Sri Lanka Institute of Information Technology

Assignment 1 - Practical test

Y3.S1.WD.DS.0102
IT22577610 – Jayarathnage J.V.D.

Machine Learning and Optimization Methods

IT3071

Summary of Work

Objective

The goal was to use TensorFlow to create a deep learning model that would use different features to categorize student performance into three groups: High, Middle, and Low.

Data Preparation

- **import pandas as pd**: By doing this, the pandas library is imported and given the alias pd. Pandas is a robust data manipulation and analysis library that offers data structures such as DataFrames and Series.

- **import numpy as np**: *Numpy* is a Python library used for numerical calculations, mainly with arrays and matrices. This imports the library using the alias np.

- **from sklearn.model_selection import train_test_split**: The train_test_split function from sklearn.model_selection is imported in this way. A dataset can be divided into training and testing sets using this function.

- **from sklearn.preprocessing import LabelEncoder, StandardScaler**: Two preprocessing tools are imported in this way from *sklearn.preprocessing:*

- **LabelEncoder**: used to translate numerical labels from category labels.

- **StandardScaler**: used to scale to unit variance and remove the mean from features to standardize them.

- **import matplotlib.pyplot as plt**: Thus, matplotlib is *imported.pyplot* using the plt alias. Python visualizations that are static, interactive, and animated can be made with matplotlib.

- **import tensorflow as tf**: By using the alias tf, this imports the TensorFlow library. *Tensorflow* is an open-source library that offers tools for creating and refining neural networks in machine learning and deep learning.

- **pd.read_csv("academic_dataset.csv"):** This function call loads the contents of the CSV file "academic_dataset.csv" into a DataFrame by reading it. Pandas will take care of the file processing and DataFrame creation automatically. In this DataFrame, each row represents a record from the CSV file, and each column represents a field.

- The CSV file needs to be in the current working directory or I need to enter the correct path to the file for this to function correctly. I have the option of specifying the entire path or relative path to the file if it is located in a separate directory.

- Once loaded, I may use functions like **data.info()** to obtain an overview of the DataFrame's structure or **data.head()** to display the first few rows of the DataFrame.

- **data.isnull():** This function yields a DataFrame with the same shape as the input data, but it returns False otherwise and True when there are missing values (NaN values). , **.sum():** This function is used on the data.isnull()-created boolean DataFrame. I get the total number of missing values for each column by adding together the True values, which are considered as 1.

- **data.drop():** This function clears rows or columns of given labels. **['PlaceofBirth,'SectionID,'NationalITy,'Relation'**]: The column names I wish to remove from the DataFrame are listed here. axis=1 indicates that I wish to remove columns rather than rows. Axis=0 denotes rows, whereas axis=1 denotes columns. inplace=True: This means that the modifications will be done directly to the DataFrame data and no new DataFrame will be produced. If inplace=False, the original DataFrame would remain unaltered and a new DataFrame with the columns removed would be returned.

- **data.head():** The first five rows of the DataFrame data are shown using this method. It's a practical method to rapidly review the contents and confirm that the undesirable columns have been properly removed.

- To employ categorical columns in deep learning models, and convert them to numerical values. Construct a blank dictionary to hold encoders.
  To encode, enumerate the category columns. Use **LabelEncoder** to transform category data to integers for each column. These numbers should be used in place of the DataFrame's original values. For later usage, keep each encoder in the dictionary. My categorical columns will be in a numerical format and prepared for modelling once this code has been performed.

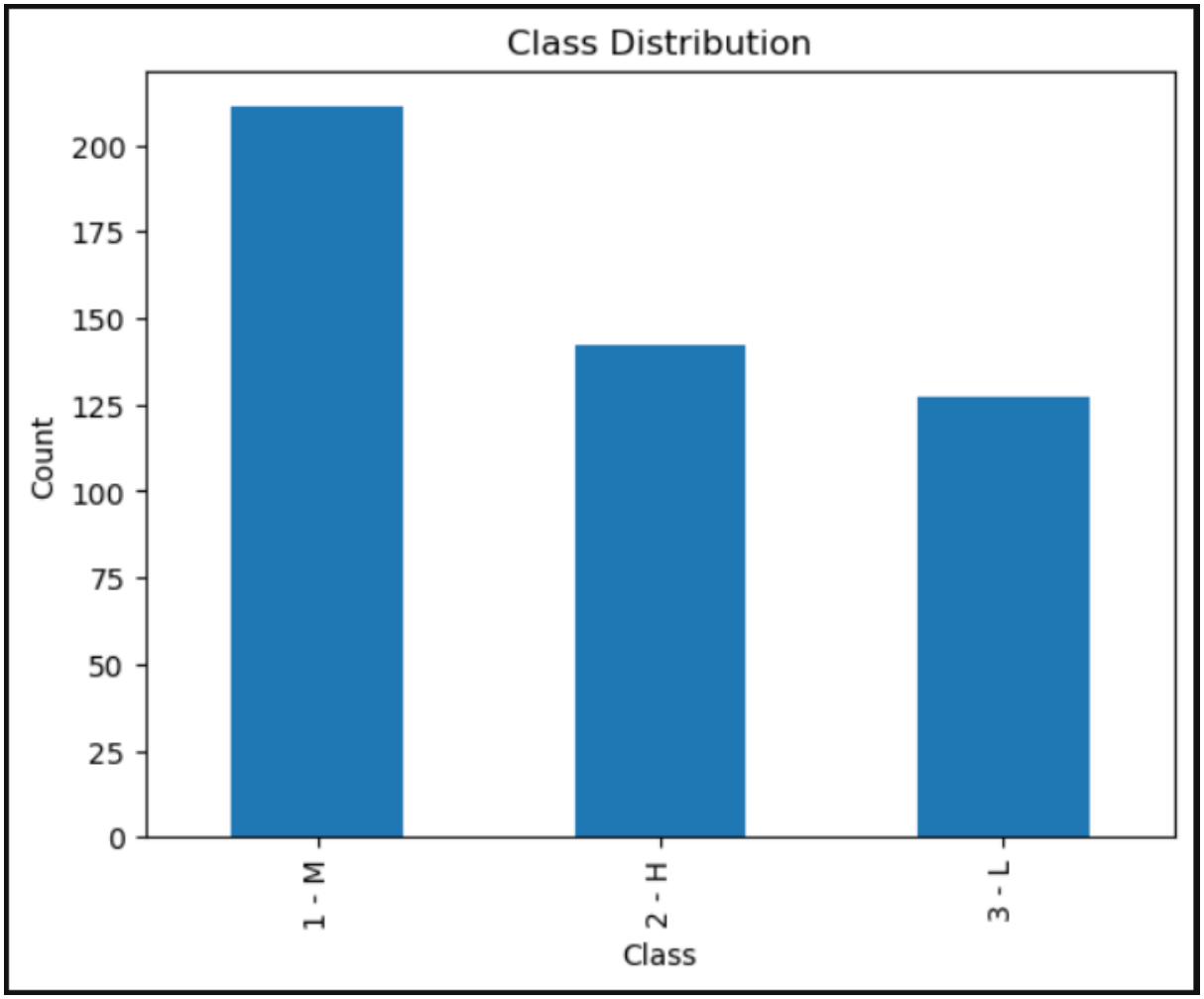
- The values "M," "H," and "L" in the Class column are changed to 0, 1, and 2, respectively, using this code. This is an additional method of transforming categorical data into numerical representation to prepare it for deep learning models.

- **Split Data**: *Features (X):* All columns except Class, *Target (y):* The Class column.

- My code performs feature scaling using *MinMaxScaler* from *sklearn.preprocessing*. Imports the MinMaxScaler class, which is used to scale features to a specified range (usually between 0 and 1). **x_scaled = scaler.fit_transform(X)**: Fits the scaler to the features in X and transforms them. This scales all feature values to the range [0, 1].

- The code validates the new data type of the target variable y after converting it to integer type. By doing this, I can make sure the target variable is prepared correctly for modelling.

- Visualize the use of a bar chart to illustrate the target classes' (M, H, and L) distribution. To improve readability, the x-axis labels are customized, and labels and a title are added to give context.



- Split the data into training (70%) and validation (30%) sets to train and evaluate the model.

Model Building

- **model = models.Sequential()**: Initializes a sequential model, where layers are stacked linearly.

- Create and train a neural network for multi-class classification utilizing a learning rate scheduler to modify the learning rate if the model reaches a plateau and early stopping to avoid overfitting.

- **BatchNormalization():** Normalizes the activations of the previous layer to stabilize and speed up training.

- **Dense(32, activation='relu'):** Adds another dense layer with 32 units and ReLU activation.

- **Dense(3, activation='softmax'):** *Output* layer with 3 units (for 3 classes), using the softmax activation to output probabilities for multi-class classification.

- **Training the Model**: ***model.fit(...)****:* Trains the model on the training data. ***epochs=100****:* Maximum number of training epochs. ***batch_size=32****:* Number of samples per batch during training.

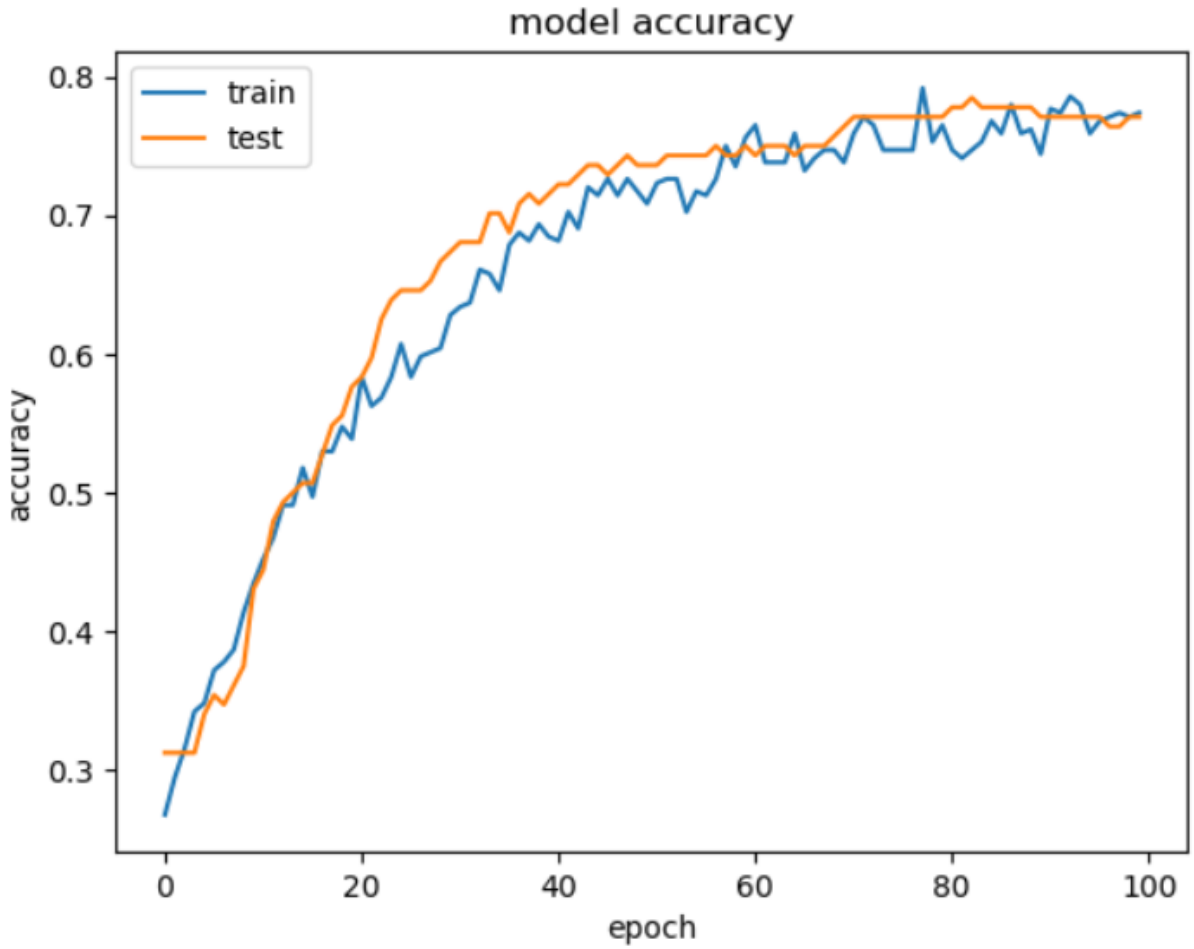
- If the validation loss doesn't get better, the model will terminate early and dynamically change the learning rate.

- **model.summary():** An extensive synopsis of the model architecture is given by this feature. It displays the number of parameters, the total number of trainable parameters in the model, the layers, and the output shape of each layer.

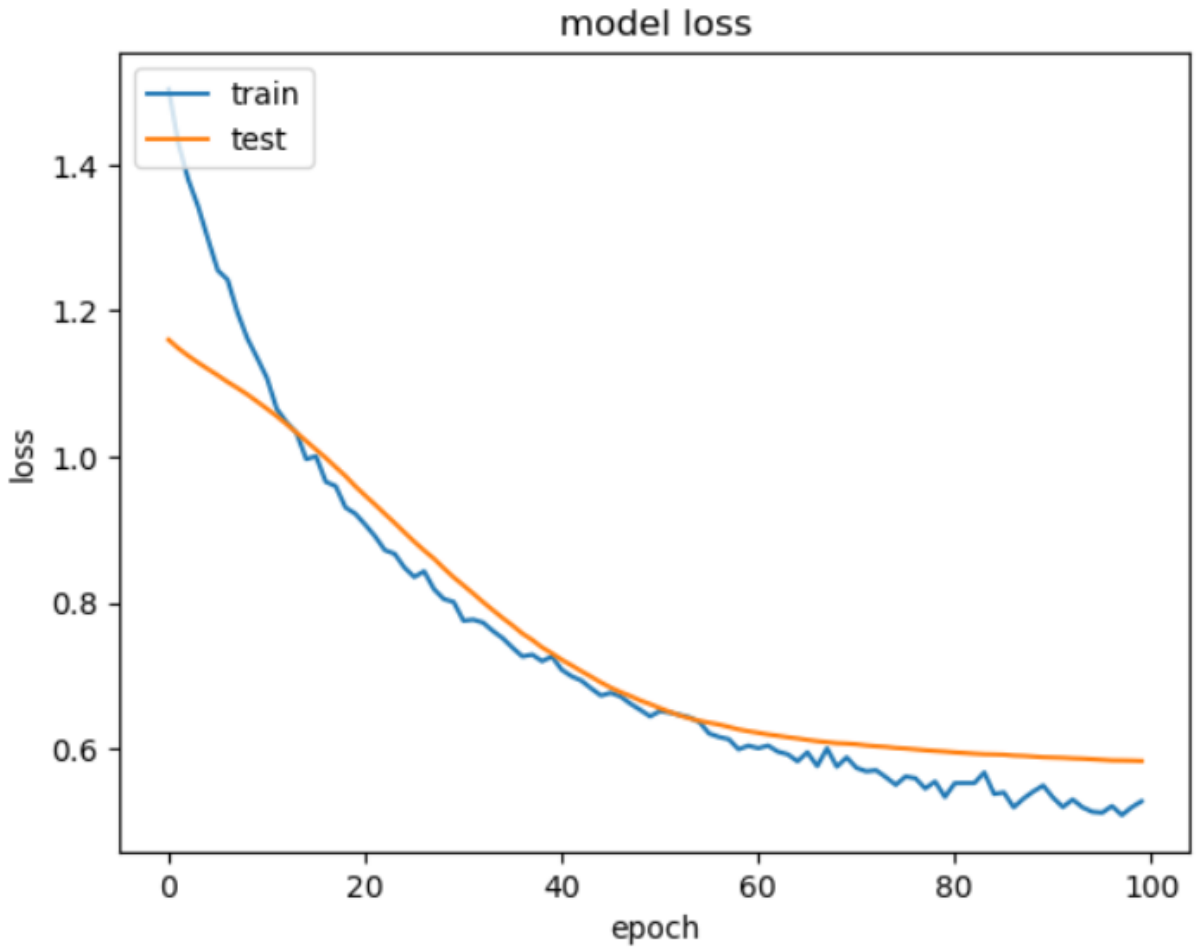
Model Evaluation

- I can evaluate how well I trained model performs on the validation set by using the **model.evaluate(X_val, y_val)** function.

- This code allows me to see how effectively the model is learning and generalizing by plotting the accuracy of training and validation across epochs. I'll show two lines that indicate training accuracy and validation accuracy, respectively, and these should help me identify any overfitting or underfitting problems.



- To monitor performance and determine whether the model is overfitting (validation loss increasing while training loss decreasing), visualize the model's training and validation losses. To assess how well the model is learning and generalizing, the plot will display two lines over the epochs: one for training and one for validation loss.



- Once the projected probabilities have been converted into actual class labels, review the initial predictions. I can now compare the model's predictions for the validation set with the true labels (y_val) because they are now expressed in terms of class labels (0, 1, or 2).

- Examine the initial predictions once the projected probability has been translated into actual class labels. Now that the model's predictions for the validation set are given in terms of class labels (0, 1, or 2), I can compare them with the true labels (y_val).

- Analyse the model's performance using the validation set and report on specific metrics such as F1-score, precision, and recall. I obtained a thorough understanding of the I model's performance in several classes as well as overall, which aids in identifying its advantages and disadvantages.

Findings

- **Model Performance:** Provide the outcomes of the model's operation on the validation set. For instance, "For the high-level class, the model attained an accuracy of 77% and a precision of 76%."

- **Insights:** Talk about any conclusions I draw from the model and data. For example, "More neurons in the hidden layers improved the model's performance, suggesting that a deeper network recorded complicated patterns more successfully."

- **Impact:** Talk about how my findings affect comprehending or resolving the issue. In the case of educational interventions, for instance, "The high accuracy of the model demonstrates its effectiveness in predicting student performance."

Challenges Faced

- **Data Issues:** Describe any data-related difficulties I am having, including missing values or unbalanced classes. For instance, "Managing missing values was difficult; to solve this problem, I used mean imputation."

- **Model Training:** Bring up any challenges I had when training the model, including overfitting or problems with convergence. As an example, "I adjusted the dropout rate and implemented early stopping because the model initially overfitted the training data."

- **Conceptual Issues:** Talk about any difficulties I have comprehending or using theoretical topics. found it difficult to choose the ideal model parameters for optimum performance.

- **Technical Difficulties:** Add any technical problems I ran with, like TensorFlow faults or hardware constraints. For instance, "TensorFlow compatibility issues were encountered and resolved by updating the library."

Lessons Learned

- **Skills Acquired:** Emphasise the new abilities or information I learned from the task. As an illustration, "Learnt how to use TensorFlow to effectively preprocess tabular data and implement deep learning models."

- **Best Practices:** List any successful tactics or best practices that surfaced. "Deeply enhanced model performance was found by normalizing numerical features," for instance.

- **Future Improvements:** Consider the things I would do differently in my next projects. As an illustration, "In future work, I would consider additional feature engineering and explore more advanced hyperparameter tuning techniques."

- developed abilities in model optimization and data preprocessing. Expert data cleaning was one of the best approaches. Improved feature engineering will be the main emphasis of future developments.