Project 2

==========

Group 4 :
Hariom Meena - 190020017
Ritwik Singh - 180010028
Suryavijoy Kar - 190020039

Problem statement:
The dataset consists of 3 types of entities:
(a) the specification of an auto in terms of various characteristics
(b) its assigned insurance risk rating
(c) its normalized losses in use as compared to other car
There were 204 examples in the dataset, each having 26 attributes. We have to develop a model to predict the ‘symboling’ feature from all the other attributes in the dataset. Cars are initially assigned a risk factor symbol associated with its price. Then, if it is more risky (or less), this symbol is adjusted by moving it up (or down) the scale. Actuarians call this process "symboling". A value of +3 indicates that the auto is risky, -3 that it is probably pretty safe.

Github Link -
https://github.com/notwarnite/MFDS_Project/blob/main/Grp-4_Project_2/Grp_4_project_2.ipynb

Feature engineering:
The dataset provided is sparse with some entries having no values. All such values have to be engineered to better fit the model.
For the continuous features, we replace the missing data-feature with the mean calculated from the non sparse data-feature.
For the nominal features, we replace the missing data-feature with the mode calculated from the non sparse data-feature.
For the nominal datasets, we also encoded the nominal values to integers using the LabelEncoder function from the sklearn python library.
The target feature ‘symboling’ has value as ‘-2, -1, 0, 1, 2, 3’. So we have to do Multi-Class Classification. We added 2 to all the values to shift it to ‘0, 1, 2, 3, 4, 5’. We then OneHotEncoded the values using keras to_categorical function to form a new target-feature vector. We use this new feature in our training.

Base Model:
We made a 2 layer Neural-Network with relu activations in the hidden layers and softmax for the Output layer using tensorflow and keras libraries.
For pre-processing, we scale our features using the StandardScaler function to better fit the model. We use the train-test-split library function to split the data into training and cross validation.

We use Adam optimizer with a learning rate of 0.0005 and the loss function is Categorical Cross Entropy. The model was run for 150 epochs (epochs is the number of times we loop over the entire dataset) and the entire batch was used as the mini-batch.

Model without regularization:
The base model is used as a model without regularization. After running for 150 epochs, we get the training accuracy as 94.48% and validation accuracy as 78.05%. We get a training loss of 0.3959 and a validation loss of 0.7805. We see that the model without regularization overfits the dataset.
On predicting on the new data_point, we get the insurance risk rating (symboling) for the given data_point as -1.

Model with regularization:
The model with regularization is similar to the base model except that we have added L1 regularization in the hidden layers using tensorflow keras regularizers with a regularization parameter as 0.0005. After running for 150 epochs, we get the training accuracy as 78.53% and validation accuracy as 73.17%. We get a training loss of 0.6596 and a validation loss of 0.7317. We see that the model without regularization does not overfit the dataset. So adding a regularization helped in removing the problem of overfitting.
On predicting on the new data_point, we get the insurance risk rating (symboling) for the given data_point as -1.

Loss Function:
The loss function chosen by us is Categorical CrossEntropy. Categorical cross entropy is a loss function that is usually used in multi-class classification tasks. These are tasks where an example can only belong to one out of many possible categories (in our case symboling), and the model must decide which one. Formally, it is designed to quantify the difference between two probability distributions. Using this we can classify new data points depending on which probability is highest.
As our problem was a multi-class classification, we chose Categorical CrossEntropy as our loss function.

Contributions:
All the 3 members discussed the problem and tried out various models. The feature engineering and data cleaning part of the code was done by Hariom Meena and Ritwik Singh. The Neural Network modelling part was done by Ritwik Singh and Suryavijoy Kar. For the finetuning and hyper-parameter optimisation, all 3 members carried out testing and discussions.