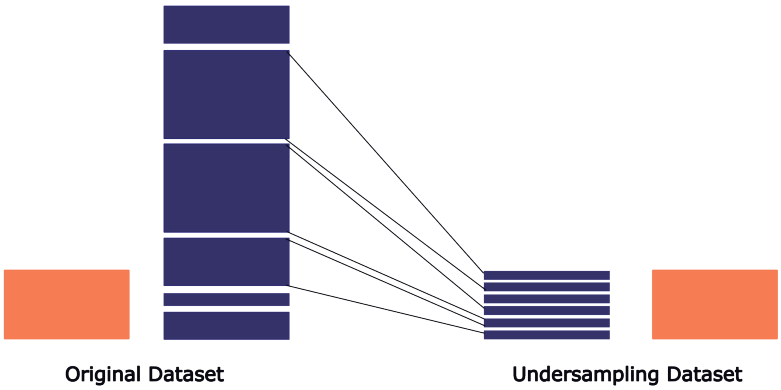


Figure 3.7: Undersampling process.

The Nearmiss method is one of the most commonly used techniques for performing dataset undersampling [39]. Its objective is to choose a sample from the majority class nearest to multiple samples from the minority class. The selection criterion for samples from the majority class is the one with the smallest average distance to the three nearest samples from the minority class.

3.9 Evaluation Metrics

One way to evaluate machine learning models is through measurements. These measurements, commonly called *evaluation metrics*, allowed us to measure certain aspects, trends, or results. Furthermore, additional techniques have been utilized to support the acquisition of evaluation metrics for an ML model.

3.9.1 Confusion Matrix

The Confusion Matrix (CM) is used to observe the estimates of the classification possibilities of the respective True (T) and False (F) values and the Positive (P) and Negative (N) predicted classes [40]. True Positives (TP) occur when a positive result is predicted and the value is also positive. False Positives (FP) occur when a positive result is predicted, but the value is negative. False Negatives (FN) occur when a negative result is predicted, but the value is positive. True Negatives (TN) are obtained when a negative result is predicted and the value is negative. The values estimated by the CM [40] are detailed in Figure 3.8.

3.9.2 Classification Measurements

For classification problems, the most common metrics are Prediction Accuracy Rate (PAR), True Positive Rate (TPR) - Sensitivity, True Negative Rate (TNR) - Specificity, F1 Score, and AUC [40].

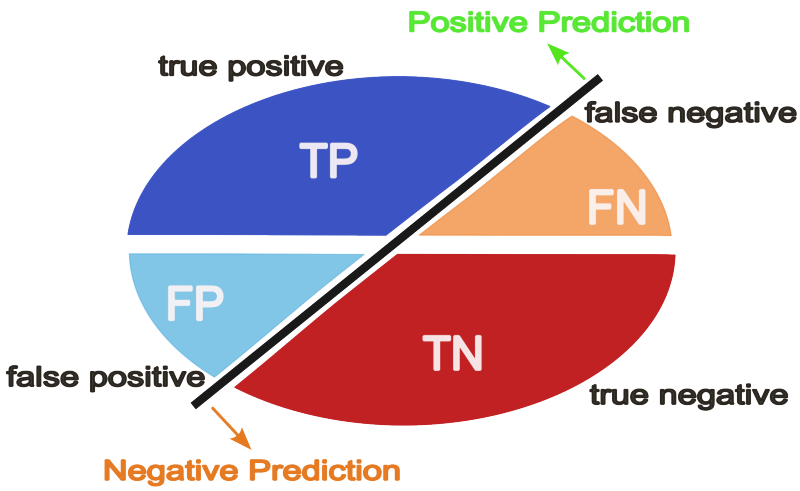


Figure 3.8: Confusion Matrix values.

The metrics described in [8] were used to evaluate the effectiveness of the models developed in this work. The Specificity (SPE) is defined in Equation 3.9. SPE is calculated by dividing the number of correct negative predictions TN by the total number of negatives F. The Sensitivity (SEN) is defined in Equation 3.10. SEN is calculated by dividing the number of accurate positive predictions TP by the total number of positives T. The Accuracy (ACC) is defined in Equation 3.11. ACC is calculated by dividing the sum of two accurate predictions, TP + TN, by the total number of data P + N. The elapsed time (Et) in seconds is used to calculate the training time and validate the models.

$$SPE = \frac{TN}{TN + FP} = \frac{TN}{N}, \tag{3.9}$$

$$SEN = \frac{TP}{TP + FN} = \frac{TP}{P}, \tag{3.10}$$

$$ACC = \frac{TP + TN}{TN + TP + FN + FP} = \frac{TP + TN}{P + N}. \tag{3.11}$$

3.9.3 Cross-Validation

Cross-validation, a type of Monte Carlo method [41], is a data resampling technique used to evaluate the generalization ability of predictive models and prevent *overfitting* to avoid creating a model perfectly adapted to the dataset at hand but unable to generalize well to new, unseen data. On the other hand, a simpler model may be less affected by inherent

noise, but it may not capture the relation between variables well, resulting in *underfitting*. Finding the right balance between overfitting and underfitting is crucial for achieving good generalization performance.

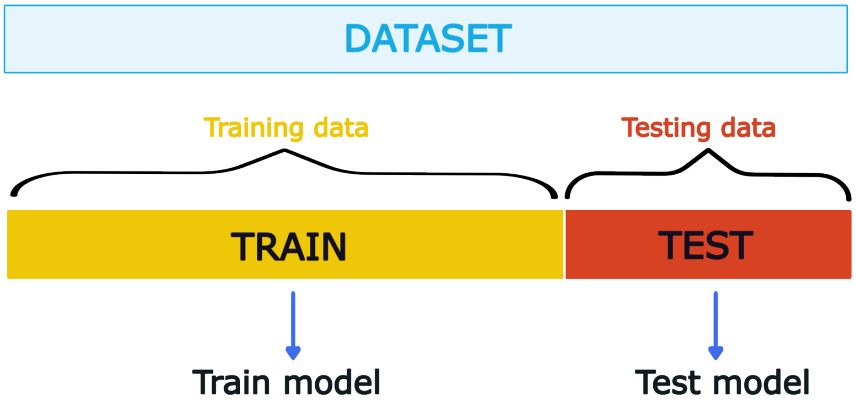


Figure 3.9: Holdout method.

There are several types of cross-validation, of which the most important are the holdout method and K-fold cross-validation [41].

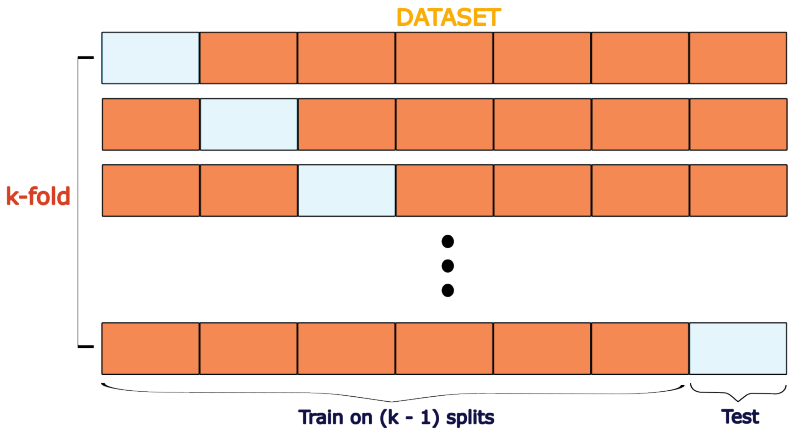


Figure 3.10: K-fold method.

The *Holdout* method is a simple form of cross-validation. It involves dividing the dataset into two sets: the training set and the testing set. The function approximator fits a function using only the training set. Then, the approximator function predicts the output values for the data in the testing set (which it has never seen before). The errors made are accumulated to give the mean absolute test set error, which is used to evaluate the model. The method has the

advantage of taking no longer to compute. However, its evaluation can have a high variance. Figure 3.9 illustrates the Holdout method.

The *K-fold cross-validation* method improves upon the *Holdout* method. The data set is divided into k subsets, and the holdout method is repeated k times. Each time, one k subset represents the test set, and the other k-1 subsets are combined to form a training set. Then, the average error across all k trials is computed. This method has the advantage of being less sensitive to how the data is divided. Each data point is included in the test set once and in the training set k-1 times. Increasing k reduces the variance of the resulting estimate. However, this method has the disadvantage of requiring the training algorithm to be rerun from scratch k times, resulting in k times more computation needed to make an evaluation. This method is shown in Figure 3.10.

Chapter 4

METHODOLOGY

This chapter outlines the research paradigm, methods and techniques, and elements used to design the algorithm models and conduct experiments. It includes materials, computational software, a methodology overview, a dataset description, a model configuration, and hyperparameter setup information.

4.1 Research Paradigm

This research utilized a *postpositivist* paradigm, known for embracing the scientific method and allowing for a deep analysis of the *quantitative approach* to obtain cause-effect results. In this manner, the research's traceability and reproducibility are ensured.

4.2 Research Methods and Techniques

The *experimental* method was chosen for its scientific criteria, which allows for data collection through experimentation. This method enables the determination of causes and effects through the analysis of quantitative variables, leading to a conclusion regarding the acceptance or rejection of the proposed hypothesis. To support this method, *statistical* and *comparative analysis* techniques were used to interpret and analyze the findings.

4.3 Materials

The laptop detailed in Table 4.1 was the primary tool for developing, executing, and evaluating the ANN and classification algorithms.

Table 4.1: Hardware specifications

Hardware	Specifications
Laptop	Intel Core i7-12700H CPU 1TB SSD of storage 16 GB of RAM NVIDIA GeForce RTX 3060 GPU

4.4 Computational Software

The software utilized in this study is outlined in Table 4.2.

Table 4.2: Required software

Type	Software	Version
Laptop OS	Windows	11 Pro x64 bits
IDE	Visual Studio Code	1.87.0
Programming language	Python	3.11.4
Programming language	R	2.8.2
Python Library	Scikit-learn	1.3.0
Machine learning platform	Tensorflow	2.15.0

4.5 Methodology overview

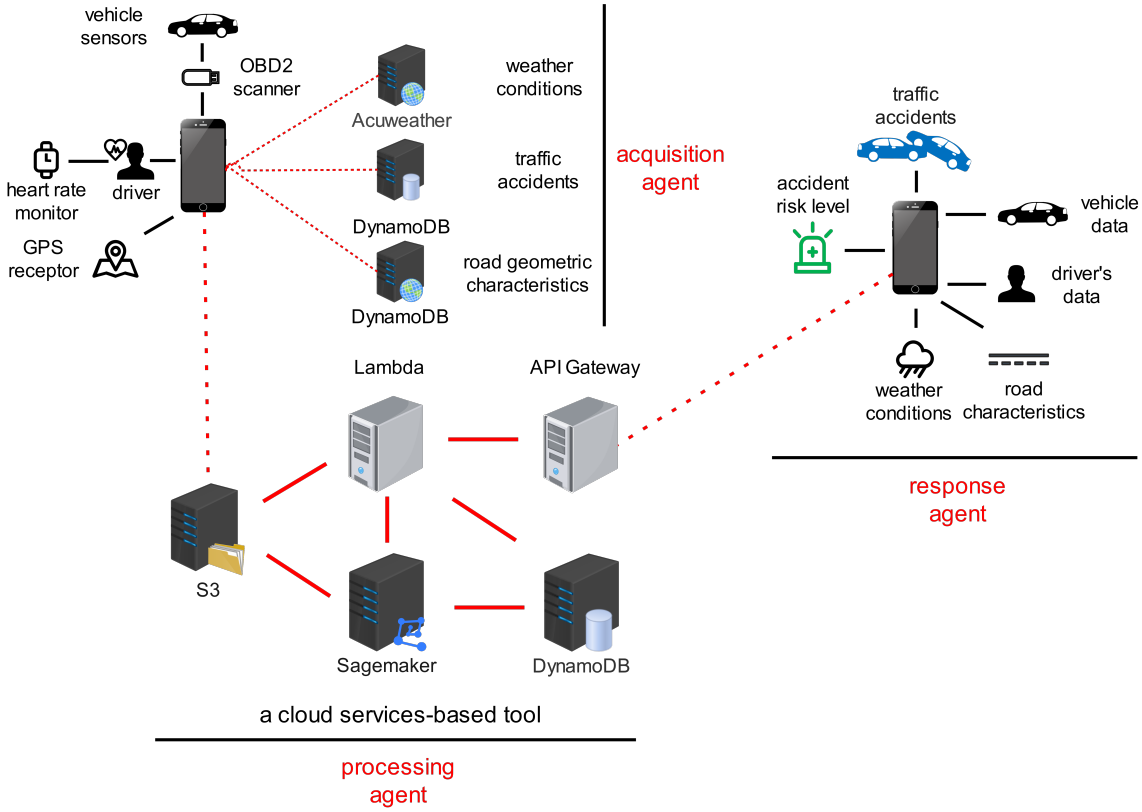


Figure 4.1: Project for predicting traffic accident risk levels.

This investigation is part of a larger project that aims to construct a dataset gathering information on driver's data, vehicle information, environment conditions, and traffic accidents in various locations throughout Quito city and its surroundings. By utilizing DL and ML algorithms, the researchers hope to obtain models to assess the risk level of traffic accidents and integrate them into a secure alert system that allows drivers to receive notifications about

their current situation via their mobile phones.

The project comprises three phases or agents: acquisition/storage, processing, and presentation. The acquisition/storage agent comprises a mobile application and an OBD2 scanner. Its purpose is to collect information about weather conditions, traffic accident data, and a driver's vital sign information and store all this data in a repository.

The processing agent consists of software tools that enable the reading of available driving data, processing it in an ML model, and reorganizing the resulting data in a repository. The response or presentation agent is a mobile application that enables querying of data available from a repository and presenting this information to end users, who represent the drivers that will use this application. Figure 4.1 displays the complete context of the project mentioned above.

This work aims to contribute to the processing phase by developing and evaluating models that can be used to predict accident levels. Thus, similar to Kumeda [17], the primary objective of this study was to propose creating models that treated this classification problem for predicting accident risk. These models helped predict accident risk levels by analyzing a driving incident dataset based on various factors, including driver's data, weather conditions, deaths in accidents at the location, and car characteristics. The methodology used to implement the different classification models of this study is described in Figure 4.2.

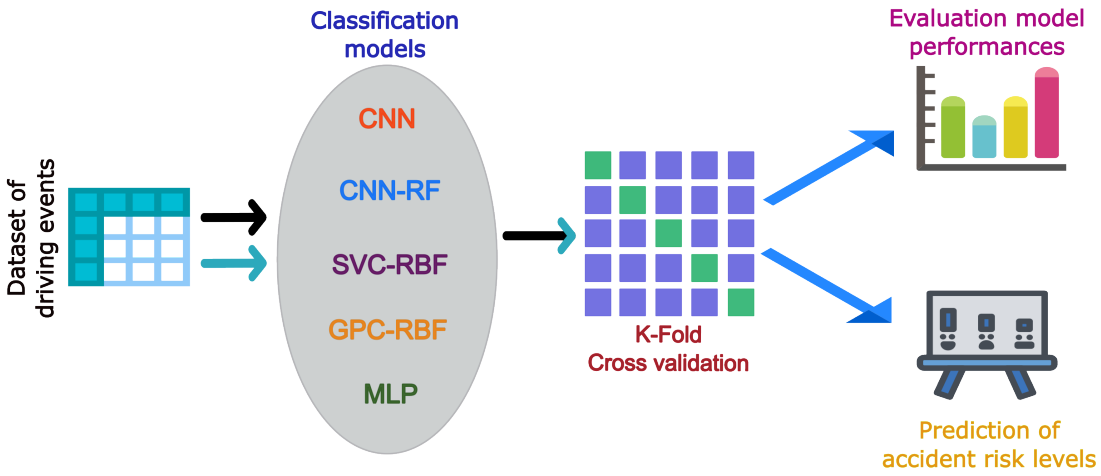


Figure 4.2: Overview of the proposed methodology for implementing models.

The study aims to show how feasible it is to apply a dynamic dataset approach that allows the incorporation of new data to improve risk level prediction and generate more accurate models for handling driver warnings. The task is to find an algorithm configuration that reduces the time required for validation and processing when working with this dataset type.

4.6 Dataset description

The *PoliDriving* dataset was used in this study, which was generated after an earlier investigation by some team members of artificial intelligence at the Escuela Politécnica Nacional [42]. It comprises 2634 samples with 23 numerical features and one predictive class, its fields corresponding to driver information, vehicle data, weather conditions, and traffic accidents.

The data was acquired, processed, and updated to provide a dynamic dataset that supported the development, training, and testing of all obtained models. Table 4.3 provides a general description of the *PoliDriving* dataset.

Table 4.3: Dataset description

No.	Feature	Description	Unit Type
1	steering_angle	Vehicle steering wheel plane angle respect to the road surface in sexagesimal degrees.	grades
2	speed	Vehicle speed in meters per second.	m/s
3	rpm	Vehicle engine speed in revolutions per minute.	rpm
4	acceleration	Vehicle acceleration in meters per second squared.	m/s^2
5	throttle_position	Sensor used to monitor the air intake of the engine.	%
6	engine_temperature	Temperature of the air entering the engine.	°C
7	system_voltage	Voltage of the vehicle's electrical system.	V
8	barometric_pressure	Pressure variable to change based on weather conditions.	kPa
9	distance_travelled	Distance traveled by the vehicle in one-time unit.	km
10	latitude	Latitude coordinates of the geographic position.	λ degrees
11	longitude	Longitude coordinates of the geographic position.	ω degrees
12	heart_rate	Number of contractions of the heart per minute.	bpm
13	current_weather	Category for daily weather forecast in a specific location.	numerical
14	temperature	Forecasted temperature for a specific location.	°C
15	real_feel_temperature	Temperature for a specific location.	°C
16	wind_speed	Relationship of the distance traveled by air concerning the time spent traveling it.	m/s
17	relative_humidity	Measurement of water vapor content in the air.	%
18	uv_index	Ultraviolet solar radiation intensity that incident on the Earth's surface.	UV
19	cloud_cover	Percentage of the sky covered by clouds in a specific location, measured in octas.	octa
20	ceiling	Height of the lower level cloud layer or broken clouds.	m

Table 4.3: Dataset description (cont.)

No.	Feature	Description	Unit Type
21	pressure	Force per unit area exerted by the atmosphere at a specific point.	inch
22	precipitation	Water that falls from the atmosphere to the Earth's surface.	l/m^2
23	accidents_onsite	Number of deaths occurred in accidents at the location.	deaths

4.6.1 Feature analysis

Reducing the dimension of a dataset is a critical technique in data analysis. This task is complex because it involves reducing the number of related variables while retaining the most important information. This reduction is necessary to improve computational efficiency and prevent overfitting.

During the dataset analysis, feature selection was necessary to obtain an optimal set of features. The use of Principal Component Analysis (PCA) and linear discriminant analysis (LDA) was not considered appropriate since these techniques are particularly effective only when the data have a linear relationship, meaning that each characteristic of the dataset can be expressed in terms of the others. The Mutual Information matrix (MI) was used to observe the dependence between non-linear variables.

To generate the correlation graph that simulates the objective of the MI matrix, we utilized the *cor* package developed in the R programming language. This package is described in Appendix A.1, and the corresponding code is displayed in Appendix A.2.

The scores of each feature were calculated using the *mutual_info_classif* function from the Python scikit-learn library, representing the degree of dependence between each variable and the target class. The *mutual_info_classif* function is described in Appendix A.3.

4.6.2 Undersampling dataset

The dataset presents the problem of imbalanced data concerning the predictor class. Imbalanced data significantly affects the learning process.

For preprocessing the data, the undersampling technique was used to reduce the number of samples to the minority class to generate a balanced dataset. It was implemented utilizing *imbalanced-learn* [37], an open-source Python toolkit that offers a broad array of methods for dealing with the common issue of imbalanced datasets in pattern recognition and ML.

Thus, the *Nearmiss* method version 1 was employed to balance the data [39]. Appendix A.4 describes the *imbalanced-learn* toolkit, while Appendix A.5 explains the *Nearmiss* method.

4.7 Model selection and configuration

Four approaches are analyzed to build models and evaluate their performance in classifying accident risk levels. Based on the evidence presented in the related works analyzed in this study, CNN, RF, and MLP networks were chosen since they are commonly used for their high efficiency. The results obtained by these networks showed that they have a high accuracy rate and good prediction estimates.

The objective is to compare these results against the RBF networks to test and confirm their quality in terms of speed and accuracy. The crucial point of comparison will be the time these algorithms require to validate and process the acquisition of new models using a dynamic dataset.

4.8 Hyperparameters

The models tested include CNN and an RF classifier. Subsequently, two variants of the RBF algorithm were examined, followed by the MLP classifier. The *GridSearchCV* class was used from the Python scikit-learn library to adjust the best hyperparameters. The details of this class can be found in Appendix A.6.

4.8.1 CNN model

The CNN model was implemented using *TensorFlow* with a 1D input layer and four 1D convolutional layers. The model also included a fully connected layer and a 1D output layer. The ReLU activation function was used for the input, convolutional, and fully connected layers, while the output layer employed the Softmax activation function. Appendix A.7 describes the TensorFlow platform.

Additionally, all convolutional layers have a Maxpooling1D with a kernel_size. Moreover, Dropout was finally applied to the fully connected layer. Figure 4.3 shows the configuration of the CNN model.

4.8.2 CNN-RF model

The CNN-RF model was created using the *RandomForestClassifier* class from the Python scikit-learn library. Appendix A.8 provides additional information on the *RandomForestClassifier* class. The input of the CNN-RF model is an intermediate layer (conv1D) of the

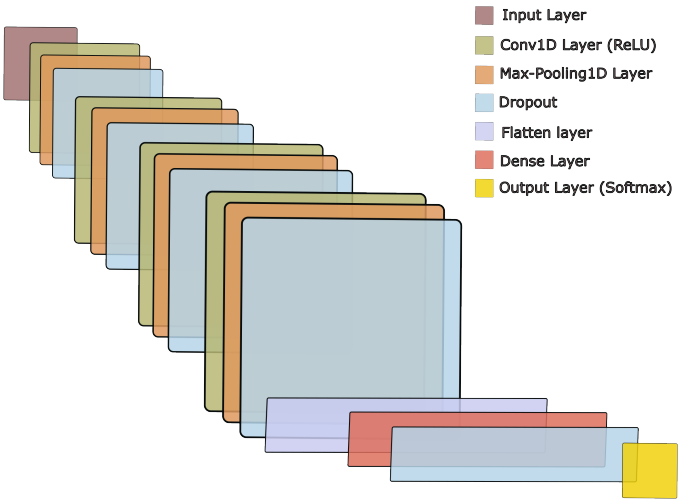


Figure 4.3: CNN configuration.

CNN model. Figure 4.4 shows the configuration of the CNN-RF model.

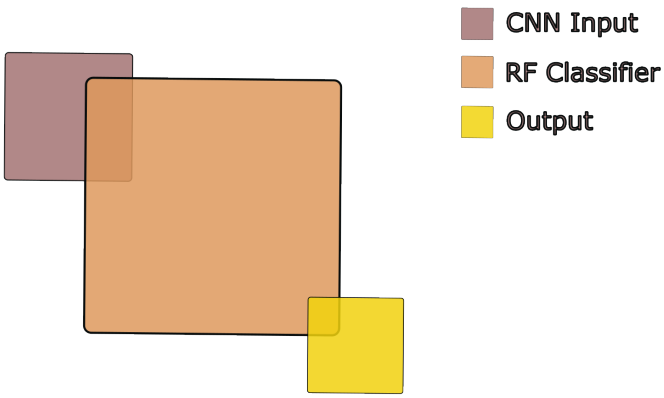


Figure 4.4: CNN-RF configuration.

4.8.3 RBF models

The Gaussian function is the main base function for RBF, and it was implemented for the two analyzed approaches. The first algorithm was implemented through the *GaussianProcessClassifier* class and the RBF kernel from the Python scikit-learn library. The *GaussianProcessClassifier* class is detailed in Appendix A.9.

The second RBF approach used the C-Support Vector Classification (*SVC*) class from the same Python scikit-learn library. The class *SVC* is described in Appendix A.10. The configurations of these models are depicted in Figure 4.5.

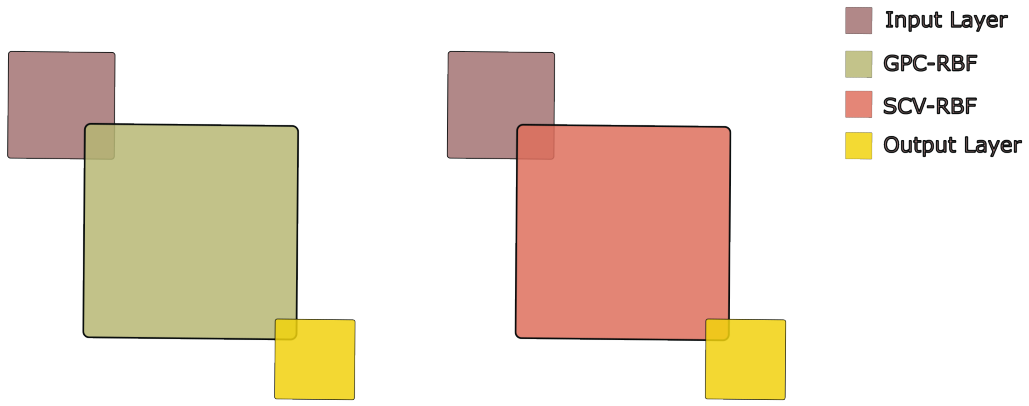


Figure 4.5: RBF configurations.

4.8.4 MLP model

The *MLPClassifier* class from the Python scikit-learn library was used to implement the MLP classifier model. The *MLPClassifier* class is described in Appendix A.11. The configuration used is shown in Figure 4.6.

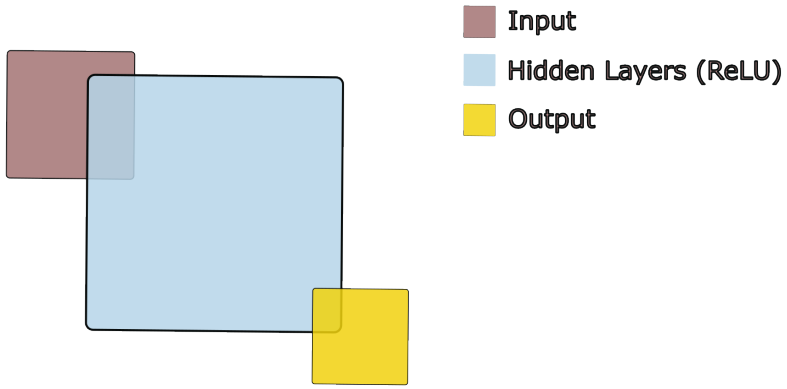


Figure 4.6: MLP configuration.

4.9 Implementation and evaluation models

These models were developed to assess their efficiency and performance in predicting levels of risk for traffic accidents. All models were tested and validated using the *POLIDriving* dataset. They were implemented in Python using the *Visual Studio Code* IDE mentioned in Appendix A.12. The hyperparameters obtained for each model were utilized, and a cross-validation approach with a fold of 5 was employed to evaluate each model.

The CM provided values to compute for three main metrics used in this study: ACC, SPE, and SEN. These metrics were chosen for specific reasons. ACC validates the correctness of

the predictions made by the developed models. SPE helps determine whether risk levels are correctly excluded from non-risk events. In other words, it allows distinguishing between events that appear risky but are not. Finally, using SEN allows for determining whether an event is risky and enables assessing whether a driver is driving safely or engaging in risky driving behavior.

The list below outlines the activities conducted to carry out each experiment:

1. Dataset
    (a) The MI matrix should be applied to the dataset to generate the corresponding correlation graph.
    (b) Calculate the scores of each dataset feature using the MI matrix.
    (c) Select the features with the highest scores and determine the number of features encompassing the largest number of factors while avoiding the largest number of related features.
    (d) Finally, perform model validation using the resulting dataset.
2. Models
    (a) Once the dataset has been preprocessed, apply the undersampling method.
    (b) The number of predictor classes is determined, their values are coded, and all data is normalized.
    (c) The hyperparameters are then calculated, and the corresponding values are used to create the architecture of each model.
    (d) Each model is implemented in Python using the calculated hyperparameters.
    (e) Cross-validation is performed using 5-fold, and in each fold, the CM values and the corresponding evaluation metrics for each model (ACC, SPE, SEN) are obtained.
    (f) Finally, the average scores for each evaluation metric computed in each fold and for each model are obtained.
    (g) In the case of the CNN model, it is necessary to save the generated model.

Chapter 5

RESULTS

This chapter presents the study results, including feature analysis and selection, dataset undersampling results, obtained hyperparameters, and a comparison of model performance.

5.1 Feature analysis

The MI matrix was used to identify correlations between attributes, allowing for the recognition of the main attributes in the dataset and their relationships with the target class. Figure 5.1 shows the correlation between the dataset's features.

5.2 Feature selection

We considered the 12 most important features. Figure 5.2 describes the obtained score values of each feature.

The significance of these characteristics lies in their ability to encompass a wide range of factors that impact accident risk prediction and their correlation with the target class. Table 5.1 describes the selected features with the scores respectively.

Table 5.1: Selected features with the highest score

Position	Feature	Score
1	distance_travelled	1.050792
2	latitude	0.860045
3	speed	0.733158
4	longitude	0.701887
5	rpm	0.447969
6	throttle_position	0.288088
7	steering_angle	0.236202
8	system_voltage	0.201812
9	accidents_onsite	0.194004
11	heart_rate	0.112412
10	engine_temperature	0.080994
12	barometric_pressure	0.077778

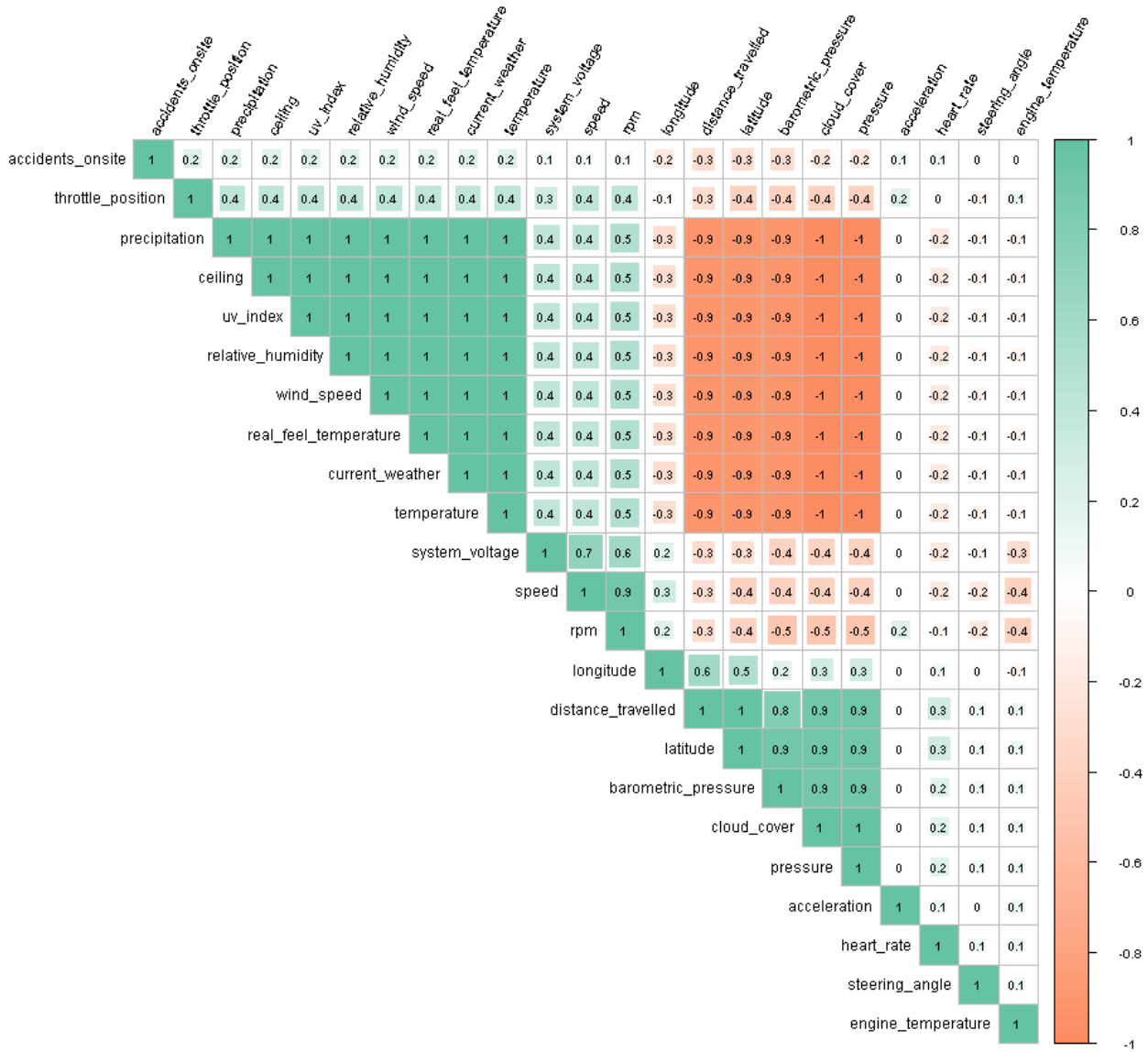


Figure 5.1: Mutual Information matrix.

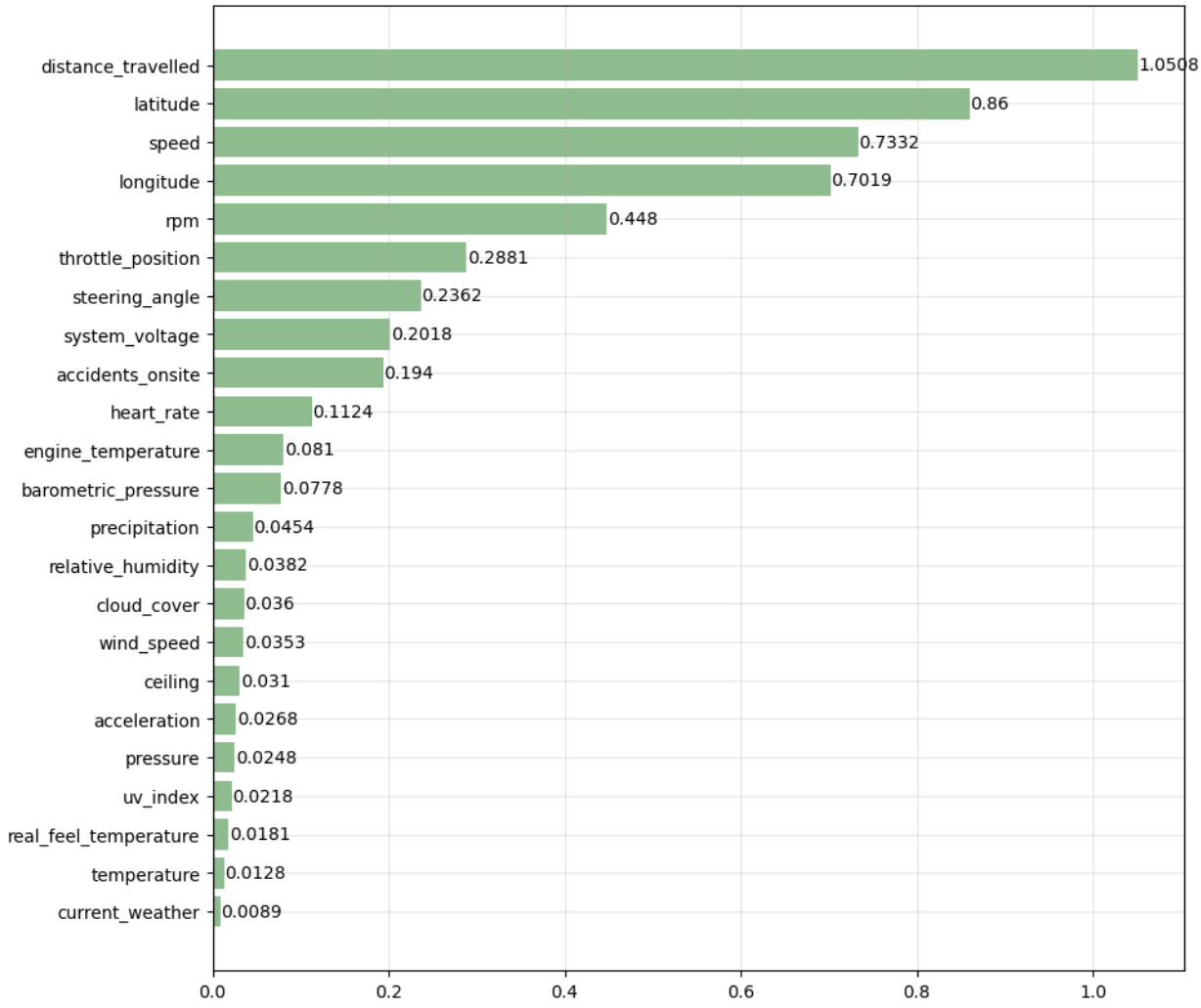


Figure 5.2: Mutual information scores.

5.3 Dataset undersampling

Undersampling was necessary to balance the dataset. Figure 5.3 shows the number of samples, 480, 1155, 496, and 503, for each class in the original imbalance dataset.

The *Nearmiss* method version 1 mentioned in Appendix A.5 was used to balance the dataset. This method transformed the unbalanced data into four balanced classes, each with 480 samples. These four predictor classes represent different risk levels: low, medium, high, and very high.

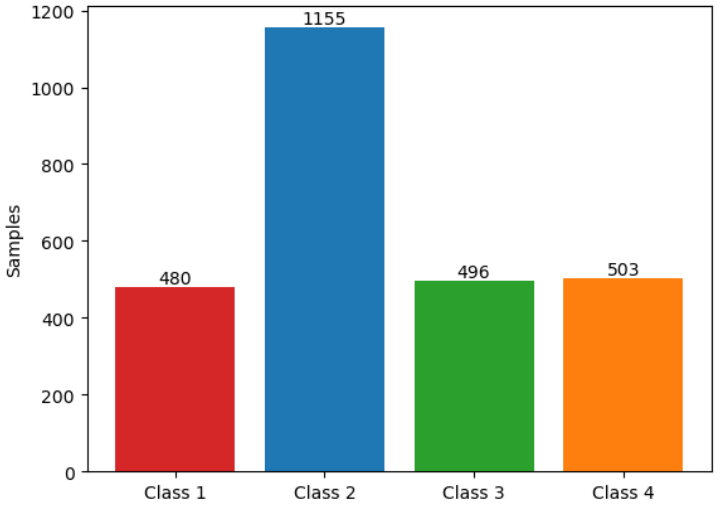


Figure 5.3: Imbalance dataset.

The *POLIDriving* dataset consisted of 2634 samples. However, to ensure balanced categories of the predictor class through undersampling, the dataset was reduced to 1920 samples.

5.4 Hyperparameters tuning

The hyperparameters obtained for each model are presented in this section.

5.4.1 CNN

The architecture and the hyperparameters used for the CNN were a 1D input layer of 32 neurons and four 1D convolutional layers with 128, 64, 128, and 256 neurons, respectively. A fully connected layer with 512 neurons and a 1D output layer. All convolutional layers have a Maxpooling1D of 1 with a kernel_size of 3. A Dropout of 0.5 was applied to the fully connected layer. The hyperparameters included the Adam optimizer with a learning_rate of 0.001, a beta of 0.9, and a momentum of 0.99. The testing phase consisted of 100 epochs with a batch_size of 32. Table 5.2 presents the hyperparameters calculated for the CNN network.

Table 5.2: CNN hyperparameters

Hyperparameters	Values
1D input layer	32, ReLu
1D convolutional layer 1	128, ReLu
1D convolutional layer 2	64, ReLu
1D convolutional layer 3	128, ReLu
1D convolutional layer 4	256, ReLu
1D fully connected layer	512, ReLu
1D output layer	4, Softmax
maxpooling1D	1
dropout	0.5
kernel_size	3
optimizer	adam
learning_rate	0.001
beta_1	0.9
beta_2	0.999
epsilon	1e-07
ema_momentum	0.99
epochs	100
batch_size	32

5.4.2 CNN-RF

Like input, the CNN-RF model takes an intermediate layer of the CNN network, with an output shape of (None, 2, 256) and 98560 params. CNN-RF is a mixed model because it utilizes a DL network and a classification algorithm. The hyperparameters considered were a max_depth and n_estimators. The hyperparameters of the CNN-RF model are shown in Table 5.3.

Table 5.3: CNN-RF hyperparameters

Hyperparameters	Values
Input layer	(None, 2, 256)
max_depth	15
n_estimators	50

5.4.3 RBF models

For the GPC-RBF model, the main hyperparameter is a composite kernel with a constant scale (1**2) multiplied by the base kernel RBF with a length scale of 1. The SVC-RBF model has a kernel RBF and regularization hyperparameters C of 7000 and gamma of 0.01. Table 5.4 shows the hyperparameters mentioned above.

Table 5.4: RBF hyperparameters

Model	Hyperparameters	Values
GPC-RBF	kernel_type	RBF
	kernel	1**2
	max_iter_predict	20
SVC-RBF	kernel	RBF
	C	7000
	gamma	0.01

5.4.4 MLP

The hyperparameters utilized by the MLP model were the ReLU activation, alpha, hidden_layer_sizes, learning_rate, max_iter, and solver. The hyperparameters of this model are shown in Table 5.5.

Table 5.5: MLP hyperparameters

Hyperparameters	Values
activation	ReLU
alpha	0.0001
hidden_layer_sizes	(100,50,30)
learning_rate	adaptative
max_iter	1500
solver	adam

5.5 Model evaluation results

The section below outlines the evaluation metrics for each of the implemented models. It is important to note that these results were obtained using 5-fold cross-validation.

5.5.1 CNN

The evaluation results of the CNN model are presented in Table 5.6. The results calculated for each fold are detailed in Appendix B.1.

Table 5.6: CNN model results

Fold	Accuracy	Specificity	Sensitivity
1	0.9270	0.9748	0.9244
2	0.8958	0.9650	0.8913
3	0.9479	0.9826	0.9478
4	0.9687	0.9895	0.9686
5	0.9661	0.9886	0.9672

The evaluation of the model lasted 694.2 seconds.

5.5.2 CNN-RF

Table 5.7 presents the results of evaluating the CNN-RF model. Appendix B.2 presents the evaluation metrics results obtained in each fold.

Table 5.7: CNN-RF model results

Fold	Accuracy	Specificity	Sensitivity
1	0.9661	0.9884	0.9658
2	0.9479	0.9824	0.9484
3	0.9687	0.9895	0.9681
4	0.9609	0.9871	0.9619
5	0.9583	0.9861	0.9573

The evaluation of the model took approximately 695.9 seconds.

5.5.3 GPC-RBF

The evaluation results for the GPC-RBF model are presented in Table 5.8. The outcomes of each fold of this model are displayed in Appendix B.3.

Table 5.8: GPC-RBF model results

Fold	Accuracy	Specificity	Sensitivity
1	0.8958	0.9651	0.8966
2	0.9036	0.9680	0.9041
3	0.9088	0.9696	0.9097
4	0.9114	0.9705	0.9144
5	0.8880	0.9628	0.8881

The evaluation of the model took 302.3 seconds.

5.5.4 SVC-RBF

The results of the SVC-RBF model evaluation are shown in Table 5.9. Appendix B.4 presents the results generated by this model in each fold and its respective CM.

The model was evaluated in an elapsed time of 1.7 seconds.

Table 5.9: SVC-RBF model results

Fold	Accuracy	Specificity	Sensitivity
1	0.9114	0.9704	0.9134
2	0.9140	0.9715	0.9130
3	0.9036	0.9681	0.9027
4	0.9348	0.9783	0.9342
5	0.9062	0.9685	0.9059

5.5.5 MLP

The evaluation results of the MLP model are presented in Table 5.10. Appendix B.5 presents the evaluation metrics results and the respective CM for each fold.

Table 5.10: MLP model results

Fold	Accuracy	Specificity	Sensitivity
1	0.9244	0.9744	0.9252
2	0.9088	0.9699	0.9099
3	0.9296	0.9769	0.9298
4	0.9192	0.9734	0.9177
5	0.8958	0.9654	0.8983

The model was evaluated using the cross-validation method, which took 10.7 seconds.

5.6 Comparing model results

After conducting the experiments and evaluating the presented models, this section compares and analyzes the obtained evaluation metric values.

5.6.1 Comparing evaluation metric scores

Table 5.11 displays the evaluation metrics and the elapsed time obtained for each evaluated model.

The results show that the **CNN-RF** model achieved the highest accuracy (0.9604) and better classification capability, but it took longer to execute and evaluate (695.9 seconds) than the other models. The **CNN** model achieved the second-best accuracy (0.9411) and had a similar execution time to **CNN-RF** with 694.2 seconds. The **MLP** model achieved an accuracy of 0.9156 and had a short execution time of 10.7 seconds. The **SVC-RBF** model had the best run time, taking only 1.7 seconds. Its accuracy was also good, with a score of 0.9140, similar to the **MLP** model's accuracy. Finally, the **GPC-RBF** model achieved a comparable accuracy score of 0.9016 and an execution time of 302.3 seconds.

Table 5.11: Comparison of the model results

Model	ACC	SPE	SEN	Time (s)
CNN	0.9411	0.9801	0.9398	694.2
CNN-RF	**0.9604**	**0.9867**	**0.9603**	695.9
GPC-RBF	0.9016	0.9672	0.9026	302.3
SVC-RBF	0.9140	0.9713	0.9139	**1.7**
MLP	0.9156	0.9720	0.9162	10.7

The ACC values for the models are displayed in Figure 5.4.

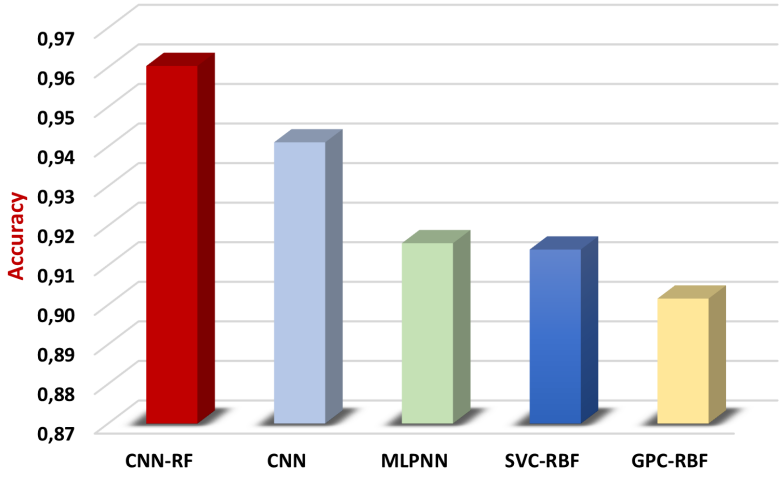


Figure 5.4: Accuracy model scores.

It is important to note that while the accuracies of the rest of the models are comparable to the best model accuracy **CNN-RF** with 0.9604, the **SVC-RBF** model in 1.7 seconds of evaluation achieved a significant accuracy (0.9140) obtaining similar good results to **MLP** and **CNN** models. Finally, the less-performing but not worst was the **GPC-RBF** model, which achieved less accuracy with 0.9016.

Based on the analysis of SPE values, it is evident that the positional order of the models remains unchanged. Consequently, the trend of SPE follows that of ACC. The SPE values for the models are displayed in Figure 5.5.

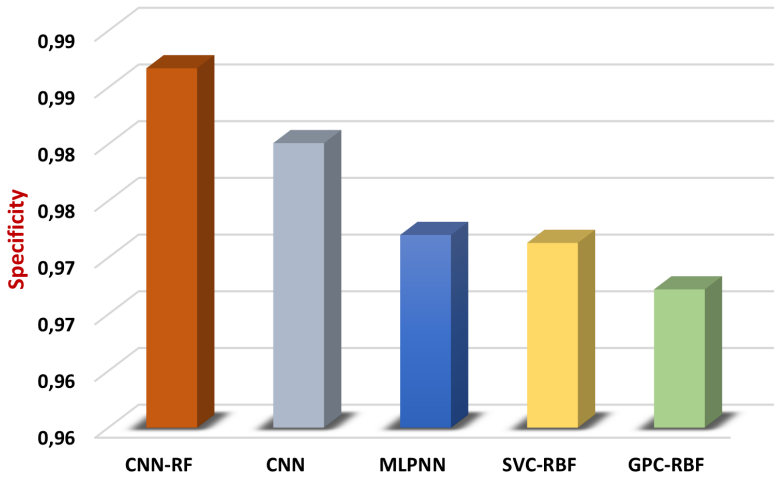


Figure 5.5: Specificity model scores.

The performance of the models about the previous evaluation metrics is consistent with that of the SEN. The SEN values for the models are displayed in Figure 5.6.

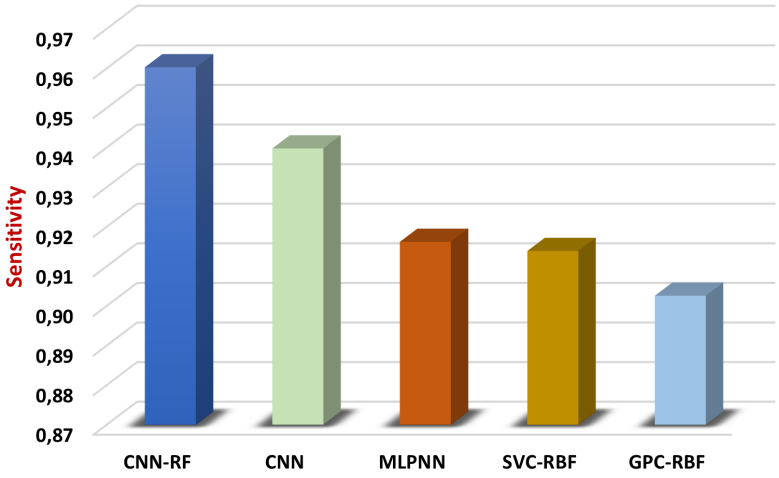


Figure 5.6: Sensitivity model scores.

Upon analyzing these results, it is evident that two DL models, **CNN-RF** and **CNN**, stand out in ACC, SPE, and SEN.

5.6.2 Comparing execution time

Based on the execution time, it is evident that the DL models take much more time to be evaluated than the rest of the implemented models. The evaluation times of each model are presented in Figure 5.7.

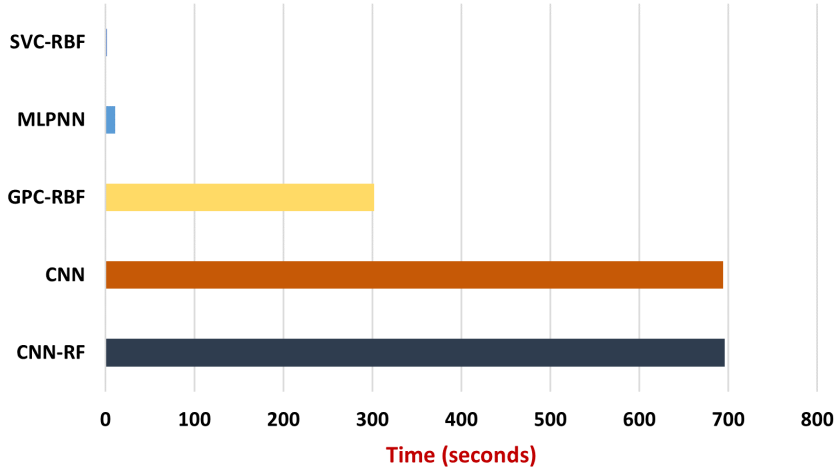


Figure 5.7: Model evaluation times.

The scores obtained by the **SVC-RBF** model were compared to those of the **CNN-RF** model. The two models obtained comparable values for SPE (0.9713 - 0.9867) and SEN (0.9139 - 0.9603), but the training times were significantly different: 694.2 seconds for the **CNN-RF** model versus 1.7 seconds for the **SVC-RBF** model. The **MLP** model has an execution time of 10.7 seconds, a better time than the DL models, and is only a little slower than the **SVC-RBF** model, which is the fastest.

5.7 Discussion

Data is the fundamental resource for any algorithm or model. The results and conclusions can be reached depending on the dataset quality and the target. Therefore, the first issue analyzed in this study was the dataset. Amorim et al. [30] state that SL techniques in ML have demonstrated good results when the dataset's most successful characteristics or attributes are chosen.

The MI matrix was utilized to identify these attributes, which enabled the recognition of the main attributes of the dataset and their relationships. It is noteworthy that when some multiple related attributes or attributes pertain to a common area, for example, the attributes related to climatic conditions, selecting the most representative attribute is sufficient to avoid the need to select the remaining related attributes. This approach also helps to reduce redundancy. Sometimes, selecting a larger number of related features may not improve the accuracy of algorithms and may even result in no advantage. Amorim et al. [30] also note that the dataset must be balanced for an ML algorithm to be effective. This affirmation is especially important in SL, where an imbalance in the predictor class can cause the algorithm to favor predicting the classes with the largest number of samples while performing poorly on

classes with fewer samples.

This study validated that prediction accuracy is poor when using an unbalanced dataset. This problem is further complicated when dealing with a dynamic dataset that constantly adds new information. However, techniques like *undersampling* can be used to maintain balance in the number of samples for each predictor class category. The study analyzed a dynamic dataset approach by updating only the values of a few more driving event tuples without changing the total number of samples. The purpose was to observe if there were relevant changes in the results and performance metrics of the models.

Nevertheless, there was no evidence to confirm that this process affects the training process; for example, when adding new data and the correct balance of the predictor class is maintained, it could not be proven to impact negatively the performance or accuracy of the models significantly. It may not be possible to obtain conclusive evidence since we did not work with a larger amount of data. However, it is evident that when the dataset grows, the training times increase and the prediction accuracy varies. The issue of the dynamic dataset could be analyzed in greater depth in future work.

On the other hand, focusing on the analyzed DL, RBF, and ML models, referring firstly to what Tian and Zhang mentioned [11], is about DL approaches; the most used DL networks are CNN and RNN. This statement can be explained by the fact that this approach obtains robust models and develops a good generalization of a particular problem. In this study, we were able to provide evidence, particularly with the CNN-RF, where its prediction accuracy is the best compared to the rest of the analyzed models; the applied cross-validation technique indicated that the DL models, CNN, and CNN-RF obtained the best accuracy results (0.9411 and 0.9604) respectively. Hence, Ye et al. [9] also mention that many researchers in recent times are using DL networks to create models for risk assessment and prediction.

Another aspect that is also important to mention about DL is the fact that when using the CNN in combination with a classification algorithm like RF, the prediction accuracy increased; so, for example, with the CNN model, a prediction accuracy of 0.9411 was obtained, while when this same ANN was combined with the RF, this new model obtained a prediction accuracy of 0.9604. From this fact, we can also affirm that a CNN could obtain better results when working with another algorithm, at least if it is a classification algorithm.

However, the critical aspect discussed of these DL approaches is the time consumed to evaluate and train models. For example, in this research, we observed that the CNN-RF needs approximately 695.9 seconds to validate a prediction model by classification using a relatively small dataset (1920 samples), compared to the SVC-RBF, the best time execution model obtained, whose run time is only 1.7 seconds. With this evidence, an RBF approach allows faster

training even though these models do not always achieve good generalization and robustness in solving specific problems that could be scaled in magnitude and complexity. However, the scores obtained by the SVC-RBF model were compared to those of the CNN-RF model. The two models obtained comparable values for SPE (0.9713 - 0.9867) and SEN (0.9139 - 0.9603), but the training times were significantly different: 694.2 seconds for the CNN-RF model versus 1.7 seconds for the SVC-RBF model.

Finally, it should be noted that the MLP model also obtained good efficiency performances, with an accuracy of 0.9156. Its execution time is quite fast, taking only 10.7 seconds for its evaluation. These experiments suggest that the SVC-RBF model can be used to evaluate and predict traffic accident risk levels quickly and effectively.

Chapter 6

CONCLUSIONS

Traffic accidents represent a significant threat to human life and a daily danger. Therefore, new tools and technologies are needed to enhance a driver's prediction and risk measurement capabilities while using transportation. Developing IA models has enabled us to determine accident risk values previously unknown in our environment. These data allow us to identify the riskiest points or those with a higher risk level in each stretch of road. The implemented configurations with DL and ML approaches were tested with a dataset of driving events of traffic accident levels. This process allowed the generation of prediction models for these risk levels. Comparing and evaluating these approaches showed that RBF models were faster in evaluating prediction than DL models. This study concluded that RBFNN models are simpler in configuration and have fewer hyper-parameters to consider, contrary to DL network configurations.

Furthermore, RBF allows faster training and comparable efficiency. This fact is advantageous because when using a dynamic dataset that will continuously be updated with new information, RBF allows us to quickly obtain new predictive models, thereby improving the predictive capabilities with new information. The advantage of working with a dynamic dataset is adapting this new information to generate more accurate and useful predictions. Hence, the advantage of having an RBF model is that it allows us to find new prediction models agilely, even in real-time, due to its processing speed.

The DL models showed the best performance results than the other models. It is evident that the predictive ability of DL models stands out, and they approach optimal values for a predictive model. However, the most crucial drawback of the DL approach is that the time required to test and train a model is high, and it is estimated that it tends to increase according to the greater amount of information in the dataset used. For example, in this study, a dataset that can be considered small has been used, and it has already been confirmed that the times used by the DL models implemented for evaluation and training are high. The MLP model proved to be a good prediction model; its execution times are very good compared to the DL models and only a little slower when compared to the RBF models.

This work is part of a larger project; therefore, this contribution has been made precisely

to discover the optimal models that can be used for the project's plans. The study's main contribution was finding fast and efficient models for predicting accident risk levels since these models are expected to be implemented on a cloud server. When comparing the CNN, RF, RBF, and MLP models, it was evident that the RBF model performed the best, presenting the best execution time and comparable performance and prediction efficiency.

Finally, this study will enable the development of new RBF-oriented models to achieve more accurate predictions of high-risk events and traffic accidents. It will also help drivers make better decisions to avoid tragic accidents in our city.

The main limitation of this study was that a relatively small dataset was used for the experimentation due to the amount of data collected and the balancing process. However, this is real information on driving events obtained from the Quito city's roads and surroundings, data that cannot be obtained from any other source. This fact represents progress in generating predictive models, allowing us to issue risky driving alerts in our environment.

Future work on this topic would involve fully implementing the dynamic dataset and incorporating new data from time to time to evaluate the behavior and execution times when using RBF models to generate predictive models. Furthermore, other approaches can further improve the prediction of accident risk levels.

Bibliographic references

[1] World Health Organization, *Global status report on road safety 2023.* Available online: https://www.who.int/teams/social-determinants-of-health/safety-and-mobility/global-status-report-on-road-safety-2023 (accessed on 11 March 2024).

[2] Ecuadorian National Transit Agency, *National accident rate viewer.* Available online: https://www.ant.gob.ec/visor-de-siniestralidad-estadisticas/ (accessed on 26 March 2024).

[3] H. Ren, Y. Song, J. Wang, Y. Hu, and J. Lei, "A deep learning approach to the prediction of short-term traffic accident risk", *arXiv preprint arXiv:1710.09543*, Aug. 2017.

[4] Z. Yang, W. Zhang, and J. Feng, "Predicting multiple types of traffic accident severity with explanations: A multi-task deep learning framework", *Safety Science*, vol. 146, p. 105 522, Feb. 2022, ISSN: 09257535. DOI: 10.1016/j.ssci.2021.105522.

[5] P. Trirat and J.-G. Lee, "Df-tar: A deep fusion network for citywide traffic accident risk prediction with dangerous driving behavior", *Proceedings of the Web Conference 2021*, pp. 1146–1156, Apr. 2021. DOI: 10.1145/3442381.3450003.

[6] M. Zheng *et al.*, "Traffic accident's severity prediction: A deep-learning approach-based cnn network", *IEEE Access*, vol. 7, pp. 39 897–39 910, 2019, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2903319.

[7] B. Pradhan and M. I. Sameen, "Review of traffic accident predictions with neural networks", *Laser Scanning Systems in Highway and Safety Assessment, Advances Science, Technology & Innovation*, pp. 97–109, 2020. DOI: 10.1007/978-3-030-10374-3_8.

[8] S. K. Satapathy, S. Dehuri, A. K. Jagadev, and S. Mishra, "Empirical study on the performance of the classifiers in eeg classification", *EEG Brain Signal Classification for Epileptic Seizure Disorder Detection*, vol. Volume 3, pp. 45–65, 2019. DOI: 10.1016/B978-0-12-817426-5.00003-X.

[9] Q. Ye, Y. Li, and B. Niu, "Risk propagation mechanism and prediction model for the highway merging area", *Applied Sciences*, vol. 13, p. 8014, 14 Jul. 2023, ISSN: 2076-3417. DOI: 10.3390/app13148014.

[10] X. Fan *et al.*, "Deep learning for intelligent traffic sensing and prediction: Recent advances and future challenges", *CCF Transactions on Pervasive Computing and Interaction*, vol. 2, pp. 240–260, 4 Dec. 2020, ISSN: 2524-521X. DOI: 10.1007/s42486-020-00039-x.

[11] Z. Tian and S. Zhang, "Deep learning method for traffic accident prediction security", *Soft Computing*, vol. 26, pp. 5363–5375, 11 Jun. 2022, ISSN: 1432-7643. DOI: 10.1007/s00500-022-07096-7.

[12] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey", *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, pp. 2222–2232, 10 Oct. 2017, ISSN: 2162-237X. DOI: 10.1109/TNNLS.2016.2582924.

[13] H. Huang, Q. Zeng, X. Pei, S. C. Wong, and P. Xu, "Predicting crash frequency using an optimised radial basis function neural network model", *Transportmetrica A: Transport Science*, vol. 12, pp. 330–345, 4 Apr. 2016, ISSN: 2324-9935. DOI: 10.1080/23249935.2015.1136008.

[14] R. Yu and X. Liu, "Study on traffic accidents prediction model based on rbf neural network", *2010 2nd International Conference on Information Engineering and Computer Science*, pp. 1–4, Dec. 2010. DOI: 10.1109/ICIECS.2010.5678126.

[15] B. Pérez-Sánchez, O. Fontenla-Romero, and B. Guijarro-Berdiñas, "A review of adaptive online learning for artificial neural networks", *Artificial Intelligence Review*, vol. 49, pp. 281–299, 2 Feb. 2018, ISSN: 0269-2821. DOI: 10.1007/s10462-016-9526-2.

[16] A. Agarwal, "Predicting road accident risk using google maps images and a convolutional neural network", *International Journal of Artificial Intelligence & Applications*, vol. 10, pp. 49–59, 6 Nov. 2019, ISSN: 09762191. DOI: 10.5121/ijaia.2019.10605.

[17] B. Kumeda, F. Zhang, F. Zhou, S. Hussain, A. Almasri, and M. Assefa, "Classification of road traffic accident data using machine learning algorithms", *2019 IEEE 11th International Conference on Communication Software and Networks (ICCSN)*, pp. 682–687, Jun. 2019. DOI: 10.1109/ICCSN.2019.8905362.

[18] S. Moosavi, M. H. Samavatian, S. Parthasarathy, R. Teodorescu, and R. Ramnath, "Accident risk prediction based on heterogeneous sparse data: New dataset and insights", *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp. 33–42, Nov. 2019. DOI: 10.1145/3347146.3359078.

[19] H. Zhao, J. Zhang, X. Li, Q. Wang, and H. Zhu, "Deep learning-based prediction of traffic accident risk in vehicular networks", *2020 IEEE Globecom Workshops (GC Wkshps*, pp. 1–5, Dec. 2020. DOI: 10.1109/GCWkshps50303.2020.9367497.

[20] T. Huang, S. Wang, and A. Sharma, "Highway crash detection and risk estimation using deep learning", *Accident Analysis & Prevention*, vol. 135, p. 105 392, Feb. 2020, ISSN: 00014575. DOI: 10.1016/j.aap.2019.105392.

[21] S. Lee, J. H. Kim, J. Park, C. Oh, and G. Lee, "Deep-learning-based prediction of high-risk taxi drivers using wellness data", *International Journal of Environmental Research and Public Health*, vol. 17, p. 9505, 24 Dec. 2020, ISSN: 1660-4601. DOI: 10.3390/ijerph17249505.

[22] P. Li, M. Abdel-Aty, and J. Yuan, "Real-time crash risk prediction on arterials based on lstm-cnn", *Accident Analysis & Prevention*, vol. 135, p. 105 371, Feb. 2020, ISSN: 00014575. DOI: 10.1016/j.aap.2019.105371.

[23] W. Wang, S. Yang, and W. Zhang, "Risk prediction on traffic accidents using a compact neural model for multimodal information fusion over urban big data", *arXiv preprint arXiv:2103.05107*, Feb. 2021.

[24] Z. Purkrábková, J. Růžička, Z. Bělinová, and V. Korec, "Traffic accident risk classification using neural networks", *Neural Network World*, vol. 31, pp. 343–353, 5 2021, ISSN: 23364335. DOI: 10.14311/NNW.2021.31.019.

[25] L. Brühwiler, C. Fu, H. Huang, L. Longhi, and R. Weibel, "Predicting individuals' car accident risk by trajectory, driving events, and geographical context", *Computers, Environment and Urban Systems*, vol. 93, p. 101 760, Apr. 2022, ISSN: 01989715. DOI: 10.1016/j.compenvurbsys.2022.101760.

[26] D.-J. Lin, M.-Y. Chen, H.-S. Chiang, and P. K. Sharma, "Intelligent traffic accident prediction model for internet of vehicles with deep learning approach", *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, pp. 2340–2349, 3 Mar. 2021, ISSN: 1524-9050. DOI: 10.1109/TITS.2021.3074987.

[27] N. K. Charandabi, A. Gholami, and A. A. Bina, "Road accident risk prediction using generalized regression neural network optimized with self-organizing map", *Neural Computing and Applications*, vol. 34, pp. 8511–8524, 11 Jun. 2022, ISSN: 0941-0643. DOI: 10.1007/s00521-021-06549-8.

[28] R. C. Park and E. J. Hong, "Urban traffic accident risk prediction for knowledge-based mobile multimedia service", *Personal and Ubiquitous Computing*, vol. 26, pp. 417–427, 2 Apr. 2022, ISSN: 1617-4909. DOI: 10.1007/s00779-020-01442-y.

[29] B. Wang, C. Zhang, Y. D. Wong, L. Hou, M. Zhang, and Y. Xiang, "Comparing resampling algorithms and classifiers for modeling traffic risk prediction", *International Journal of Environmental Research and Public Health*, vol. 19, p. 13 693, 20 Oct. 2022, ISSN: 1660-4601. DOI: 10.3390/ijerph192013693.

[30] B. de Sousa Pereira Amorim, A. A. Firmino, C. de Souza Baptista, G. B. Júnior, A. C. de Paiva, and F. E. de Almeida Júnior, "A machine learning approach for classifying road accident hotspots", *ISPRS International Journal of Geo-Information*, vol. 12, p. 227, 6 May 2023, ISSN: 2220-9964. DOI: 10.3390/ijgi12060227.

[31] Z. Jin and B. Noh, "From prediction to prevention: Leveraging deep learning in traffic accident prediction systems", *Electronics*, p. 4335, 20 Oct. 2023, ISSN: 2079-9292. DOI: 10.3390/electronics12204335.

[32] T. Yuan, W. R. Neto, C. E. Rothenberg, K. Obraczka, C. Barakat, and T. Turletti, "Machine learning for next-generation intelligent transportation systems: A survey", *Transactions on Emerging Telecommunications Technologies*, vol. 33, 4 Apr. 2022, ISSN: 2161-3915. DOI: 10.1002/ett.4427.

[33] S. Chuanxia, Z. Han, and Y. Peixuan, "Machine learning and iots for forecasting prediction of smart road traffic flow", *Soft Computing*, vol. 27, pp. 323–335, 1 Jan. 2023, ISSN: 1432-7643. DOI: 10.1007/s00500-022-07618-3.

[34] D. Wei, "Network traffic prediction based on rbf neural network optimized by improved gravitation search algorithm", *Neural Computing and Applications*, vol. 28, pp. 2303–2312, 8 Aug. 2017, ISSN: 0941-0643. DOI: 10.1007/s00521-016-2193-z.

[35] H. He, Y. Yan, T. Chen, and P. Cheng, "Tree height estimation of forest plantation in mountainous terrain from bare-earth points using a dog-coupled radial basis function neural network", *Remote Sensing*, vol. 11, p. 1271, 11 May 2019, ISSN: 2072-4292. DOI: 10.3390/rs11111271.

[36] F. Lv, S. Yu, C. Wen, and J. C. Principe, "Interpretable fault detection using projections of mutual information matrix", *Journal of the Franklin Institute*, vol. 358, no. 7, pp. 4028–4057, Jul. 2021. DOI: 10.1016/j.jfranklin.2021.02.016.

[37] G. Lemaitre, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning", *Journal of machine learning research*, vol. 18, no. 17, pp. 1–5, 2017.

[38] A. S. Hussein, T. Li, C. W. Yohannese, and K. Bashir, "A-smote: A new preprocessing approach for highly imbalanced datasets by improving smote", *International Journal of Computational Intelligence Systems*, vol. 12, no. 2, pp. 1412–1422, 2019. DOI: 10.2991/ijcis.d.191114.002.

[39] A. R. B. Alamsyah, S. R. Anisa, N. S. Belinda, and A. Setiawan, "Smote and nearmiss methods for disease classification with unbalanced data", *Proceedings of The International Conference on Data Science and Official Statistics*, vol. 2021, pp. 305–314, 1 Jan. 2022, ISSN: 2809-9842. DOI: 10.34123/icdsos.v2021i1.240.

[40] Ž. Đ. Vujovic, "Classification model evaluation metrics", *International Journal of Advanced Computer Science and Applications*, vol. 12, 6 2021, ISSN: 21565570. DOI: 10.14569/IJACSA.2021.0120670.

[41] D. Berrar, "Cross-validation", *Encyclopedia of Bioinformatics and Computational Biology*, pp. 542–545, 2019. DOI: 10.1016/B978-0-12-809633-8.20349-X.

[42] P. Marcillo, Á. L. V. Caraguay, and M. Hernández-Álvarez, "A systematic literature review of learning-based traffic accident prediction models based on heterogeneous sources", *Applied Sciences*, vol. 12, p. 4529, 9 Apr. 2022, ISSN: 2076-3417. DOI: 10.3390/app12094529.

Appendix A

Development software, packages, and code lines

A.1 cor R package

cor compute the variance of x and the covariance or correlation of x and y if these are vectors. If x and y are matrices, then the covariances (or correlations) between the columns of x and the columns of y are computed.

Usage

```
cov(x, y=NULL, use="everything", method=c("pearson", "kendall", "
    spearman"))
```

Parameters

x: A numeric vector, matrix, or data frame.

y: **NULL** (default) or a vector, matrix, or data frame with compatible dimensions to x. The default is equivalent to $y = x$ (but more efficient).

A.2 MI correlation graph code

This line of code was implemented in the R programming language to generate the MI matrix graph.

```
library(readxl)
library(corrplot)
library(RColorBrewer)

datos <- read.csv(file = "C:/Users/Cristian/Documents/EPN/Maestría
    en Computación/OneDrive - Escuela Politécnica Nacional/2022-B/
    Tesis/Dataset/20221215_151443_clustering.csv")
datos <- datos[, c(
```

```
    "steering_angle", "speed", "rpm", "acceleration", "throttle_
       position", "engine_temperature", "system_voltage", "
       barometric_pressure", "distance_travelled", "latitude", "
       longitude", "heart_rate", "current_weather", "temperature",
       "real_feel_temperature", "wind_speed", "relative_humidity",
       "uv_index", "cloud_cover", "ceiling", "pressure", "
       precipitation", "accidents_onsite")]

# Correlation Graph
png(height = 1024, width = 1024, file = "C:/Users/Cristian/
   Documents/EPN/Maestría en Computación/OneDrive - Escuela
   Politécnica Nacional/2022-B/Tesis/Images/MI_matrix.png")
corr <- round(cor(datos), 1)

mi_matrix.plot <- corrplot(corr,
    type = "upper",
    is.corr = FALSE,
    order = "hclust",
    addCoef.col = TRUE,
    method = "square",
    col = colorRampPalette(c(
        brewer.pal(7, "Set2")[2],
        "white", brewer.pal(7, "Set2")[1]
    ))(100),
    tl.col = "black",
    tl.srt = 60,
    number.cex = 0.75,
    diag = TRUE
)
mi_matrix.plot
dev.off()
```

A.3 mutual_info_classif Python function

Estimate Mutual Information for a discrete target variable. MI between two random variables is a non-negative value, which measures the dependency between the variables. It equals zero if two random variables are independent, and higher values mean higher dependency. The function relies on nonparametric methods based on entropy estimation from k-nearest neighbor distances.

Usage

```
sklearn.feature_selection.mutual_info_classif(X, y)
```

Parameters
X: (n_samples, n_features) Feature matrix.
y: (n_samples,) Target vector.

A.4 imbalanced-learn Python toolkit

Imbalanced-learn is an open-source, MIT-licensed library that relies on scikit-learn (sklearn) and provides tools for classification with imbalanced classes.

A.5 Nearmiss Python method

Class to perform undersampling based on NearMiss methods.

Usage

```
imblearn.under_sampling.NearMiss(sampling_strategy='auto', version
    =1, n_neighbors=3)
```

Parameters
sampling_strategy: Sampling information to sample the dataset.
version: Version of the NearMiss to use, possible values are 1, 2, or 3.
n_neighbors: Neighborhood size to consider to compute the average distance to the minority point samples.

A.6 GridSearchCV Python class

Exhaustive search over specified parameter values for an estimator. The estimator parameters used to apply these methods are optimized by cross-validated grid search over a parameter grid.

Usage

```
sklearn.model_selection.GridSearchCV(estimator, param_grid, scoring
    =None)
```

Parameters
estimator: This is assumed to implement the scikit-learn estimator interface.

param_grid: Dictionary with parameters names (str) as keys and lists of parameter settings to try as values.
scoring: A strategy to evaluate the performance of the cross-validated model on the test set.

A.7 TensorFlow

TensorFlow is an end-to-end open source platform for ML. TensorFlow makes it easy to create ML models.

Installation

```
# Current stable release for CPU and GPU
pip install tensorflow
```

A.8 RandomForestClassifier Python class

It is a random forest classifier. A random forest is a meta-estimator that fits several decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. Trees in the forest use the best-split strategy.

Usage

```
sklearn.ensemble.RandomForestClassifier(n_estimators=100, criterion
    ='gini', max_depth=None,)
```

Parameters
n_estimators: The number of trees in the forest.
criterion: The function to measure the quality of a split.
max_depth: The maximum depth of the tree.

A.9 GaussianProcessClassifier Python class

Gaussian process classification (GPC) based on Laplace approximation. Internally, the Laplace approximation approximates the non-Gaussian posterior by a Gaussian.

Usage

```
sklearn.gaussian_process.GaussianProcessClassifier(kernel=None,
    max_iter_predict=100)
```

Parameters
kernel: The kernel specifies the covariance function of the GP.

max_iter_predict: The maximum number of iterations in Newton's method for approximating the posterior during prediction. Smaller values will reduce computation time at the cost of worse results.

A.10 SVC Python class

C-Support Vector Classification. The implementation is based on libsvm. The fit time scales at least quadratically with the number of samples and may be impractical beyond tens of thousands of samples.

Usage

```
sklearn.svm.SVC(C=1.0, kernel='rbf', gamma='scale')
```

Parameters

C: Regularization parameter. The strength of the regularization is inversely proportional to C. It must be strictly positive.
kernel: Specifies the kernel type to be used in the algorithm. If none is given, *rbf* will be used. If a callable is given, it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shapes (n_samples, n_samples).
gamma: Kernel coefficient for *rbf*, *poly*, and *sigmoid*.

A.11 MLPClassifier Python class

Multi-layer Perceptron classifier. This model optimizes the log-loss function using LBFGS or stochastic gradient descent.

Usage

```
sklearn.neural_network.MLPClassifier(hidden_layer_sizes=(100,),
    activation='relu', solver='adam', alpha=0.0001, max_iter=200,
    learning_rate='constant',)
```

Parameters

hidden_layer_sizes: The ith element represents the number of neurons in the ith hidden layer.
activation: Activation function for the hidden layer.
solver: The solver for weight optimization.
alpha: Strength of the L2 regularization term. When added to the loss, the L2 regularization term is divided by the sample size.
max_iter: Maximum number of iterations.
learning_rate: The initial learning rate used.

A.12 Visual Studio Code IDE

Visual Studio Code is a lightweight but powerful source code editor that runs on your desktop and is available for Windows, macOS, and Linux. It has built-in support for JavaScript, TypeScript, and Node.js and a rich ecosystem of extensions for other languages and runtimes (C++, C#, Java, Python, PHP, Go, and NET).

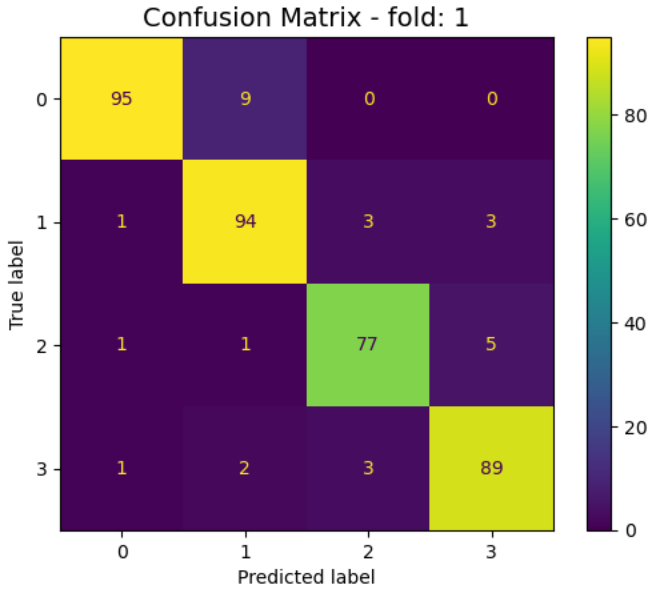
Installation

- Download the Visual Studio Code installer.
- Once it is downloaded, run the installer. The installation will only take a minute.

Appendix B

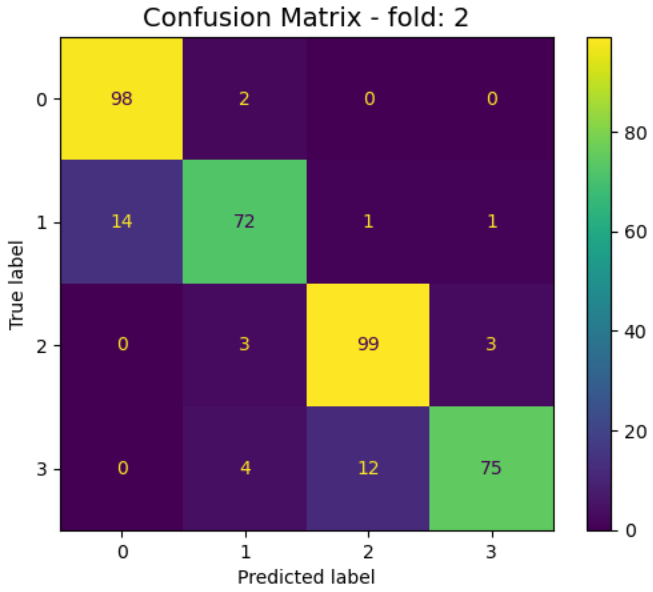
Model evaluation results

B.1 CNN model



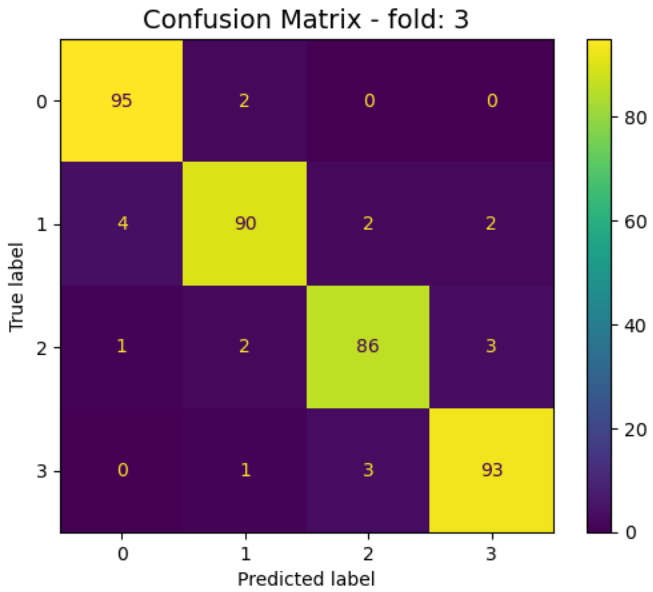
Accuracy:	0.9270
Specificity:	0.9748
Sensitivity:	0.9244

Figure B.1: 1-Fold CNN



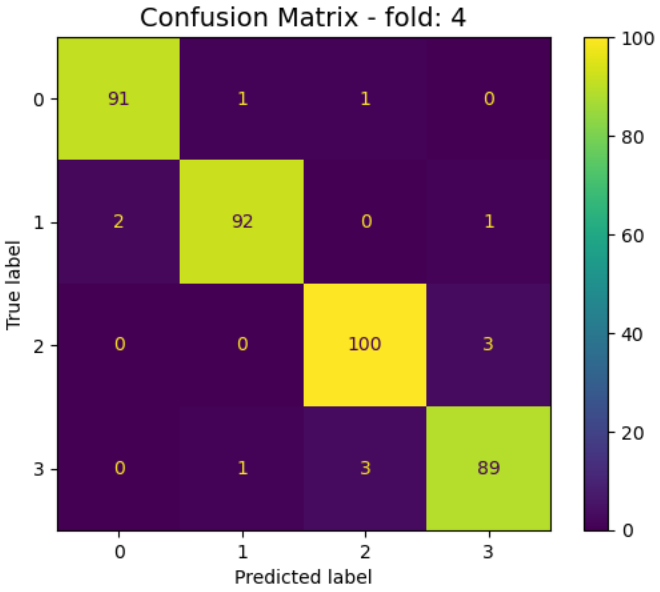
Accuracy: 0.8958
Specificity: 0.9650
Sensitivity: 0.8913

Figure B.2: 2-Fold CNN



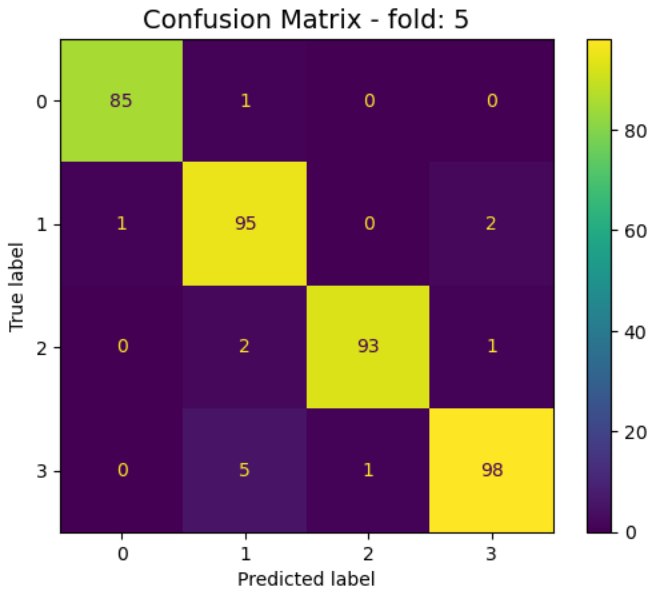
Accuracy: 0.9479
Specificity: 0.9826
Sensitivity: 0.9478

Figure B.3: 3-Fold CNN



Accuracy: 0.9687
Specificity: 0.9895
Sensitivity: 0.9686

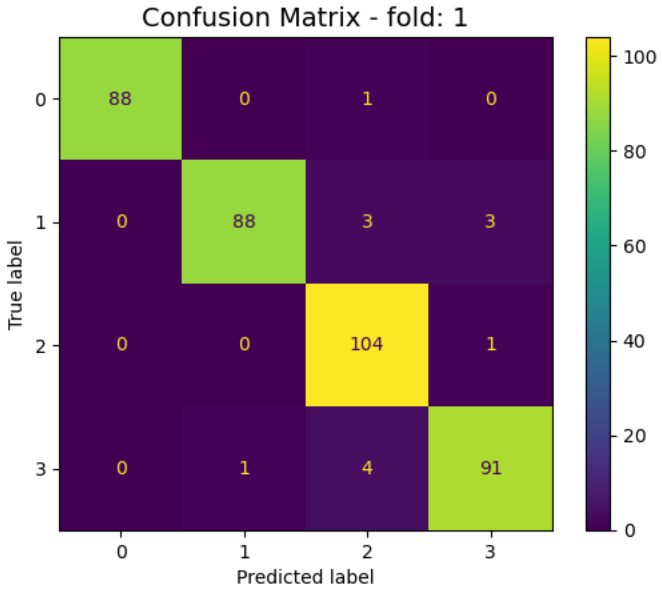
Figure B.4: 4-Fold CNN



Accuracy: 0.9661
Specificity: 0.9886
Sensitivity: 0.9672

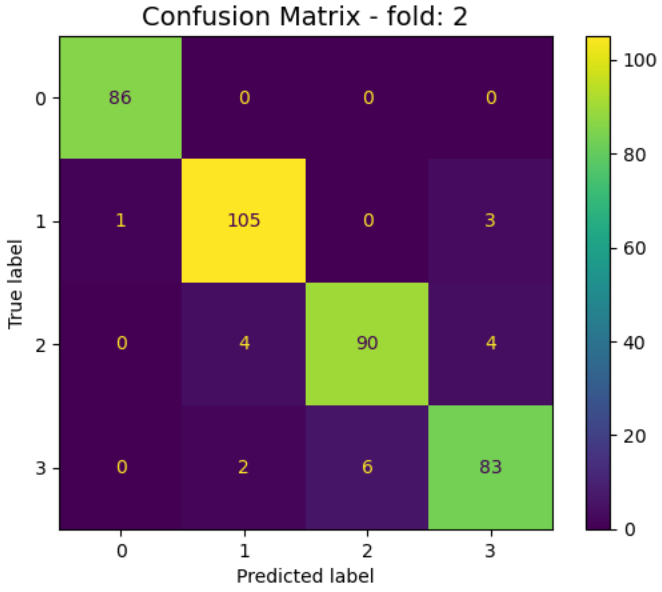
Figure B.5: 5-Fold CNN

B.2 CNN-RF model



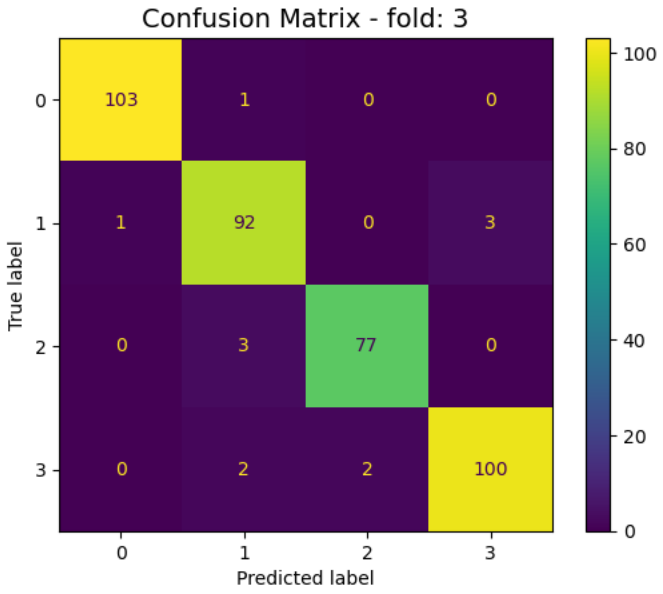
Accuracy:	0.9661
Specificity:	0.9884
Sensitivity:	0.9658

Figure B.6: 1-Fold CNN-RF



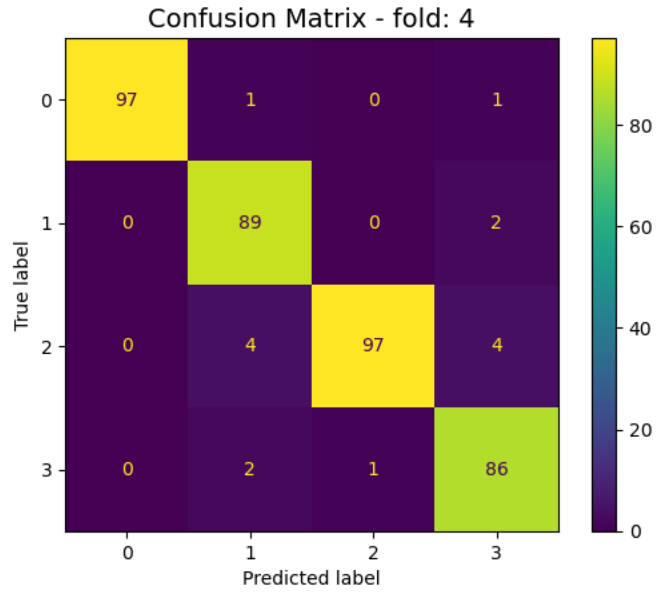
Accuracy:	0.9479
Specificity:	0.9824
Sensitivity:	0.9484

Figure B.7: 2-Fold CNN-RF



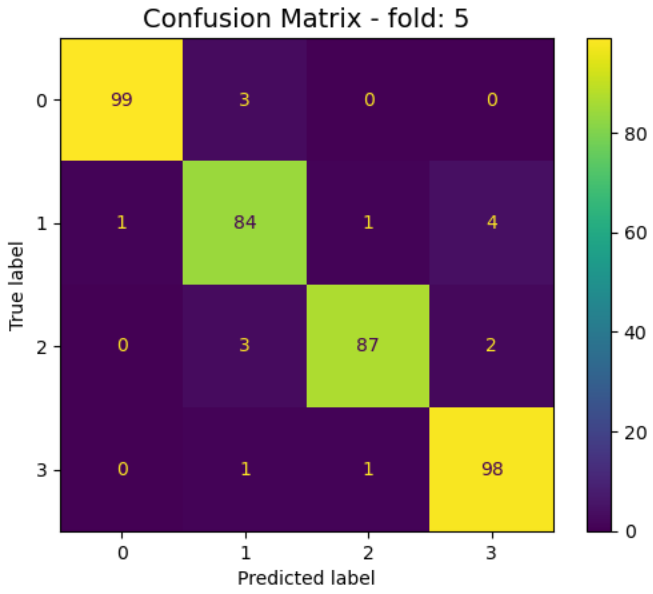
Accuracy:	0.9687
Specificity:	0.9895
Sensitivity:	0.9681

Figure B.8: 3-Fold CNN-RF



Accuracy:	0.9609
Specificity:	0.9871
Sensitivity:	0.9619

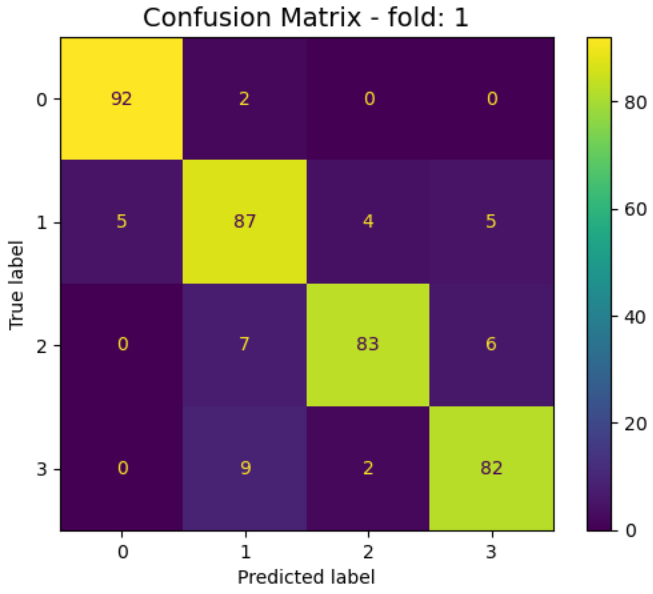
Figure B.9: 4-Fold CNN-RF



Accuracy: 0.9583
Specificity: 0.9861
Sensitivity: 0.9573

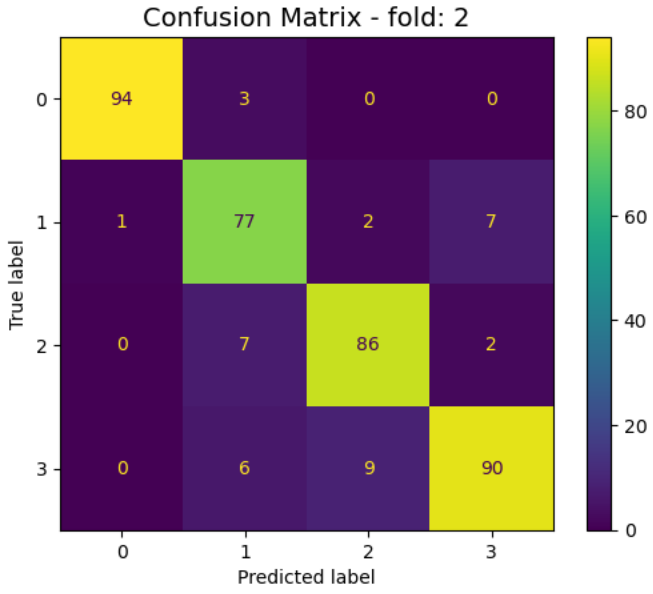
Figure B.10: 5-Fold CNN-RF

B.3 GPC-RBF model



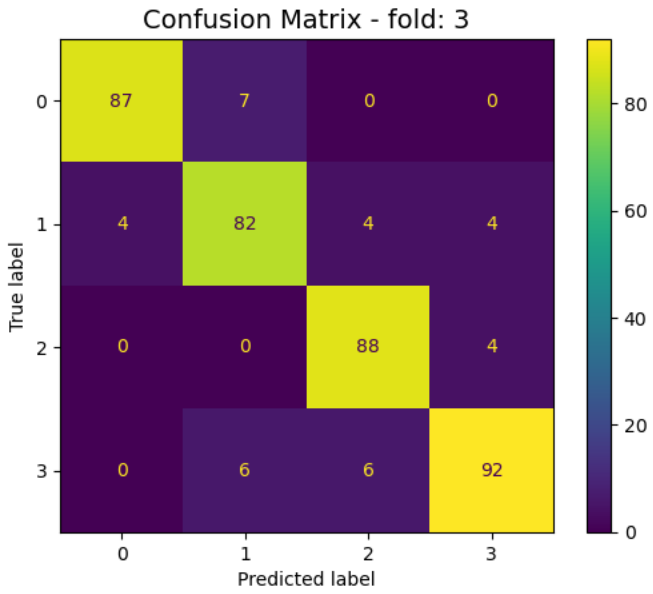
Accuracy: 0.8958
Specificity: 0.9651
Sensitivity: 0.8966

Figure B.11: 1-Fold GPC-RBF



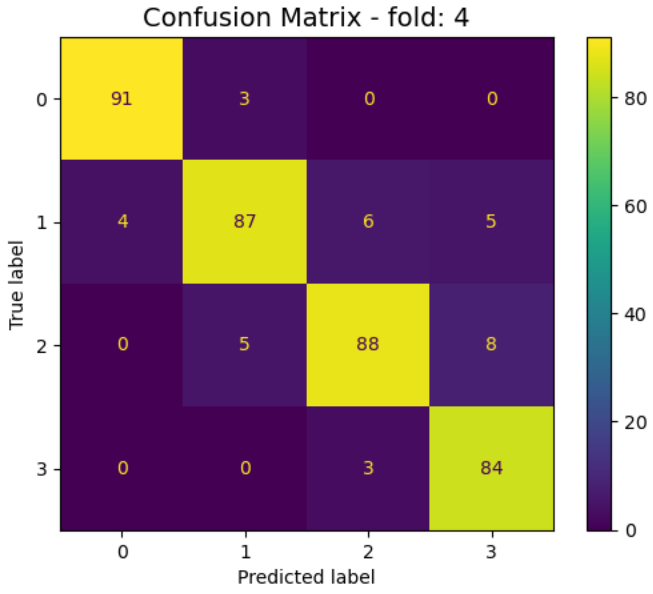
Accuracy: 0.9036
Specificity: 0.9680
Sensitivity: 0.9041

Figure B.12: 2-Fold GPC-RBF



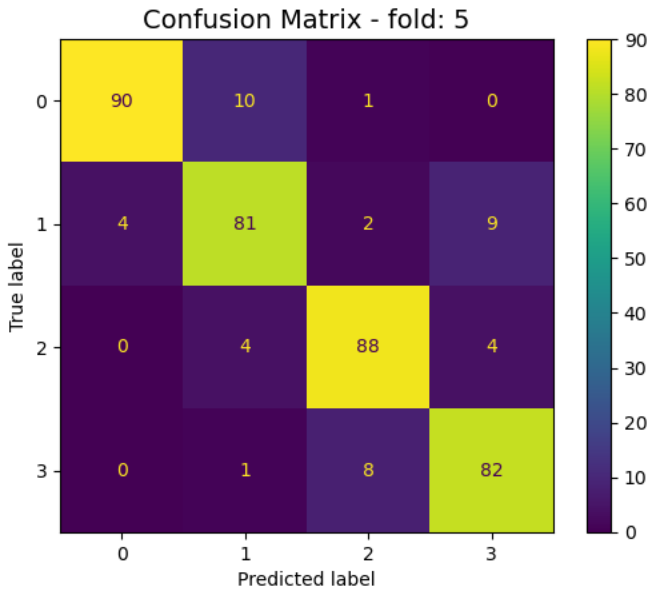
Accuracy: 0.9088
Specificity: 0.9696
Sensitivity: 0.9097

Figure B.13: 3-Fold GPC-RBF



Accuracy: 0.9114
Specificity: 0.9705
Sensitivity: 0.9144

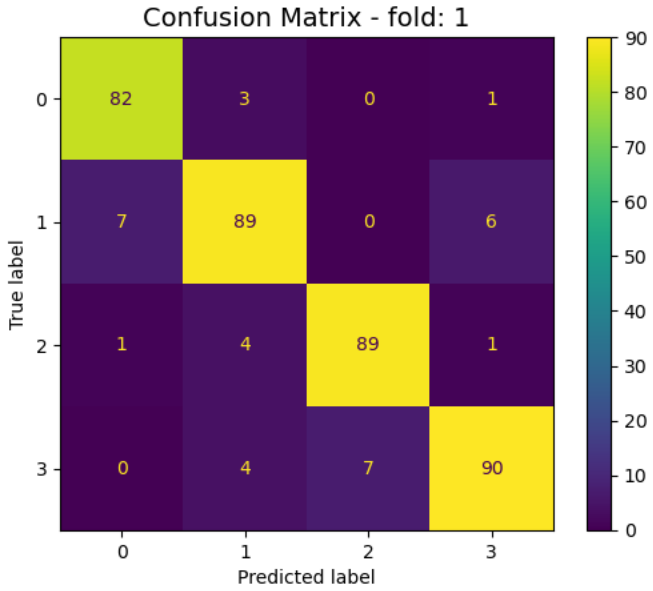
Figure B.14: 4-Fold GPC-RBF



Accuracy: 0.8880
Specificity: 0.9628
Sensitivity: 0.8881

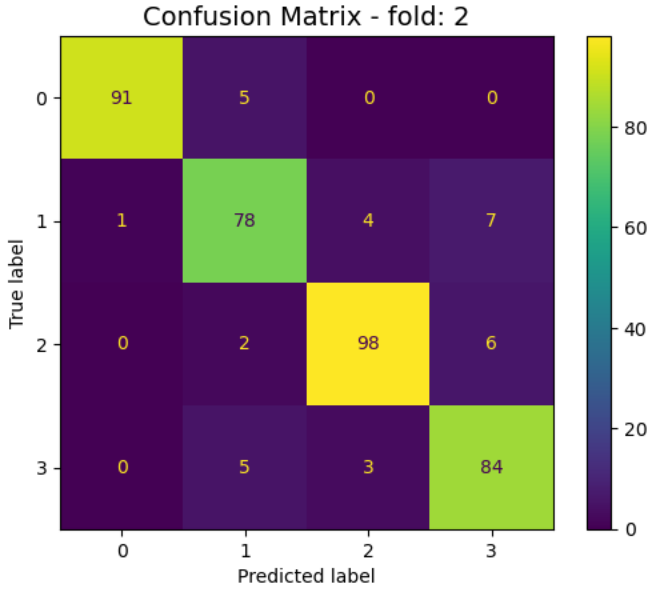
Figure B.15: 5-Fold GPC-RBF

B.4 SVC-RBF model



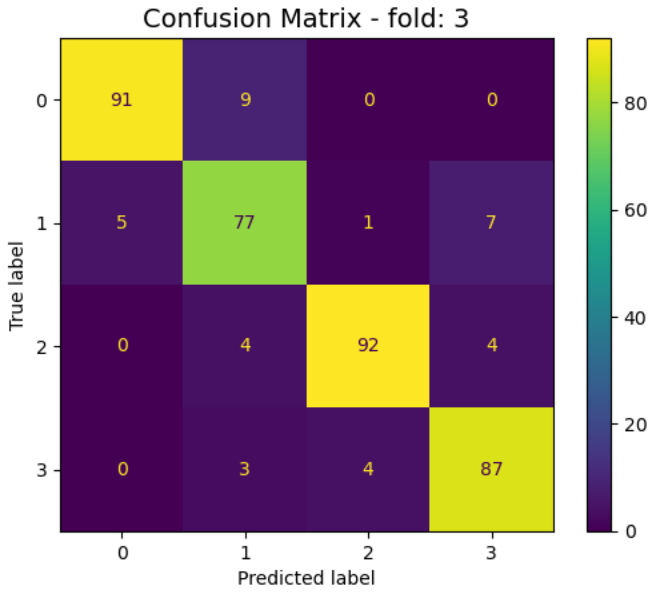
Accuracy: 0.9114
Specificity: 0.9704
Sensitivity: 0.9134

Figure B.16: 1-Fold SVC-RBF



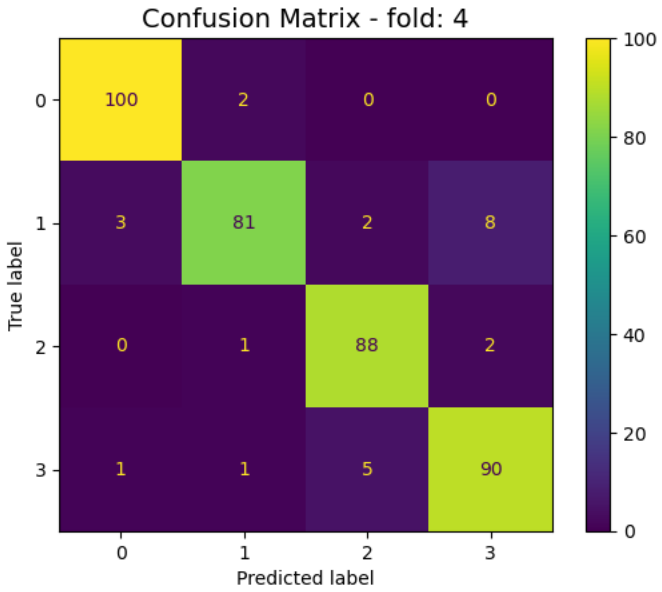
Accuracy: 0.9140
Specificity: 0.9715
Sensitivity: 0.9130

Figure B.17: 2-Fold SVC-RBF



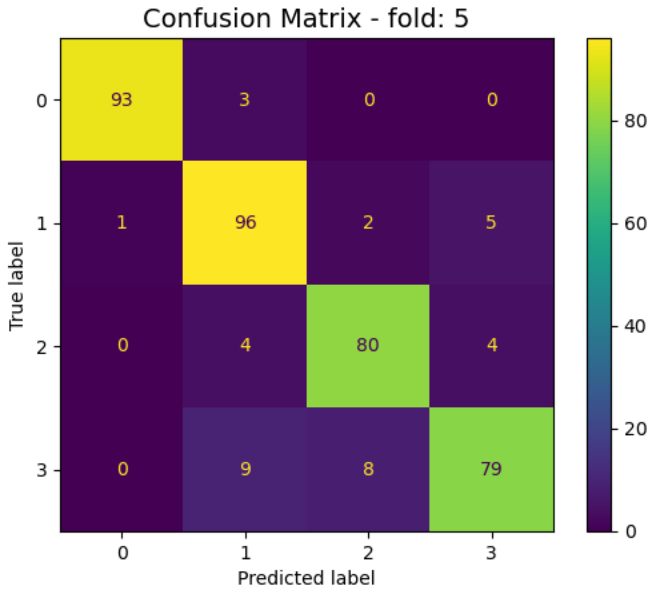
Accuracy:	0.9036
Specificity:	0.9681
Sensitivity:	0.9027

Figure B.18: 3-Fold SVC-RBF



Accuracy:	0.9348
Specificity:	0.9783
Sensitivity:	0.9342

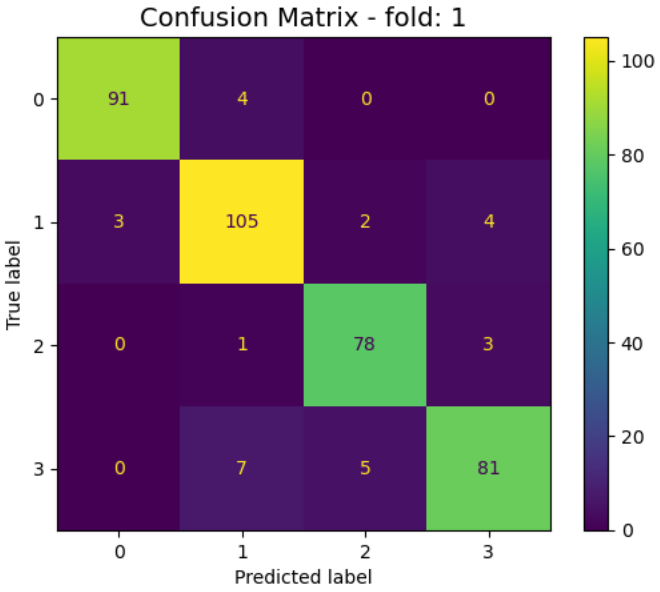
Figure B.19: 4-Fold SVC-RBF



Accuracy:	0.9062
Specificity:	0.9685
Sensitivity:	0.9059

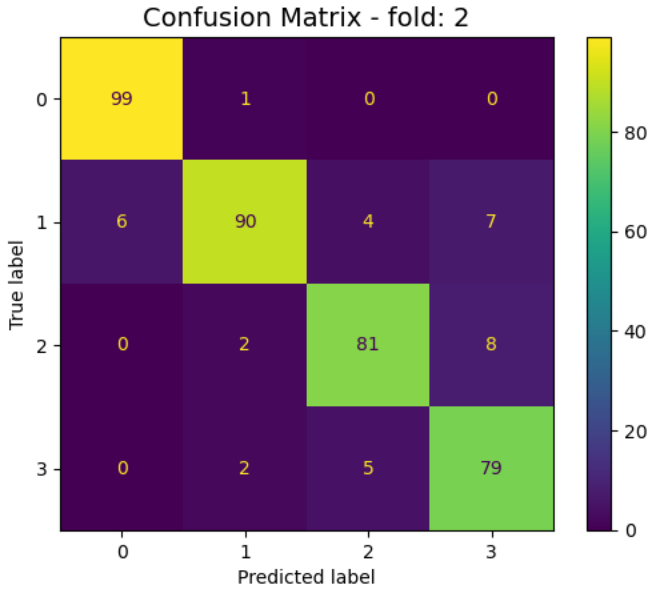
Figure B.20: 5-Fold SVC-RBF

B.5 MLP model



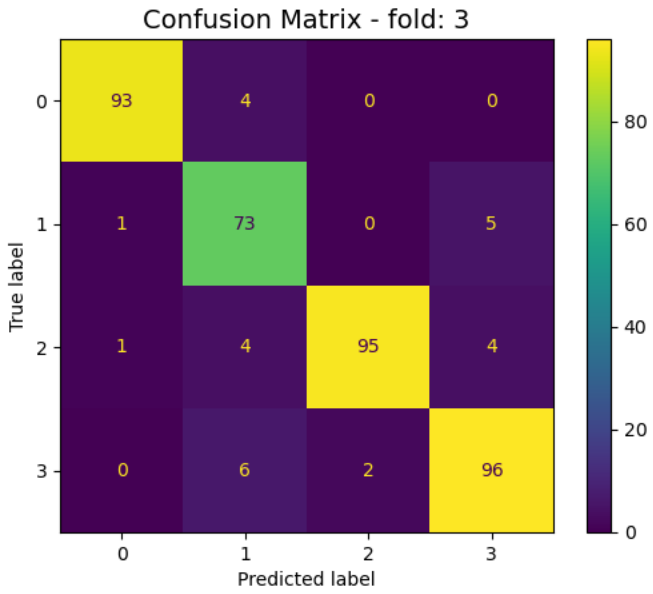
Accuracy:	0.9244
Specificity:	0.9744
Sensitivity:	0.9252

Figure B.21: 1-Fold MLP



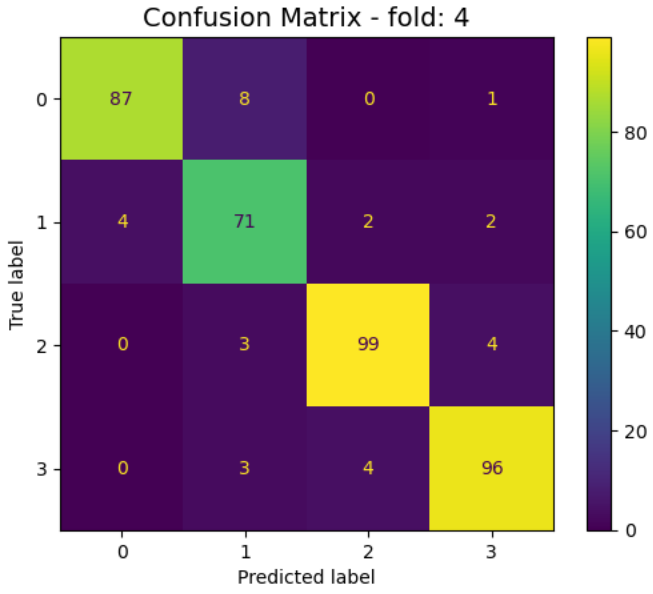
Accuracy: 0.9088
Specificity: 0.9699
Sensitivity: 0.9099

Figure B.22: 2-Fold MLP



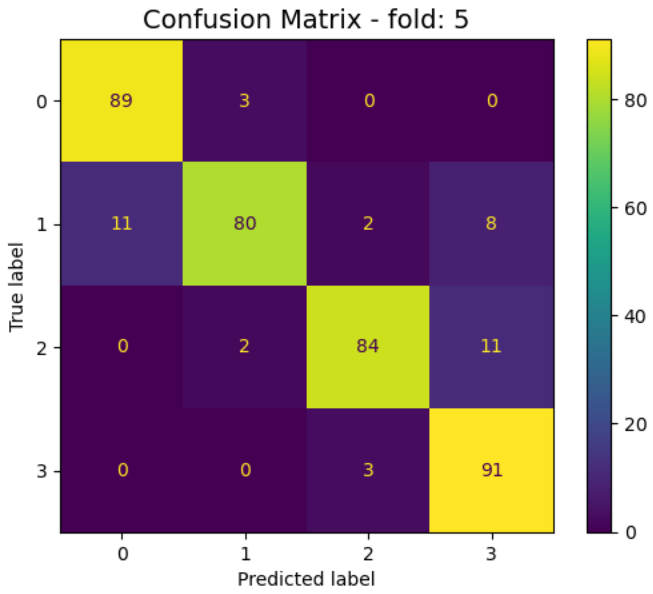
Accuracy: 0.9296
Specificity: 0.9769
Sensitivity: 0.9298

Figure B.23: 3-Fold MLP



Accuracy: 0.9192
Specificity: 0.9734
Sensitivity: 0.9177

Figure B.24: 4-Fold MLP



Accuracy: 0.8958
Specificity: 0.9654
Sensitivity: 0.8983

Figure B.25: 5-Fold MLP