LEARNING PROCESS

The model used to predict brake pad condition is the Feedforward Neural Network model. This model was trained by using backpropagation which is implicitly used in the codes. This model contemplates all features in the dataset which are worn, km, heat, z, and pr.

Before starting the process to train the model, we need to import the necessary libraries such as pandas to manipulate data, “numpy” for numerical computations, and TensorFlow to build and train the neural network. Next, we must load the dataset which is brake_Modifieds.csv and store the dataset in the pandas data frame. Then, we need to ensure that the data is ready to train by doing the pre-processing data. In this case, we need to remove the commas for data in the ‘km’ column and convert them to float data type.

After doing the data pre-processing, we split the data into training and testing data. For this project, we have approximately 80% data for training and 20% for testing. Then before we train the model, we need to normalize the feature to ensure that the input features have zero mean and unit variance thus this will help in preventing any single feature from dominating the learning process.

Next, we build the neural network model. This model consists of two dense layers which are dense layer 1 and output layer. We obtained dense layer 1 from the “model.add(Dense(16, activation=’relu’, input_shape=(X_train.shape[1],)))” statement. This dense layer has 16 units of neurons which means the model will learn 16 different patterns of the data. As every neural network must have an activation function, thus this layer used rectified linear unit (ReLu) to introduces non-linearity and helps the model to recognize complex relationships in the data.

In order to complete the neural network architecture, this model has one output layer which has been added using the “model.add(Dense(1, activation = ‘sigmoid’))” statement. This output layer has 1 unit neuron to represent the binary classification output either worn (1) or not worn (0). For the activation function, this layer used sigmoid function to compress the output between 1 and 0.

This model was compiled using optimizer and loss function. The optimizer used is Adaptive Moment Estimation (Adam) Optimization Technique to update the model's weight during the training process while the loss function used during compilation is "binary_crossentropy" which commonly being used to solve binary classification problems. Lastly, the performance of the training model was evaluated using accuracy.

This model is trained by using backpropagation algorithm used to compute the weights and biases of neural networks which will allow the optimization during the training. For the code, we can see that the model trained using "model.fit(X_train, y_train, epochs=50, batch_size=32, verbose=0)" statement which determined that the backpropagation has been handled by the 'fit' function under the TensorFlow framework. This function used computations to update the model's weight and biases on the calculated gradients. The running times for the training model is 50 epochs which means that the model will go through all the dataset 50 times and the batch size for the model is 32 which indicates that the model's weight is updated after 32 samples being processed.

After training the model, the model was tested in the testing data. The threshold for the testing data has been set for 0.5 to convert the probabilities of the testing data into binary prediction. Then the accuracy of the model's prediction is calculated using accuracy function under sklearn framework.

STEP-BY-STEP LEARNING AND ANALYSIS

There are multiple steps that we took in developing a predictive model to predict the status of brake pads. The model can identify whether the brake pad has been worn or not worn. The steps taken are as below:

1. Define the problem

Defining the problem involves stating the predictive model's goal: to anticipate the state of brake pads (used or not worn) based on specified parameters. It is critical to have a thorough knowledge of the problem, to ensure that the model creation and assessment process corresponds with the desired outcome.

2. Acquire and preprocess data

We acquired a dataset that includes relevant features such as kilometers driven, heat, and wear rate. The corresponding status of the brake pads whether it has been worn or not has also been obtained. However, for the wear rate, we generated the sample data using a rough model :

```
z = wear_rate = (0.003 * heat) + (0.004 * kilometers)-78
```

Through this, we plugged the value into the logistic probability function to get the status of the brake pads. If the pr > 50%, then worn is 1 and vice versa.

```
pr = 1 / (1 + e**-z)
```

Based on our coding below, this is everything that we did regarding the data. Firstly, we imported all the necessary libraries that will be used for data preprocessing and model evaluation. Next, we loaded the dataset into our system and printed the first few rows for initial exploration. Then, for data preprocessing, we removed the commas inside the ‘km’ column of our dataset and converted it into float data type. This is to ensure that the values are in a suitable format for analysis. Furthermore, we also split the data into two parts which are the training and testing data sets. The ‘worn’ column is removed from the DataFrame, thus assigning the other columns to be variable ‘x’. The ‘worn’ column will be set as the variable ‘y’. The testing set is set to be 20% of the entire dataset and a random state, 42 is used for reproducibility.

Coding :

```
In [2]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import accuracy_score
        import warnings
        warnings.filterwarnings('ignore')
        import tensorflow as tf
        from tensorflow.keras.models import Sequential
        from tensorflow.keras.layers import Dense
```

```
In [3]: # Load the dataset from the CSV file
        df = pd.read_csv("C:/Users/Asus/OneDrive - Universiti Teknikal Malaysia Melaka/SEM 4/NN/brake_modified1.csv")
```

```
In [4]: df.head()
```

```
In [5]: df['km'] = df['km'].str.replace(',', '').astype(float)
```

```
In [6]: # Split the data into features (X) and target variable (y)
        X = df.drop('worn', axis=1)
        y = df['worn']
```

```
In [7]: # Split the data into training and testing sets
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

3. Choose and train a model.

Then, we split the data into training and testing sets for model training and evaluation. The training set is used to educate the model patterns and correlations between characteristics and the target variable (brake pad condition).

We have selected TensorFlow framework to build the neural network consisting of input layer, dense layer, and output layer. As the classification is binary classification, thus the output layer used sigmoid function as the activation function to compress the output between 1 and 0.

The model optimizes its prediction performance by doing the iterative training process using backpropagation.

Based on our coding below, we have initialized the model as a feedforward neural network model. The normalization used to normalize the input features, thus there's no input feature that will dominate the learning process.The fit() function, is used to train the model using the training data set, to allow the model to learn patterns and relationships between features and target variables. The training and testing data will also be displayed to observe its values and structure such as the features and target variables used. The model.predict() function has been used by us to predict the target variable for testing data. The predicted values will then be stored in y_pred.

Coding :

In [8]:
```
# Normalize the input features
X_train = (X_train - X_train.mean()) / X_train.std()
X_test = (X_test - X_test.mean()) / X_test.std()
```

In [9]:
```
# Build the ANN model
model = Sequential()
model.add(Dense(16, activation='relu', input_shape=(X_train.shape[1],)))
model.add(Dense(1, activation='sigmoid'))
```

In [10]:
```
# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

In [11]:
```
# Train the model
model.fit(X_train, y_train, epochs=50, batch_size=32, verbose=0)
```

In [12]:
```
# Print training data
print("Training Data:")
print(X_train)
print(y_train)
```

In [13]:
```
# Print testing data
print("Testing Data:")
print(X_test)
print(y_test)
```

In [14]:
```
# Test the model and print prediction output
y_pred_prob = model.predict(X_test)
y_pred = (y_pred_prob > 0.5).astype(int).flatten()
```

In [15]:
```
# Print the predicted output
print("Prediction Output:")
print(y_pred)
```

4. Evaluate model

To evaluate the model's performance, we calculated the accuracy of the model. The closer the accuracy to the value of 1, the higher the performance that this model has. The accuracy is calculated by comparing the predicted values to the actual values.

Based on our coding below, we did find the accuracy of the model by comparing the predicted values that we gained using the testing dataset to the actual values. This is done to evaluate the performance of this model.

Coding :

```
In [16]: # Calculate accuracy
         accuracy = accuracy_score(y_test, y_pred)
         print("Accuracy:", accuracy)
```

ANALYSIS AND INTERPRETATION

In this project, we used Python language to develop the predicting model to predict the status of the brake pads whether it is worn or not worn. There are 6 steps of how the analysis was conducted.

1. Data preparation

We read the dataset by using ‘pd.read_csv()’ and thus stored it in a Pandas DataFrame named ‘df’. We removed the commas in the ‘km’ column of the DataFrame and converted the values to float. This is to ensure that the data is in a suitable format for analysis.

2. Data splitting

We split the data into features(X), and target variable (y).The feature for X is km, heat,z and pr while the feature for Y is worn. Next, the data is split into training and testing datasets. 80% for training data and 20% for the testing dataset.

Training Data :

```
Training Data:
           km      heat         z        pr
60  -0.541283 -0.005907 -0.257446  0.627866
227 -0.025667 -1.529785 -0.956495 -1.059136
322  1.732142  1.732497 -1.252211  0.951522
318  0.877414 -0.856615  1.989296  0.951522
17  -0.129118 -0.005907  0.089717  0.951522
..        ...       ...       ...       ...
188 -0.887641 -0.139061 -1.165229 -1.059136
71  -0.295187 -0.242626 -0.202937  0.944396
106 -0.129620  0.216017  0.119258  0.951522
270  0.098475 -0.960180 -0.330665 -1.059136
102  0.111965 -0.301805  0.482051  0.951522

[268 rows x 4 columns]
60     1
227    0
322    1
318    1
17     1
      ..
188    0
71     1
106    1
270    0
102    1
Name: worn, Length: 268, dtype: int64
```

Testing Data :

```
Testing Data:
           km      heat         z        pr
72  -0.218227 -0.076114 -0.081032  0.876205
110  1.613307 -0.814459  2.274494  0.926464
298 -1.164408  2.592166  0.492264 -1.083889
108 -0.218838 -0.083424 -0.079792  0.802218
277 -0.982598 -1.786737  0.962267  0.926464
..        ...       ...       ...       ...
299 -0.849890  0.442921  1.259276  0.926464
295 -0.764023 -1.238460 -1.075627 -1.083889
328  1.566160  1.130095 -1.163767 -1.083889
84  -0.218838 -0.003010 -0.084043  0.802218
245  1.069404  0.925405 -0.991560 -1.083889

[68 rows x 4 columns]
72     1
110    1
298    0
108    1
277    1
      ..
299    1
295    0
328    0
84     1
245    0
Name: worn, Length: 68, dtype: int64
```

3. Model initialization and training

We initialized the feedforward model using “model = Sequential()” and trained the model on training data using “model.fit(X_train, y_train, epochs=50, batch_size=32, verbose=0)”. This step allows the model to learn the patterns and relationships between features and target variables.

4. Data Exploration

Training data consisting (X_train) and corresponding target variable (y_train) is printed to observe values and structure. This is to gain insights into the distribution and relationships between the features and target variables.

```
Training Data:
           km      heat         z        pr
60  -0.541283 -0.005907 -0.257446  0.627866
227 -0.025667 -1.529785 -0.956495 -1.059136
322  1.732142  1.732497 -1.252211  0.951522
318  0.877414 -0.856615  1.989296  0.951522
17  -0.129118 -0.005907  0.089717  0.951522
..        ...       ...       ...       ...
188 -0.887641 -0.139061 -1.165229 -1.059136
71  -0.295187 -0.242626 -0.202937  0.944396
106 -0.129620  0.216017  0.119258  0.951522
270  0.098475 -0.960180 -0.330665 -1.059136
102  0.111965 -0.301805  0.482051  0.951522

[268 rows x 4 columns]
60     1
227    0
322    1
318    1
17     1
      ..
188    0
71     1
106    1
270    0
102    1
Name: worn, Length: 268, dtype: int64
```

5. Testing and prediction

The testing data including the features (X_test) and the target variables (y_test) is printed to examine the values and structure of the testing set. Then we predict the target variable for testing data using ‘model.predict(X_test)’ via the trained model.

Testing Data :

```
Testing Data:
           km      heat         z        pr
72  -0.218227 -0.076114 -0.081032  0.876205
110  1.613307 -0.814459  2.274494  0.926464
298 -1.164408  2.592166  0.492264 -1.083889
108 -0.218838 -0.083424 -0.079792  0.802218
277 -0.982598 -1.786737  0.962267  0.926464
..        ...       ...       ...       ...
299 -0.849890  0.442921  1.259276  0.926464
295 -0.764023 -1.238460 -1.075627 -1.083889
328  1.566160  1.130095 -1.163767 -1.083889
84  -0.218838 -0.003010 -0.084043  0.802218
245  1.069404  0.925405 -0.991560 -1.083889

[68 rows x 4 columns]
72     1
110    1
298    0
108    1
277    1
      ..
299    1
295    0
328    0
84     1
245    0
Name: worn, Length: 68, dtype: int64
```

Prediction :

```
Prediction Output:
[0 0 0 0 1 1 1 1 1 1 1 0 1 1 0 1 0 0 1 0 1 0 0 0 1 1 0 1 0 1 0 1 1 0 0 0 1
 0 0 0 1 1 0 1 0 0 0 0 1 1 1 0 0 1 0 1 0 1 0 1 1 1 1 1 0 0 0 0 0]
```

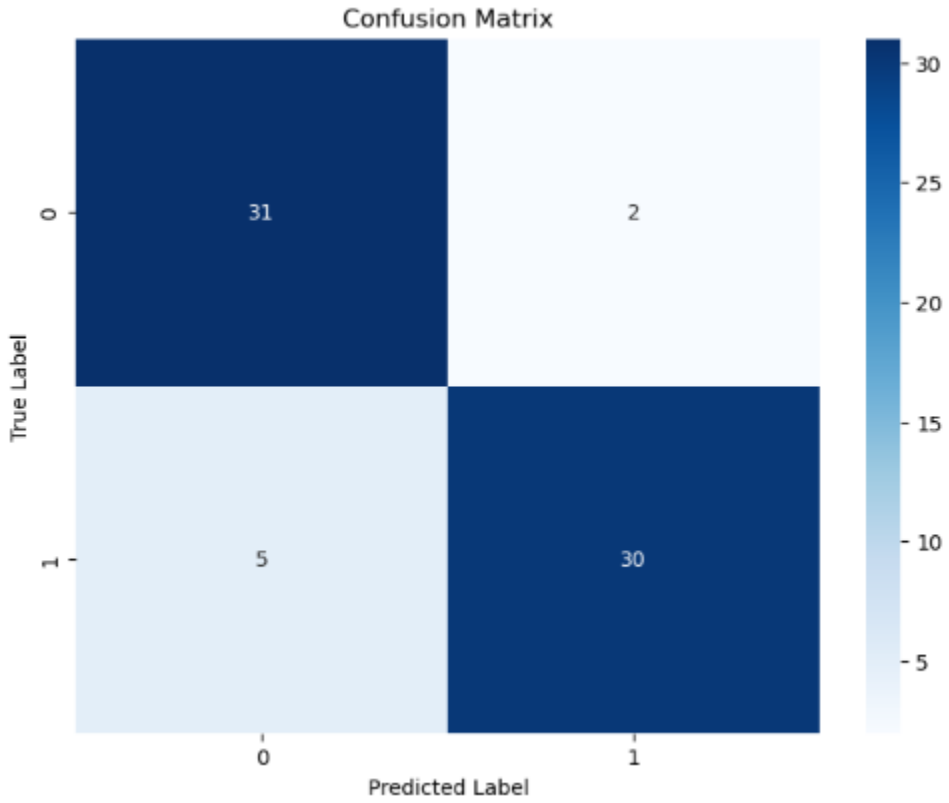
6. Model evaluation

To evaluate our model, we calculated its accuracy using the 'accuracy_score(y_test,y_pred)'. Based on the output, the accuracy of our model is 0.8970588235294118.
This is by comparing the predicted values of ('y_pred') and the actual values ('y_test'). Then the output has been visualized into confusion matrix to get the summary of the model's performance. The summary of the performance calculated and summarized in classification report. The report consists of precision, recall(sensitivity), f1-score and support. This report helps us to see the overall model's performance by giving insights of how the model distinguishes the positive and negative classes.

Output:

```
Accuracy: 0.8970588235294118
```

Confusion matrix:



Classification report:

```
Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.94      0.90        33
           1       0.94      0.86      0.90        35

    accuracy                           0.90        68
   macro avg       0.90      0.90      0.90        68
weighted avg       0.90      0.90      0.90        68
```