ADVANCED MACHINE LEARNING

SAI PRASAD BOYAPATI

TIME SERIES DATA SUMMARY REPORT

I began by downloading a dataset named 'jena_climate_2009_2016.csv.zip' and extracting its files. The next step involved parsing the data to prepare it for analysis. This dataset contains climate information, including features like temperature. I extracted both the header and the actual data records. Following that, I calculated the standard deviation of the data, which is essential for normalizing it.

To gain a clearer insight into temperature fluctuations, a time series plot is created to illustrate how temperatures vary over time.

Using predefined split percentages, it calculates the number of samples for the training, validation, and testing datasets.
To prepare and standardize the data, I normalized it by subtracting the mean and dividing by the standard deviation.
A dummy dataset is generated to demonstrate how to use the \`timeseries_dataset_from_array\` function. Datasets are then created for training, validation, and testing. Batch sizes and other relevant parameters are defined as follows:

```
num_train_samples: 210225
num_val_samples: 105112
num_test_samples: 105114
```

The code displays information about the sample shapes within the training dataset for review.

A reasonable baseline model is implemented to calculate the Mean Absolute Error (MAE) values for both the validation and test datasets, yielding the following results.

Following this, several machine learning models are examined, beginning with a connected model whose performance is evaluated. Next, a 1D convolutional model is assessed for its effectiveness.

Lastly a straightforward Long Short-Term Memory (LSTM) model is introduced to forecast time series data. The results obtained through this model is as follows

The provided code implements a Recurrent Neural Network (RNN) using NumPy to facilitate a better understanding of the internal workings of RNNs. It highlights different types of RNN layers in Keras, which can process sequences of varying lengths, return only the output step, or even provide the entire output sequence.

The results are as follows. Additionally, the code explores RNN models, examining techniques such as dropout to mitigate overfitting and the stacking of layers.

It then trains and evaluates a stacked Gated Recurrent Unit (GRU) model with dropout regularization applied. The following table presents the values obtained from the model analysis:

MODELS	TEST MAE
NAVIE METHOD	2.62
DENSLY CONNECTED NETWORK MODEL	2.62
1D CONVOLUTIONAL MODEL	3.12
SIMPLE LSTM MODEL	2.59
RNN MODEL	2.54
DROUPOUT LSTM MODEL	2.43
SIMPLE LSTM MODEL WITH 32 UNITS	2.51
STACKED LSTM WITH 64 UNITS	2.48
STACKED LSTM WITH 8 UNITS	2.78
1D CONVOLUTIONAL WITH RNN	3.7

In summary, this code offers a comprehensive exploration of time series data analysis and forecasting utilizing both machine learning and deep learning models. It begins with the essential steps of data preprocessing, which include cleaning, transforming, and preparing the dataset for analysis to ensure that the data is in a suitable format for modeling. This phase also encompasses normalization, a crucial process that adjusts the data values to a common scale without distorting differences in the ranges of values, thus improving the model's performance and convergence during training.

The code then progresses to model development, where various machine learning algorithms and deep learning architectures are implemented to capture the temporal patterns within the data. This includes experimenting with different types of models, such as Recurrent Neural Networks (RNNs) and Gated Recurrent Units (GRUs), which are particularly adept at handling sequential data. The exploration of model architecture also involves techniques like dropout for regularization, which helps prevent overfitting by randomly deactivating a fraction of the neurons during training.

Finally, the code emphasizes the evaluation of the developed models, employing metrics such as Mean Absolute Error (MAE) to assess their accuracy and effectiveness in forecasting future values. By systematically addressing each of these components—data preprocessing, normalization, model development, and evaluation—the code serves as a valuable resource for understanding and applying time series analysis techniques in practical scenarios.