MACHINE LEARNING ASSIGNMENT

GUMMA SRI SOUGANDHIKA

121AD0020

## **Problem:**

A model was to be prepared for the PEMS dataset of traffic flow provided by Caltrans official website. It captures the data from various districts and puts it together in dataset format.

I have downloaded one such dataset from the website and I have predicted the speed value from the data given.

**Model Training Data taken from:**

District 8: San Bernardino/Riverside

ID: VDS 801259

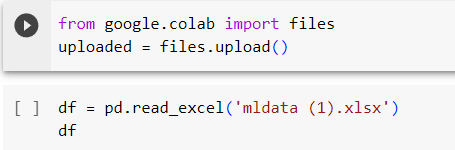
Timeline: 01/01/2018 to 01/01/23 or 5 years of data taken

**Model Forecast taken from:**

Same details, with timeline: 01/02/2023 to 28/02/23 or the February month

## **Dataset Importing:**

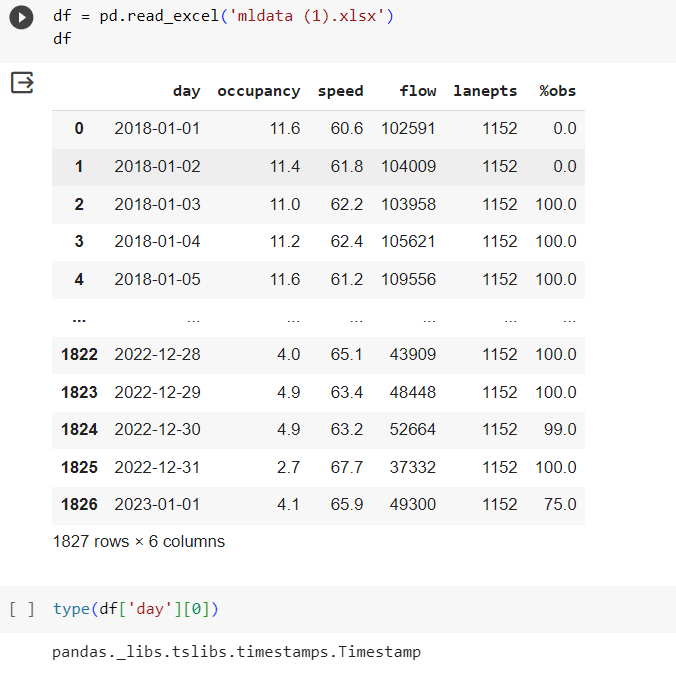
The data is imported from the computer into a pandas data frame.

­­

## **Preprocessing Data:**

The data is pre-processed by removing unnecessary columns and null value columns by using the drop statements.

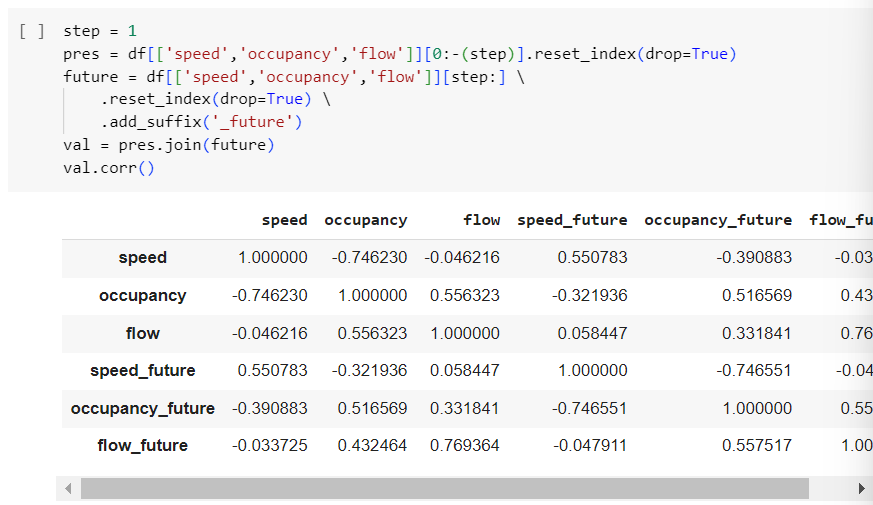
Later, we have also standardised the data in the xtrain set.



## **Checking correlation among the quantities:**

This helps us decide what to use for the prediction, which quantities are best related.

Here we have chosen to predict the occupancy using the speed and flow values as their correlation is quite good.



## **Splitting the dataset into test and train sets and setting the dimensions:**

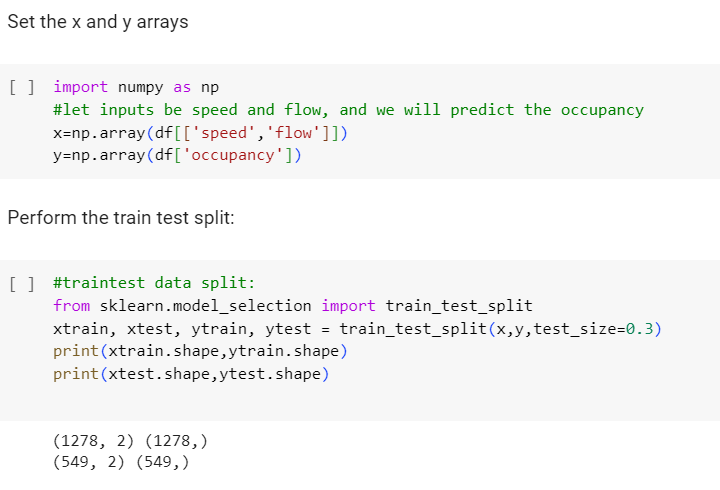
Here we have split the data using the scikit learn module and split the dataset.

We have mentioned that the test size is 0.3 which means that 30% is test set and 70% is the train set.

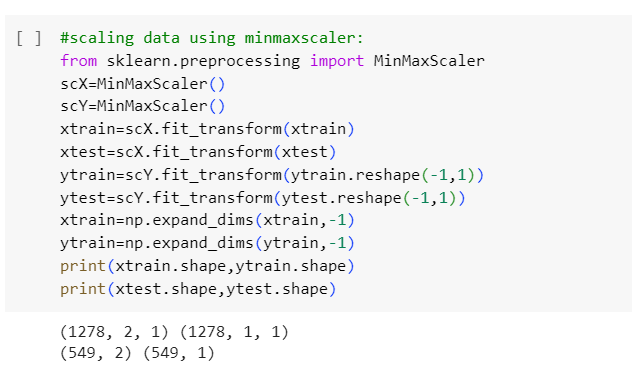
After that we have used the sklearn preprocessing method of MinMaxScaler.

Finally, we have set the dimensions to the NumPy arrays of xtrain, xtest because the input taken by the model is of specific dimensions.

After expansion by using the function expand\_dims, we can see the final output shape.

****

## **Scale the data:**

****

## **Building the model using the functional API:**

In my model, I have used the simple model from all, which is the SimpleRNN. SimpleRNN model is a recurrent neural network (RNN) that is composed of a single recurrent layer. RNNs are a type of neural network that are well-suited for processing sequential data, such as text, time series data, and audio data.

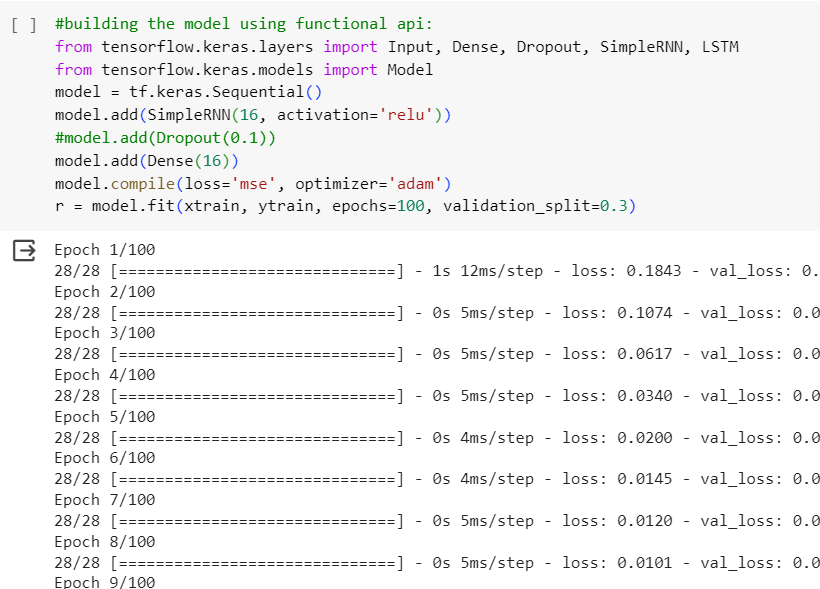
After inputting the SimpleRNN, we have used a dense layer. The Dense layer is a fully connected layer in TensorFlow. It is the most common type of layer used in neural networks. The Dense layer performs a matrix-vector multiplication between the input and the weights of the layer. The output of the layer is then passed through an activation function, ReLU here.

## **Compile and fit the model (Training the model):**

Here we have used the **compile** and **fit** methods.

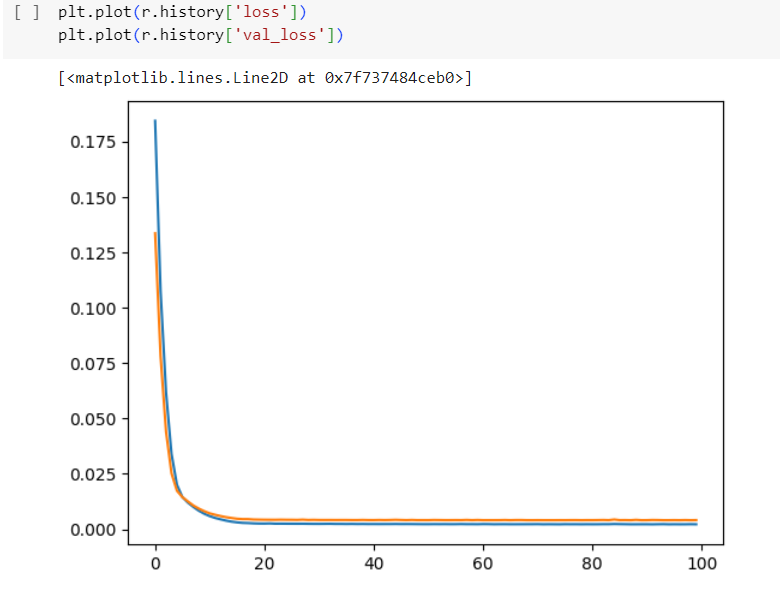
Before calling in these functions we have pre-defined the root mean square error function. In this way we are calculating the RMSE metrics.

After that we are fitting the model using the xtrain and ytrain data. This is our training part. In this, we see epochs=100, which means I am running my iterations for 100 times until the data is fit. This can be tuned later on.



## **Plotting Loss:**

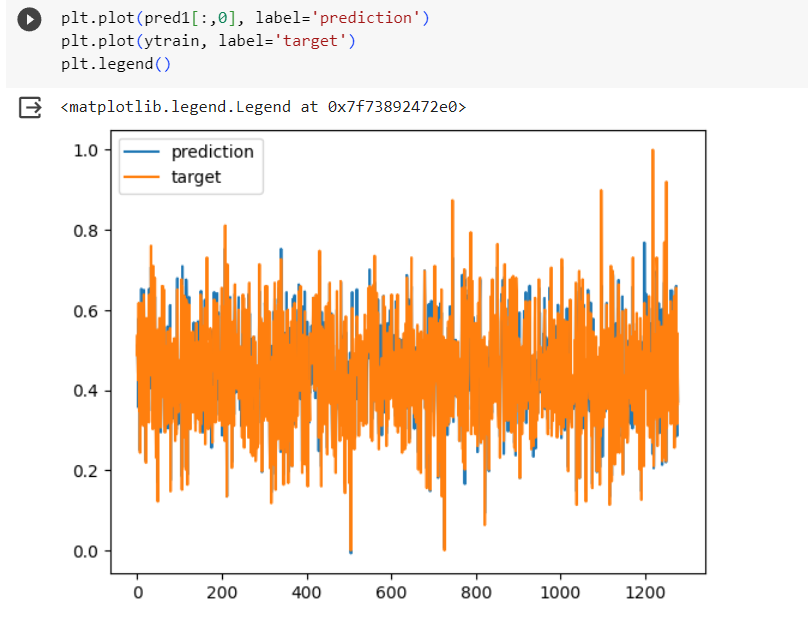
After the epochs complete execution, we can see the loss plot using matplotlib module.



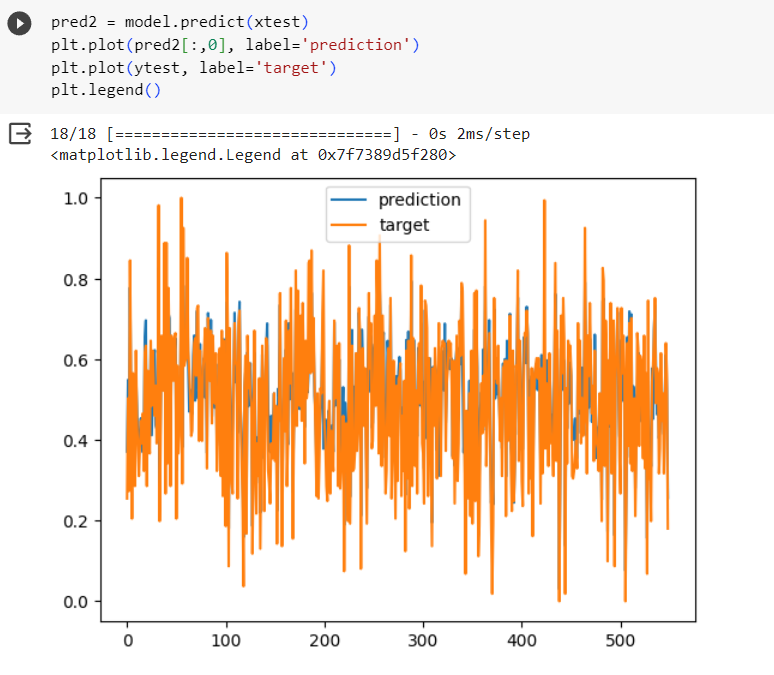
Clearly the loss has decreased nicely over the epochs meaning the model has surely learned a few things.

## **Prediction for train set:**

Here we have predicted for ytrain, and we have plotted the original ytrain. As we can see that the model is good at predicting the values.

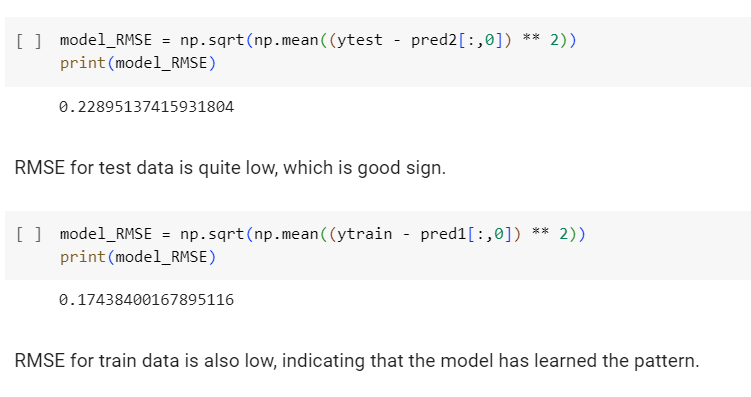


On the other hand, predictions with ytest, predictions are as follows:



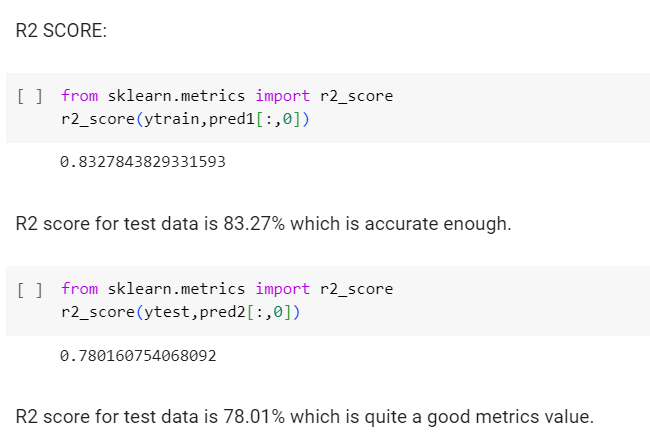
## **RMSE Metrics:**

We have seen the rmse for the ytest and ytrain predictions:



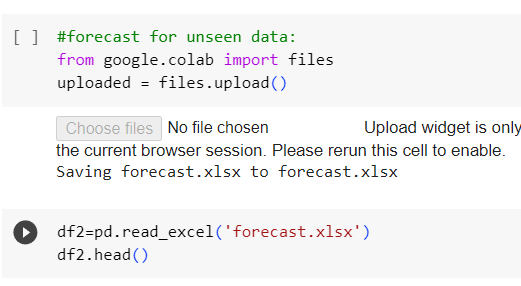
## **R2 metrics:**

Similarly we also checked for the r2 score:

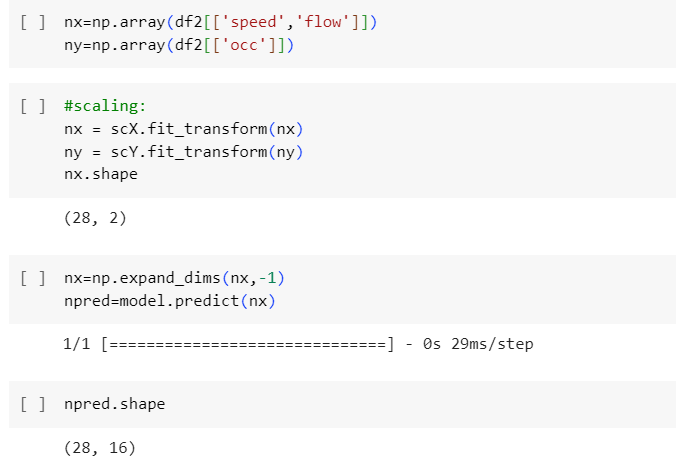


## **Unseen Data Prediction (for February 2023 data):**

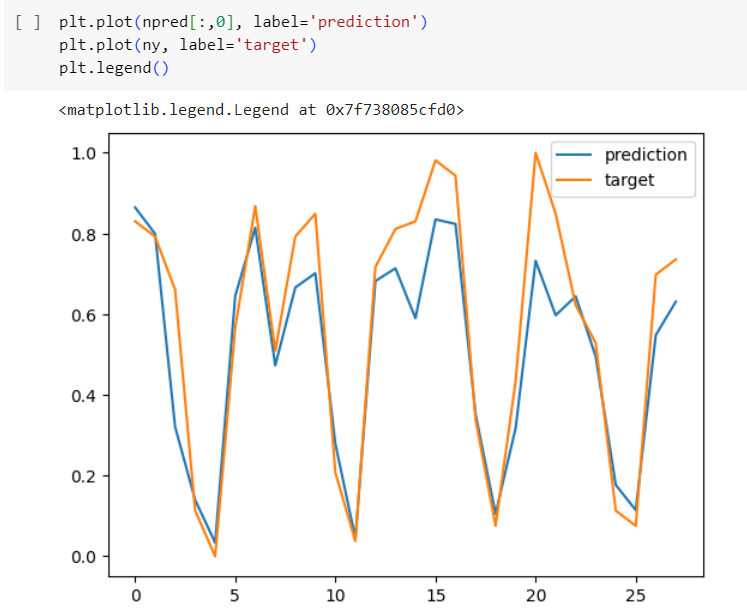
We have used the February month data to see if our model works for the unseen new data:



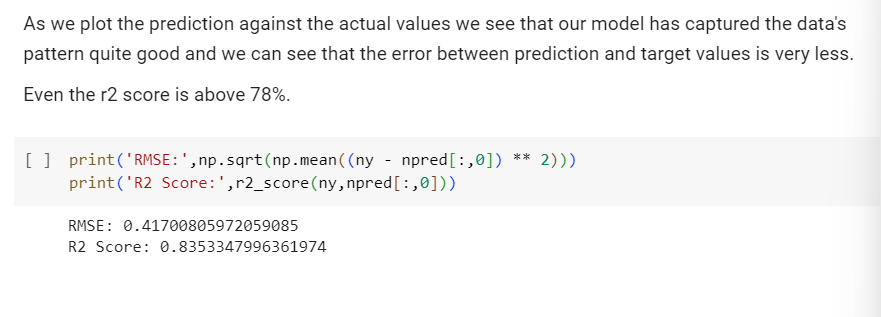
Again, the same steps. We standardise the data using MinMaxScaler and then predict the output using our model.



## **Plotting the target and prediction:**



## **R2 score metrics and RMSE metrics for unseen data:**



## **Conclusion:**

In conclusion, the SimpleRNN model was able to achieve satisfactory results on the given task. The model was able to learn the long-range dependencies in the data and make accurate predictions. The SimpleRNN model is a relatively simple type of RNN.

We can see that the test set predictions are quite good, for one step forecast method.