**Documentation for train\_station\_passenger.ipynb file**

This code is based on the tutorial of DigitalSreeni on YouTube so please watch his video <https://www.youtube.com/watch?v=97bZKO6cJfg&t=728s> for more information.

**Software used**

The code is inside a jupyter notebook file. I used VSCode to open and edit the file easily. Python version 3.8.10 is used in this code.

**Installing dependencies**

Before running this code you need to pip install libraries that will be used to process, plot, and train the data and model. You can use a python virtual environment or your global python interpreter.

List of libraries to install:

* Numpy
* Tensorflow
* Pandas
* Matplotlib
* sklearn

**Importing the dependencies**

This part of the code just imports the libraries that you installed. It is usually imported as an acronym like “tensorflow as tf” or “pandas as pd” for easier typing and a much cleaner code. It is also practiced by most of the developers that uses these libraries.

Text

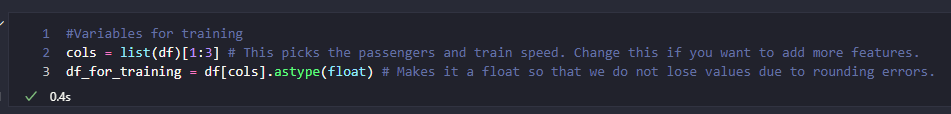
Description automatically generated

**Importing the data**

This is where the pandas library will be used because it is needed for reading the data which is in .csv file type.

Text

Description automatically generated



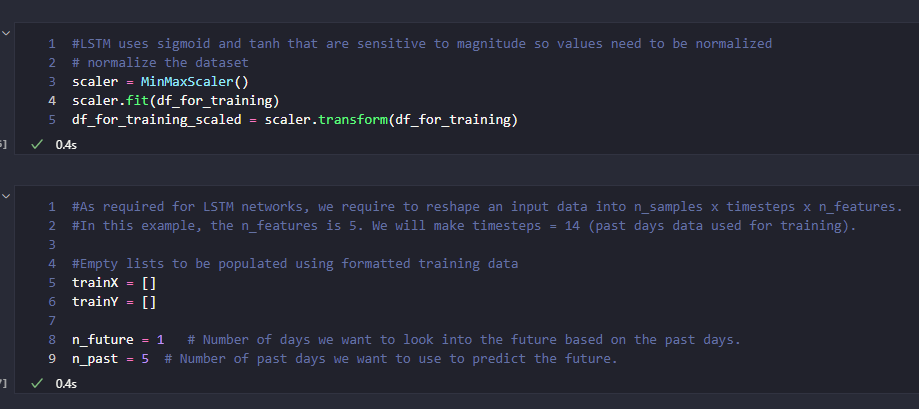
Table

Description automatically generated with medium confidenceData is saved in a variable as a dataframe type

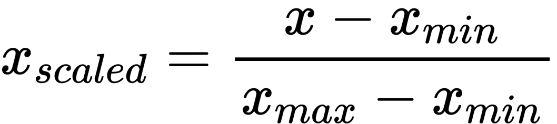
Graphical user interface

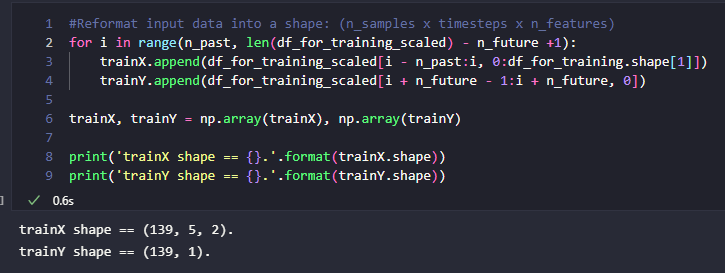
Description automatically generatedPlotting the data to visualize it

**Data preprocessing**



Before we input the data we need to scale it first because the activation function used in the LSTM model is very sensitive to magnitudes. In this type of data we will be using the MinMax scaler which has the formula:

 After scaling the data we need to create an X data which will be the input and a Y data which will be the true output and it will be used to calculate the loss. trainX will have a shape of (n\_samples, n\_past, n\_features). “n\_samples” is the number of sample data that you will be using to train the model. “n\_past” is how long should the LSTM model will look in the past data to predict the future. “n\_features” is just the number of variables in the input, in this case n\_features = 2 because the input variables are passengers and train speed.

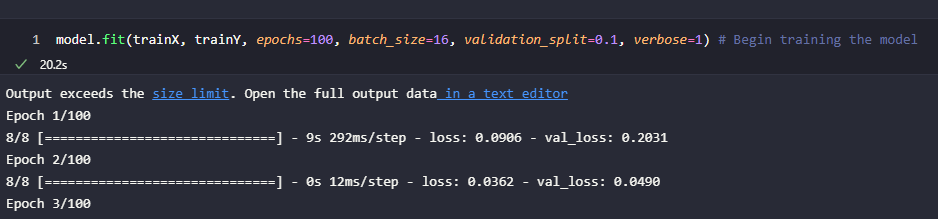


Text

Description automatically generated**Making the LSTM model**

Before adding layers, tensorflow has this tf.keras.Sequential() class that you will need to make an object of because all of the neural networks will be placed in order inside the Sequential object. For more information, please read the tensorflow documentation online. After making a Sequential object we need to add the layers. The first layer is the input LSTM layer and in this layer we need to specify the input\_shape which is in this case is (trainX.shape[1], trainX.shape[2]) or in other term (n\_past, n\_features). After that you can add additional layers by changing the n\_of\_lstm\_middle\_layers variable. After that we need to place the last LSTM layer. As you can see here there is a “return\_sequences” parameter that is equal to True in each of the LSTM layer except for the last layer. If we wanted to add multiple LSTM layers we need to set the return\_sequences = True so that each time step the LSTM layer will output the hidden state and at the very last LSTM layer we just wanted the last hidden state of the last time step so we set it to return\_sequences = False. Please note that it is possible to just have a single LSTM layer, you will need to just comment out the for loop code and the last LSTM layer and set the first LSTM layer return\_sequences parameter to False. The number 64 in each of the LSTM layer is the size of the hidden state and cell state of the LSTM layer. The bigger this number the more complex your model becomes and the more the weights and biases in each of the gate inside the LSTM cell will have.

The very last layer in this Sequential model is a dense layer. It is just a neural network layer consisting of only one neuron that takes the output of the hidden state in the last time step of the LSTM layer before it.

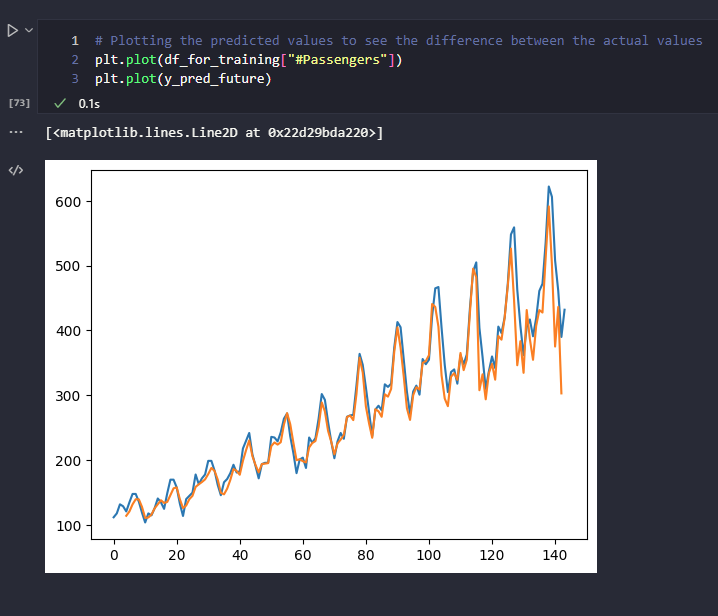
**Training the model**

This part of the code trains the whole model of LSTM. It needs an input data and the true values of the output data to calculate the loss. “Epochs” means that how many times are we going to use the trainX and trainY data to train the model. “batch\_size” means that how many samples do we need to input before backpropagation begins. “validation\_split” means that it divides the data and takes 10% off of the train data and makes it as a validation data. As you can see in the rightmost text there is a “val\_loss” there which means the loss value if we input the validation data. The validation data is never used for training. “verbose” just means what do you want to see in the output while training.

Text

Description automatically generated**Testing the model**

We tested the newly trained model using the trainX data again and see if it is properly predicting the values. After that we rescale(inverse\_transform in the code) the data so that we can plot it with the original data.



The blue line is the original data and the orange line is the LSTM model prediction.

Graphical user interface, text

Description automatically generated**Saving the model**

To save the model we just need to uncomment this part of the code and specify inside the function which location we want to save the model. For loading a model we just need to run the code tf.keras.model.load\_model(<model folder location>) and it will return the model in the specified location.

**References**

LSTM youtube tutorial by DigitalSreeni - <https://www.youtube.com/watch?v=97bZKO6cJfg&t=728s>

Tensorflow guide on neural networks - <https://www.tensorflow.org/tutorials/quickstart/beginner>