Neural Network Modeling to Assess the Risk of Eclampsia in Pregnant Women

Leveraging Hadoop, SQLPyspark, TensorFlow, and ANN for Maternal Health Risk Analysis

Nayane Oliveira Araujo   
*MSc. in Data Analytics Program*   
*CCT College Dublin*Dublin, Ireland   
2021264@student.cct.ie

*Abstract*

This study investigates/understands the application of Artificial neural network models such as Artificial Neural network (ANN) and Recurrent Neural Network (RNN), processed through PYSPARK and SQLPYSPARD with a focus on modeling and classifying prediction models using Neural Networks

The analyzed dataset is based on a study of women's health during pregnancy, due to changes in some risk factors such as age, blood pressure, heart rate, blood sugar level, and body temperature.

With this information, data import, information extraction, modeling, training and normalization, creation of pipelines, data hyperparameters to perform classification, and application of ANN and RNN neural networks began. During the application of an ANN model, the accuracy obtained was 0.656 and for the RNN, the accuracy found was 0.604, where it will be highlighted that the ANN Neural Model had a better performance than the RNN in this study.

*Keywords*

*Big Data Analytics, Neural Networks, Hadoop, SQLPySpark, Tensorflow, maternal health risk*.

*Introduction*

The immense demand that was generated by the excessive amount of data collected by all of humanity at every moment brings the opportunity and the need for programs that can process and store this information in a safe, efficient way, that generates insights, visualizes trends, presents proposals collaboration to make key decisions for business, health research, non-governmental organizations and any interested parties wishing to improve the data generated.

*Big Data, Storage, Processing and Usaging*

In data science, nowadays it is adapting and building ways to deal with Big Data, seeking applications where analyses can be stored and managed efficiently that contribute to large studies resulting in a rich use of data. Studies such as that by Bellazzi (2014), which explores 'Big Data and complex biomedical analyses: a challenge for bioinformatics research', illustrate how the analysis of large volumes of genomic data can contribute to personalized therapies and disease prevention.[1]

Hadoop is a software framework that enables the processing large data sets across computer clusters. It is known for its ability to handle large volumes of data and its fault tolerance. Hadoop uses the HDFS distributed file system and the MapReduce programming model. [ 2]

In the field of cloud computing and big data analysis, Hashem et al. (2015) present a comprehensive review of the growth of "big data" and open questions for research, highlighting challenges and opportunities associated with cloud computing (Hashem et al., 2015, p. 98-115). This discussion is fundamental to understanding how cloud computing infrastructure can be scaled to efficiently manage growing volumes of data. [ 3]

On the other hand, Zhou et al. (2014) specifically focus on the application of big data in healthcare, exploring how these technologies can be used to improve the efficiency of healthcare systems and patient care (Zhou et al., 2014, p. 2(3)) . This study is essential to understand the transformations in the health field driven by the advancement of big data technologies. [ 4]

PySpark is another very powerful application for processing Big Data and a practical example is the study by Mittal and Goel (2020), which used PySpark to analyze tweets and identify trends in sentiment regarding products. This study highlights the usefulness of PySpark in big data analysis. [ 5]

Neural networks and Big Data have the crucial ability to process large volumes of complex data efficiently. A relevant study in this context is that carried out by LeCun et al. (2015), who demonstrated the effectiveness of convolutional neural networks (CNNs) in analyzing large volumes of visual data. The study highlighted how CNNs can automatically learn relevant features from data and significantly improve performance in classification and pattern recognition tasks. [6]

*License of use:*

That dataset was collected from the UCI Machine Learning Repository. [ 7] The Data has been collected from different hospitals, community clinics, and maternal health care in the rural areas of Bangladesh through the IoT-based risk monitoring system. The variables included in the data set are the following attributes: Age, SystolicBP, Blood Pressure in mmHg, DiastolicBP - Lower value of Blood Pressure in mmHg, BS - Blood glucose levels in terms of a molar concentration, mmol/ L, BodyTem (temperature in Fahrenheit), HeartRate,

RiskLevel.

The data set consists of 1014 rows distributed in 7 Columns and has been added to HADOOP's repository through SQLPYSPARK, as a result, a temporary visualization was created to understand the dataset to extract information and detect missing values. Initially, the imported libraries to manipulate and analyze the dataset printing Using the SQLPYSPARK, it was also possible to quantify and group the number of pregnant women by type of risk present in the study. Therefore, there are 406 women at low risk, 336 women at intermediate risk, and 272 at high risk of eclampsia. Still using the SQL tool, it was also possible to plot an age distribution graph by frequency.

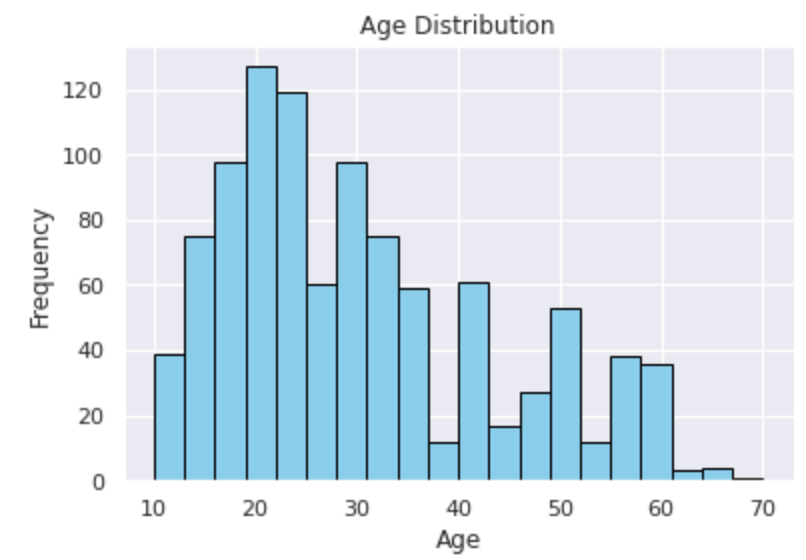
******

Figure 1 Distribution chart by age

Based on the dependencies of the RDDs by SQPPYSPARK, it was possible to calculate the average risk for different age groups. Where it is highlighted that pregnant women are in the age group with the highest risk and that of 31 to 40 years of age with 49.72%

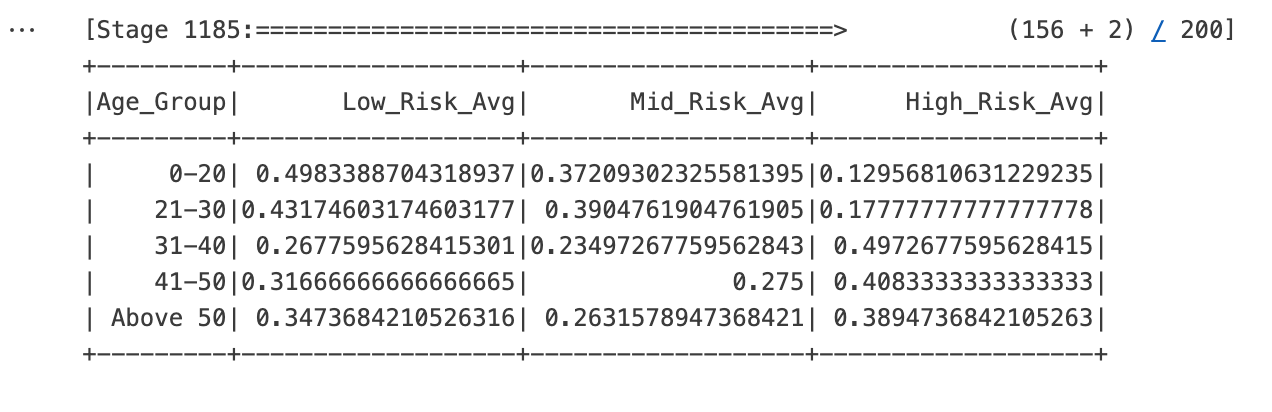
******

Figure 2 Shows the average by Age and Risk Groups

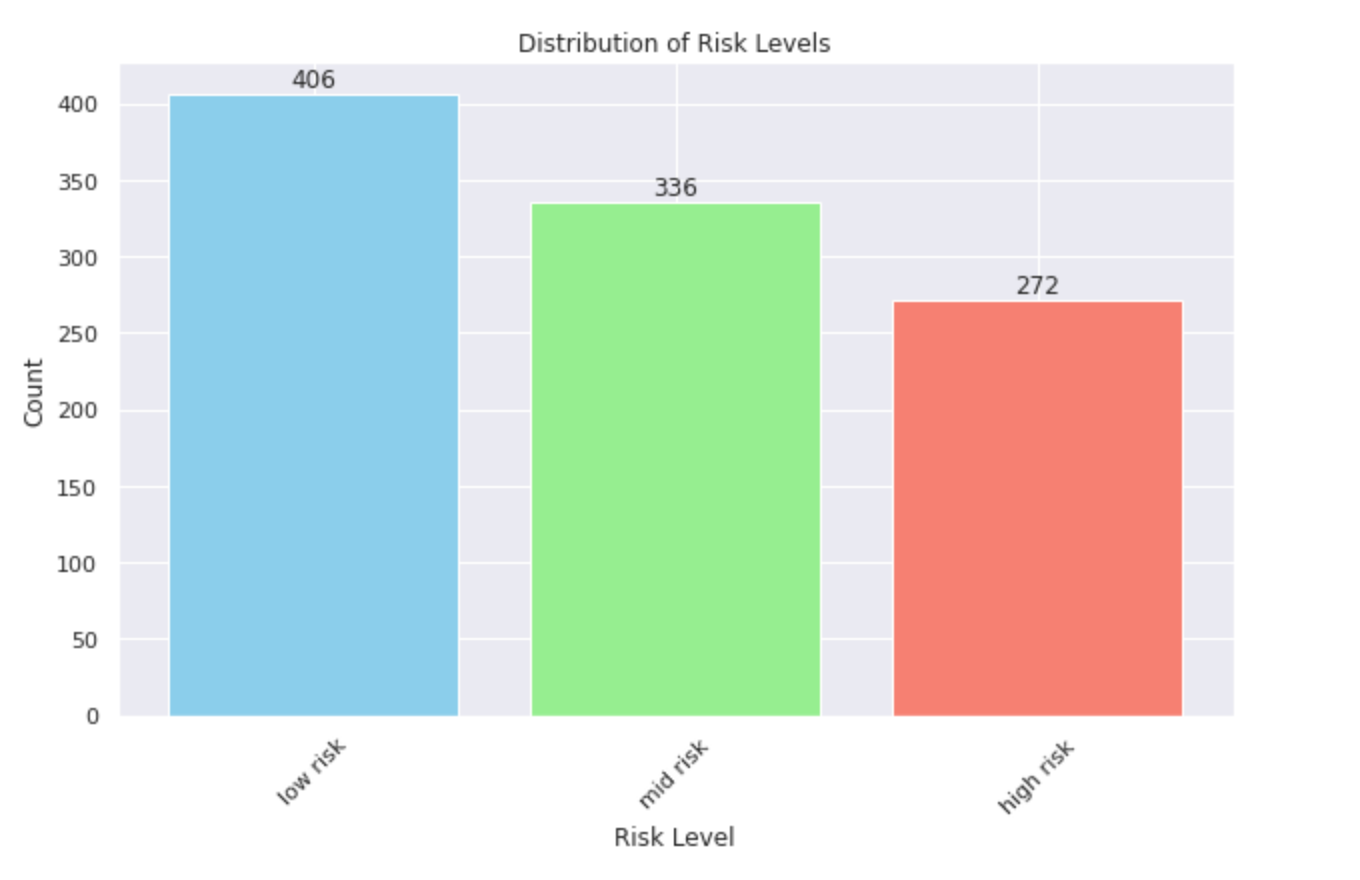
matplotlib.pyplot Library one can obtain the distribution of risk levels figure and also a Comparative chart for the average of systolic and diastolic blood pressure, by Risk Level see figure, Also visualizing theaverage risk slight figure****

Figure Comparative Average Chart

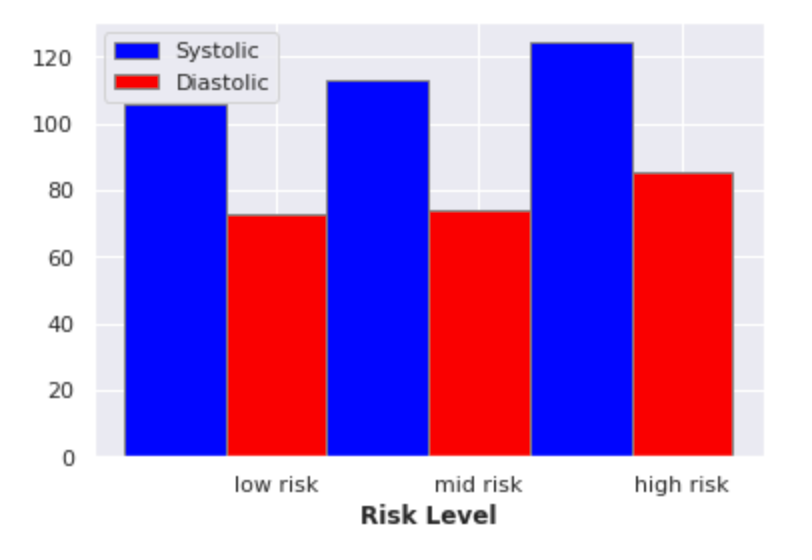


Figure Risk Level Average

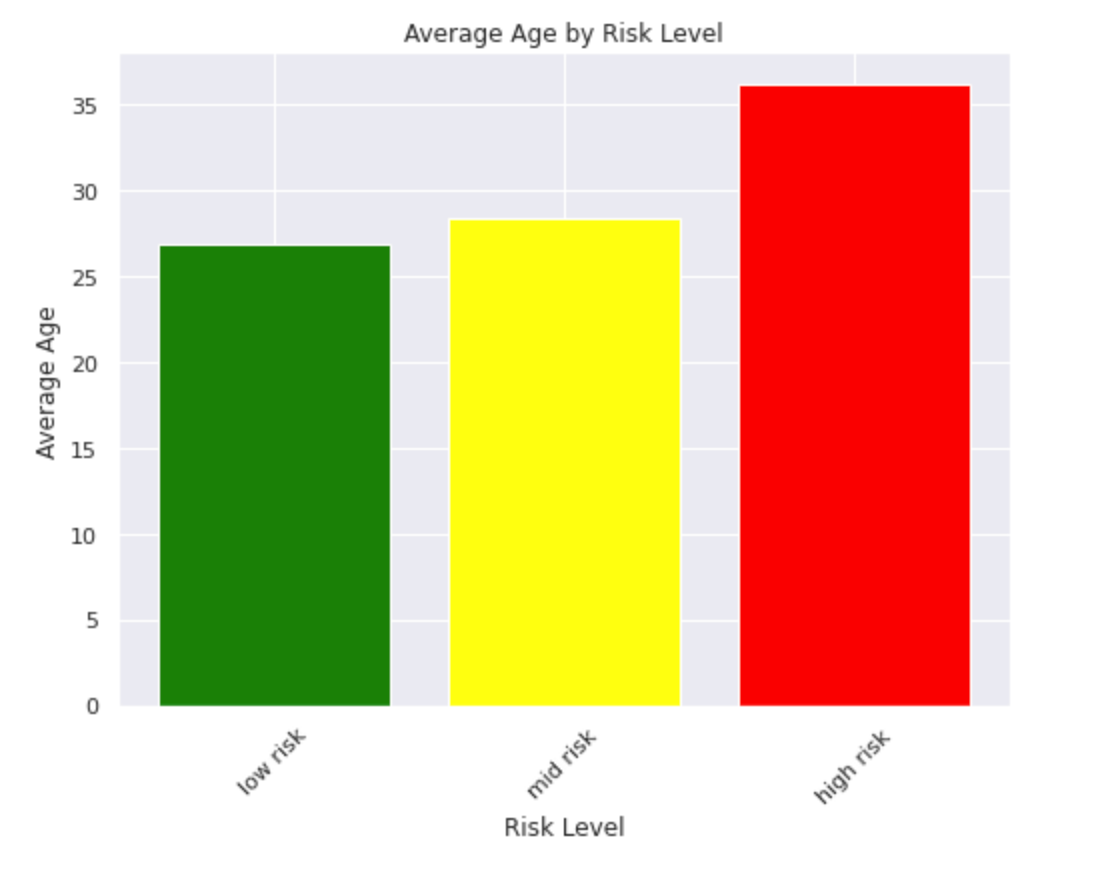


Figure 5 AG. by risk level

information including numpy (for numeric for data manipulation and analysis, including numpy (for numeric operations), pandas (for data manipulation in tabular format), and various components of pyspark.sql and pyspark.ml for Spark operations.

The Spark session is configured and launched with the application name "health\_data", preparing the environment for distributed analytics operations.

*Neural network modeling*

*RNN,* to prepare the training data for the recurrent neural network model, Apache Spark was used. It was necessary to follow some important steps such as the   
transformation with StringIndexer, applied to the 'RiskLevel' column to convert from a categorical label to a string to a numeric format. In sequence the assembly of Vectors that were used to combine multiple columns into a single feature vector, transforming the list of columns such as, 'Age', 'SystolicBP', 'DiastolicBP', 'BS' (Blood Sugar), 'BodyTemp ' and 'HeartRate' into a single vector column called "features", as modeling models in Spark expect input features with this format.

In the standardization stage, the "features" vector is processed to normalize each feature vector to have a norm unitary, helping to accelerate convergence and remove biases due to different measurement scales.   
During Pipeline construction is applied to the set of the fit method that computes the statistics necessary to fit the data. During the prediction phase in this process, the accuracy obtained was 59.27%, a result not considered good for the model. Given this result, hyperparameters were experimented with for the neural network model, using Apache, with the main objective of identifying the best layer architecture for a classifier based on the accuracy metric, evaluating the performance of classification models, considering the proportion of correct answers made by the model about the total samples.

After this process, a confusion matrix was created to generate a visualization of the models' performance, allowing the accuracy of the variations of a model to be visualized by comparing the real values with those predicted by the model, the matrix can be seen in figure [6].

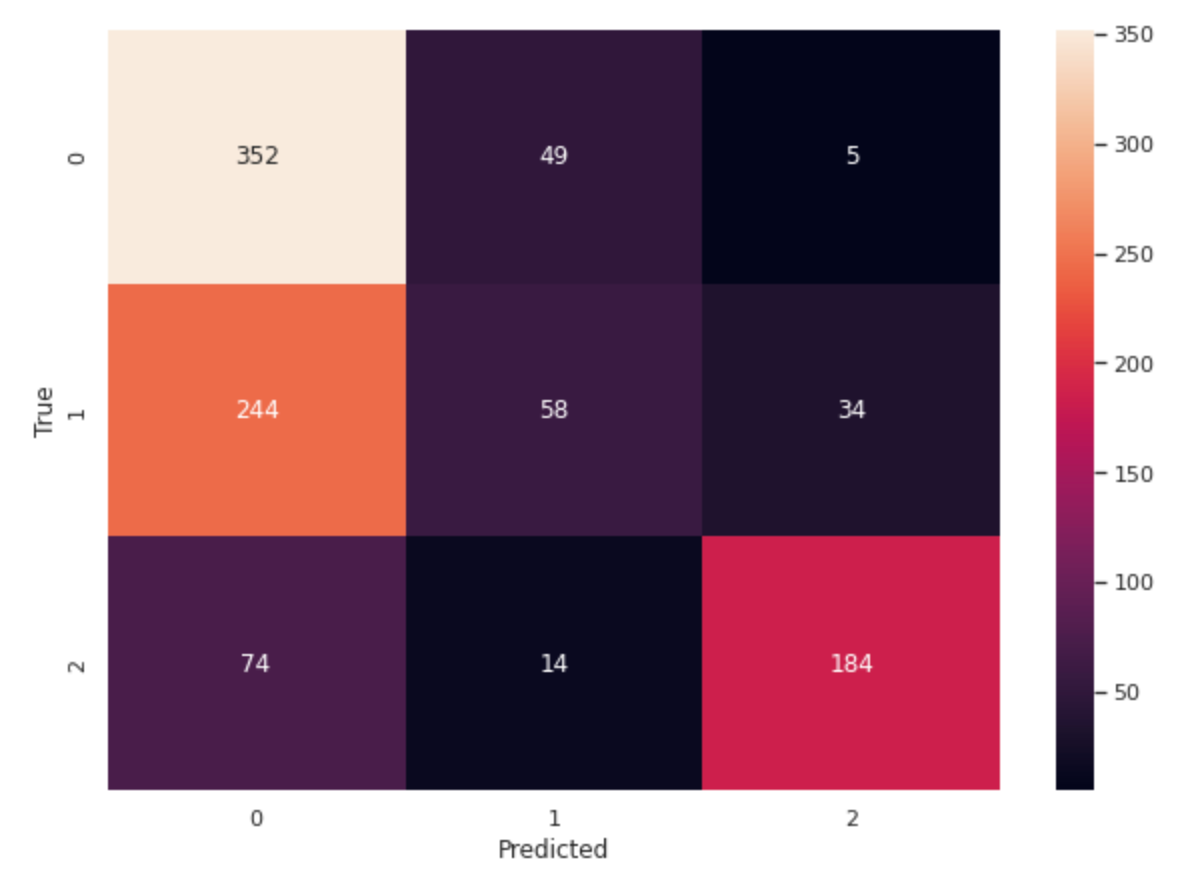
****

Figure 6 Confusion Matrix for RRN models

**ANN**

Evaluation of the classification model (ANN) created with TensorFlow and Keras, following the model training step, details the predictions with the model, converts these predictions into class labels, and then generates a classification report to evaluate the performance of the model.

Artificial neural networks can learn and model complex relationships using datasets, as described in the groundbreaking 2012 study, Krizhevsky, Sutskever and Hinton demonstrated the effectiveness of deep convolutional neural networks through AlexNet, an architecture with five convolutional layers and three fully connected layers, totaling 60 million parameters. This model significantly reduced error rates in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), outperforming previous methods. AlexNet employed techniques such as "dropout" to combat overfitting, proving to be an effective regularization strategy in deeply connected layers, setting a new standard for image classification and driving the development of deep learning in several areas of artificial intelligence (Krizhevsky, Sutskever & Hinton, 2012 ) [8]

To prepare the data for applying artificial neural reactions, it was necessary to convert the DataFrame from Spark to Pandas, to facilitate data manipulation. The initial steps consist of separating Features and Labels, dividing the DataFrame into independent variables (X) containing features such as age, systolic and diastolic pressure, blood glucose level, body temperature and heart rate, and a dependent variable (y) RiskLevel which is the label to be predicted.

The dataset is divided into training and testing sets using the function with 20% of the data reserved for testing. This helps evaluate the model on data that was not used in training, providing a more realistic performance metric. Data normalization

Standard Scaler: The characteristics are normalized to have a mean of 0 and standard deviation of 1, then the Label conversion was performed. A sequential model is created using TensorFlow/Keras. This model consists of two dense layers with relu activation and an output layer with softmax activation to predict the three classes. The model is developed with the Adam optimizer and the categorical loss function, which is designed for multi-class classification problems. The model is trained on the training set for a defined number of epochs that iterates over the entire data set, with validation on 20% of training data to monitor performance and avoid overfitting.

The model is evaluated on the test set to determine test loss and accuracy. These analyses help to understand how well the model is performing on data not seen during training, this model obtained a media test of 60.02.

The presented code plays a fundamental role in the quantitative evaluation of the results of a classification model. Detailed segmentation and analysis of model performance across multiple metrics—precision, recall, and F1 score—enables in-depth interpretation and rigorous validation of model capabilities, which is essential to support any scientific argument or conclusion derived from research results.

The Model Predictions are used to predict the probabilities of each class for the test set. The result is an array where each line represents a test sample. Convection of probabilities into class labels, the Argmax method is used to convert the predicted probabilities into class labels, by extracting the column index with the highest probability, which represents the predicted class.

With One-Hot, the conversion was made to a categorical representation for training the model, which was reverted to class labels to facilitate comparison with predictions.

Calculating various model performance metrics such as precision, recall, and f1-score for each class, as well as weighted average precision, recall, and f1-score based on support (number of true instances for each class).

The report has been converted to a pandas DataFrame for easier viewing and analysis.

Evaluation of the classification model (ANN) created with TensorFlow and Keras, following the model training step, details the predictions with the model, converts these predictions into class labels, and then generates a classification report to evaluate the performance of the model.

The Model Predictions are used to predict the probabilities of each class for the test set. The result is an array where each line represents a test sample. Convection of probabilities into class labels, the Argmax method is used to convert the predicted probabilities into class labels, by extracting the column index with the highest probability, which represents the predicted class.

With One-Hot, the conversion was made to a categorical representation for training the model, which was reverted to class labels to facilitate comparison with predictions.

Calculating various model performance metrics such as precision, recall, and f1-score for each class, as well as weighted average precision, recall, and f1-score based on support (number of true instances for each class).

Metrics by Class obtained, for Class 0 precision was 60.5%, Recall: 90.0%, and F1-Score was 72.4%, a measure that combines precision and recall, providing a balance between the two for class 0, 80 cases with the number of true occurrences of class 0 in the tested data set.

For class 1, precision: was 65.6% of cases than the previous model class 1, precision was 65.6%, Recall: was 27.6%, and F1-Score was 38.9% Indicating a relatively low in both precision and recall for class 1.

For Class 2 the precision found was 75.0%, Recall: 82.9% and F1-Score: 78.8% which show a good balance between precision and recall for class 2.

The model performs better in identifying class 0 and 2, with a good recall for class 0 and a good precision for class 2. Class 1, however, shows a significant gap between precision and recall, indicating an area of improvement.

The Figure below is the extraction and visualization of three critical evaluation metrics for classification models; precision, recall and F1 score. These metrics are extracted from a DataFrame called report\_df, which presumably contains the results of a classification report generated by machine learning model evaluation functions. Metrics are extracted for all classes except the last three entries, which may present total averages, which are not relevant for class-specific analysis. See figure 7.

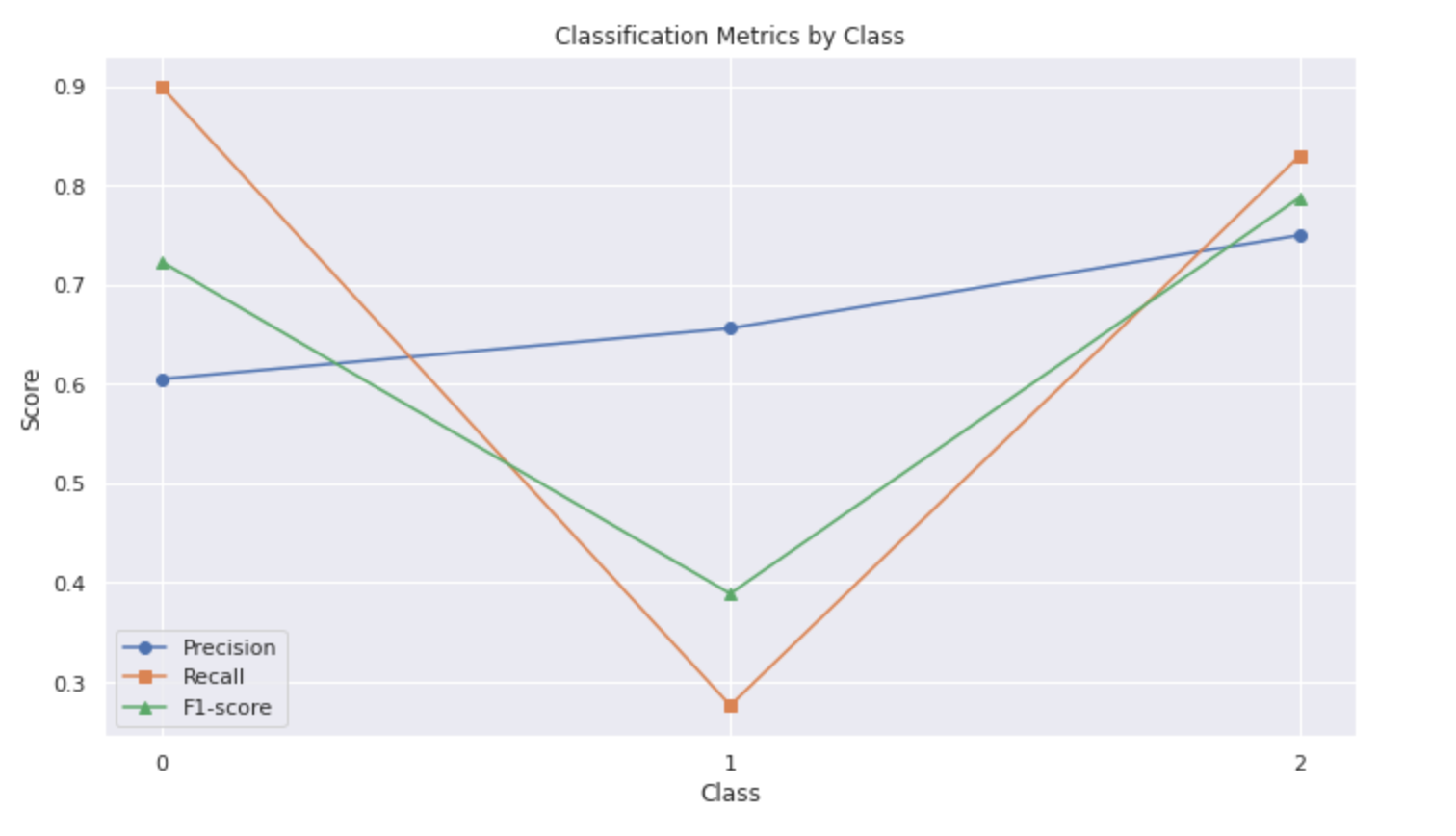
****

Figure Classification by Class

The Modeling Results for Learning Rating Adjustments evaluated the different learning rates that affect accuracy during the validation phase. The rates tested were 0.001, 0.01, and 0.1. The results indicated that, with a learning rate of 0.001, the validation accuracy was 55.8%. This lower rate generally results in slower convergence, potentially requiring more training epochs to achieve optimal accuracy.

Increasing the rate to 0.01, the validation accuracy improved significantly to 65.6%. This suggests that the model benefits from more significant adjustments to the weights, achieving faster convergence without exceeding the local minimum.

Finally, with a learning rate of 0.1, the validation accuracy reached 66.9%, the highest among the rates tested. This value indicates that the model was able to learn efficiently without the problem of "jumping" the optimal minimum, a common risk in high learning rates.

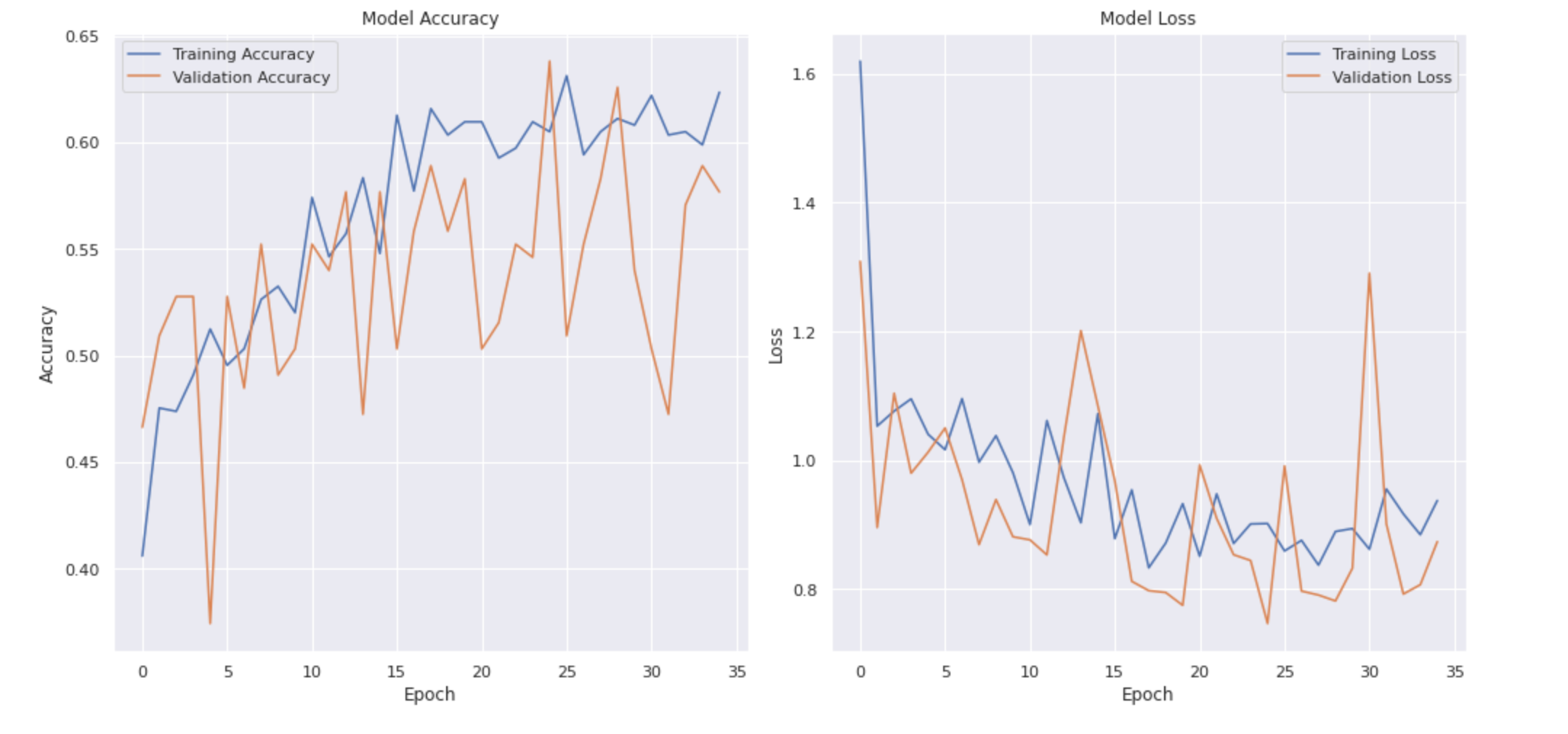
Optimizer Comparison shows model performance using three different optimizers to understand which one provides the best validation accuracy. The results for the Adam optimizer had a Validation accuracy of 39.88%, with Adam being the lowest among the tested optimizers. This may indicate that for this specific data set and network architecture, Adam may not be the most effective optimizer, or may need additional tuning of its hyperparameters.

RMSprop showed a validation accuracy of 47.85%, it had an intermediate performance, showing a higher validation accuracy than Adam, but lower than SGD.

The SGD (Stochastic Gradient Descent) presented a Validation accuracy: of 49.69% with the highest validation accuracy. This result suggests that, for this specific case, SGD may be doing better in exploring the parameter space or that it may be more stable with the type of data and model used.

Tuning the Neural Network Architecture, the model was built using a TensorFlow/Keras Sequential API, as input layers: contains 64 neurons with ReLU activation.   
Two hidden layers with 32 and 16 neurons, respectively, both with ReLU activation, and Dropout layers with a rate of 0.2 after the first two ReLU activation layers to reduce overfitting and in the Output layer: 3 neurons with softmax activation, suitable for three-class classification. Training was performed in mini-batches of 16 samples each, across 50 seasons, with a 20% validation split to monitor model performance on data not visible during training. Obtained results respectively are a Training Accuracy of 58.52%, Training Loss of 0.8969, Validation Accuracy of 50.92%, and Validation Loss: of 0.7469 A validation accuracy of approximately 51% suggests that the model has a moderate performance, but There are clear signs that it could be improved as the difference between training and validation accuracy indicates that the model may be starting to overfit the training data, despite the use of Dropout.

There is no model built for a sequential neural network with the following layers with 64 neurons, ReLU activation function, Two layers with 32 and 16 neurons respectively, both with ReLU activation, Dropout layers after the first two dense layers with rate 0.2, to reduce overfitting.The Output Layer has 3 neurons with softmax activation, suitable for three-class classification problems, the RMSprop optimizer with a learning rate of 0.1, and a metric used to monitor model performance is accuracy.   
Defined Callbacks, Early Stopping monitoring validation loss (val\_loss), and stopped training if there is no improvement after 10 consecutive seasons. The model is trained for up to 50 seasons with mini-batches of 16 samples. The dataset is divided into 80% for training and 20% for validation. Training uses the callbacks to monitor validation loss and stopped training prematurely with the Early Stopping callback, which in this case stopped training at epoch 35, restoring the weights from the best epoch (25). Training stopped at epoch 35 due to the lack of improvement in validation loss, demonstrating the effectiveness of the Early Stopping callback in preventing overfitting and the usefulness of the Model Checkpoint to capture the best version of the model during training. Even after detection with the start of the test, the test accuracy was 53.20%, which is much lower than the first ANN test, which obtained 65.72% accuracy.

***Accuracy and Loss Plot*** *Figure 8*

*The plot plots two graphs side by side to show model accuracy and loss over time.*

### **Conclusion**

### This study was designed to implement deep learning techniques for big data, exploring the tools available to deal with this context. A good practice was presented in the integration of hadoop with pyspark, extracting various important information and visualizations in SQL for data exploration. Many tools were used for different types of tests carried out, showing the possibility of dealing with the dataset.

### Regarding the applied neural networks RNN and ANN Model were observed the accuracy results of the models, the RNN model presented 0.592, and after using hyperparameters the accuracy was 0.604. So for ANN, the obtained accuracy was 0.657 and After careful evaluation with hyperparameter adjustment, learning rate adjustment, network architecture fine-tuning, batch size adjustment, and neural network training with callbacks the accuracy result was 0.532. It can be concluded that the first ANN model offers the best balance between accuracy and simplicity, so it makes sense to consider this. This creates opportunities for new studies, as the insights received reveal that there are spaces for improvement in data processing.

**References**

**[ 1] Bellazzi, R. (2014). Big data and complex biomedical data analytics: a challenge for bioinformatics research.**

**[2]** **White, T. (2012). Hadoop: The Definitive Guide. O'Reilly Media.**

**[3] Hashem, I.A.T., Yaqoob, I., Anuar, N.B., Mokhtar, S., Gani, A., & Khan, S.U. (2015).** The rise of “big data” on cloud computing: Review and open research issues. Information Systems, 47, 98-115.

**[4] Zhou, X., Chen, S., Liu, B., Zeng, X., & Zhao, H. (2014).** Harnessing the power of big data in health. Health Information Science and Systems, 2(3).

**[5]** **Mittal, S., & Goel, P. (2020). "Sentiment analysis using PySpark on Twitter data."** International Journal of Advanced Computer Science and Applications, 11(6), 373-377.

**[6] LeCun, Y., Bengio, Y., & Hinton, G. (2015). "Deep learning." Nature, 521(7553), 436-444 *.***

**[7]UCI Machine learning Repository, “30 March 2024. [Online] Available:** *https://archive.ics.uci.edu/dataset/863/maternal+health+risk*

[8] **Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).** [ONLINE] Available: https://papers.nips.cc/paper\_files/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html[Available ]