Abstract

The full journey for the time series modelling assignment started with loading data from CSV files into Pandas DataFrame. Exploratory data analysis was performed on the whole dataset and various insights obtained from the exploratory data analysis were used for cleaning the data and performing feature engineering. After Feature engineering, data was split into train, validation and test sets followed by normalization. Next, Input and Output windows were created for the three sets. At last, a total of six different models were trained and evaluated. A comparative study was done on all six models' performance on the validation data. This documentation is further divided into 7 sections namely EDA, Feature Engineering, Data Split and Normalization, Window Creation, Model Training, Comparative Study and Submission

Exploratory Data Analysis:

Exploratory Data analysis started with an understanding of data types of different columns/features present in the dataset. Features were categorized into continuous and categorical features. holiday, weather_description and weather_main were categorical features, whereas temp, rain_1h, snow_1h, clouds_all and traffic_volume were continuous features.

For continuous features, histograms were plotted to check the distribution of different features. Fig1, Fig2, Fig3, Fig4 and Fig5 depict the distribution for "temp", "rain_1h", "snow_1h", "clouds_all" and "traffic_volume" respectively. The distribution for "temp", "rain_1h" and "snow_1h" contain outliers in them as seen in the histogram plots. All the continuous features were also plotted against the "datatime" column to evaluate the trend of the features with respect to time as well.

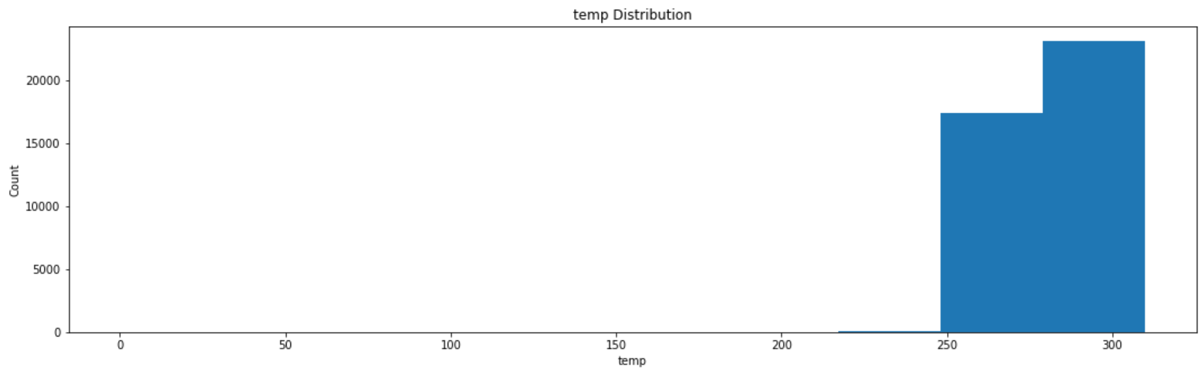

Fig 1 Temp Distribution

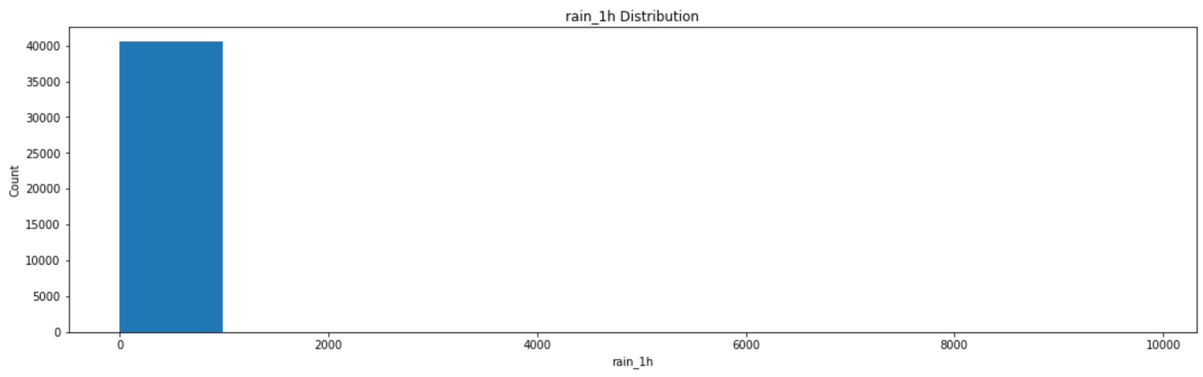

Fig 2 rain_1h Distribution

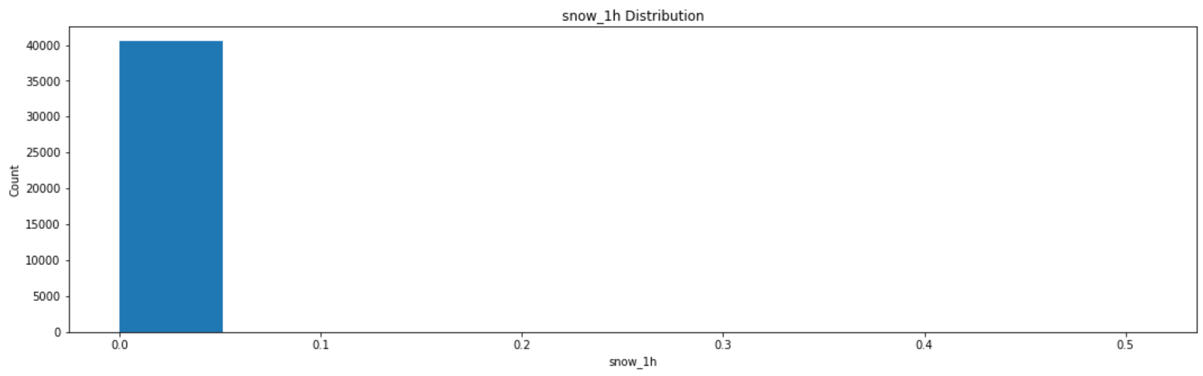

Fig 3 snow_1h Distribution

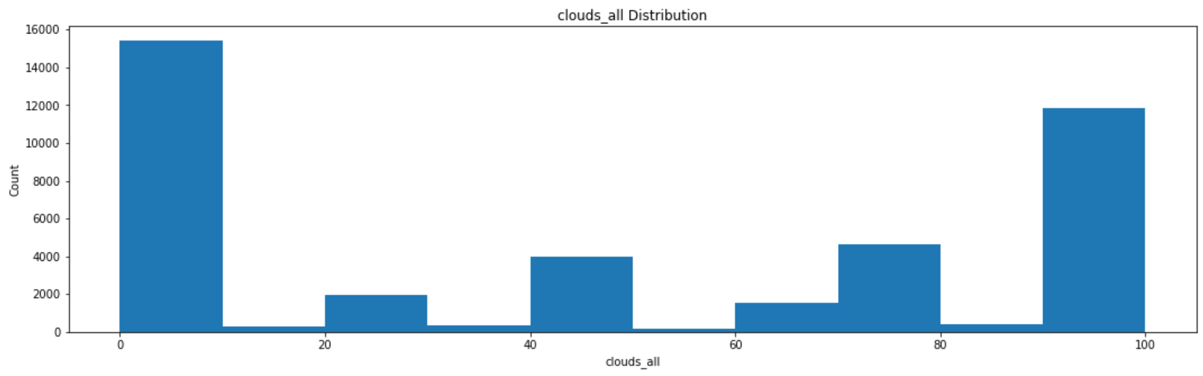

Fig 4 clouds_all Distribution

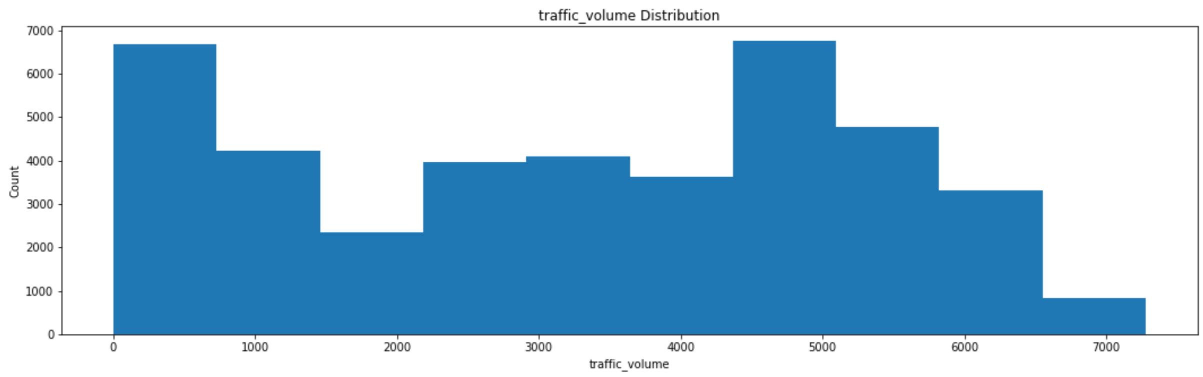

Fig 5 traffic_volume Distribution

For each categorical features, count plots were plotted. Fig 6, Fig 7 and Fig 8 represent count plots for “holiday”, “weather_description” and “weather_main” respectively.

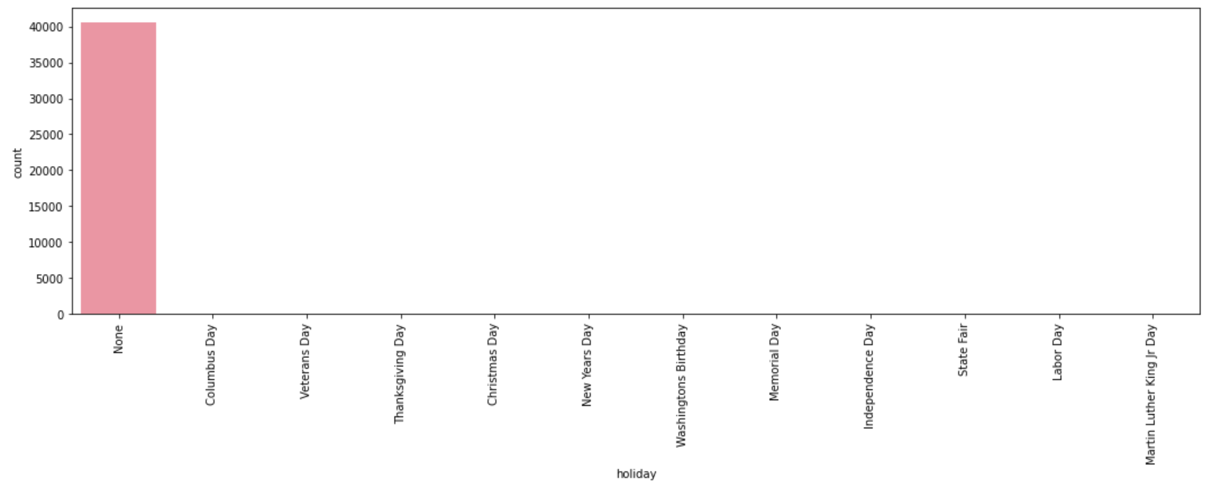

Fig 6 Holiday Count plot

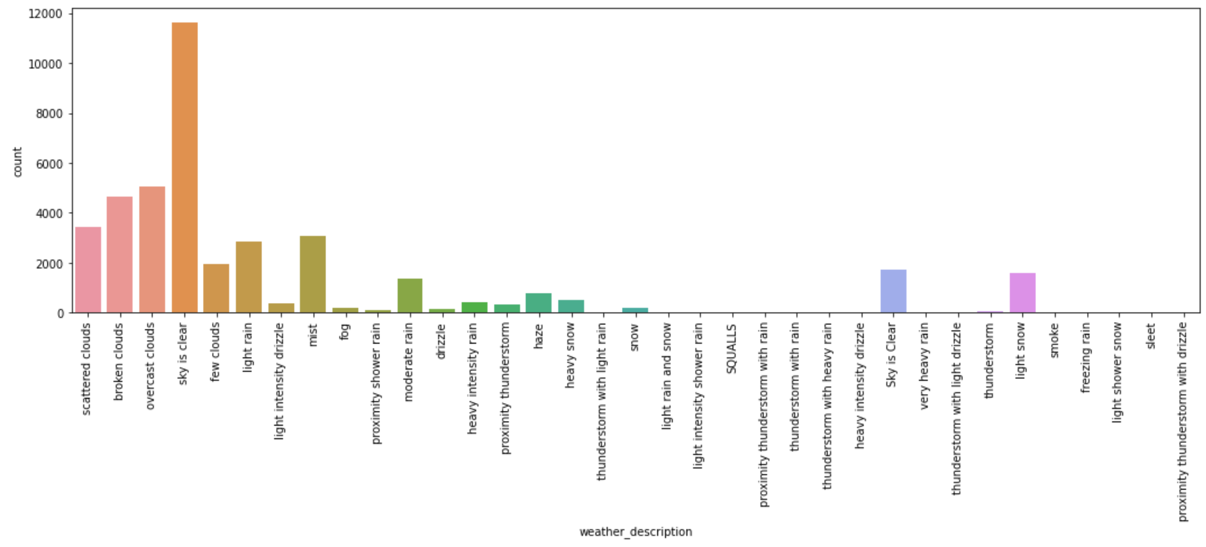

Fig 7 Weather_description Count plot

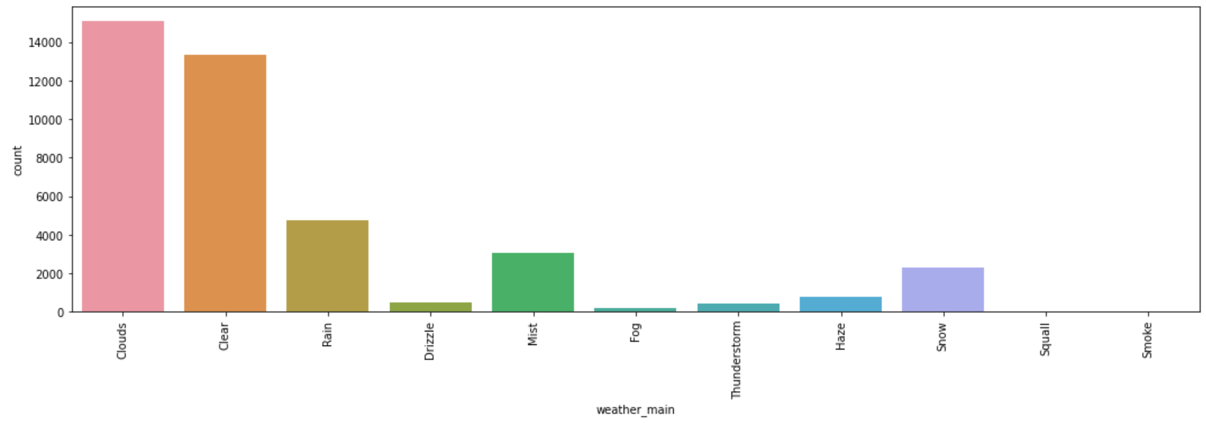

Fig 8 Weather_main Countplot

The “weather_main” and “weather_description” features consisted of a large number of categories with unequal distributions. On the other hand, the “holiday” feature majorly consisted of the “None” value. On further analysis of the “holiday” feature, it was found that 40822 values were None and the remaining values were distributed over 11 different holidays.

Feature engineering:

After diving deep into data analysis of different features, data cleaning and feature engineering were done based on insights gathered from the exploratory data analysis. Data cleaning started with filtering outliers in the three continuous features namely “temp”, “rain_1h” and “snow_1h”. For the “temp” feature, all the values were measured and recorded in Kelvin (K), but one outlier value found was 0. The 0 value was replaced with 273 (Celsius to Kelvin conversion). For the “rain_1h” and “snow_1h” features, multiple outliers were detected. The outliers were scaled down to the mean values since removing them would have affected the model’s performance.

For continuous features, “weather_description” and “weather_main” were dropped as a part of data cleaning. The reason for dropping the two columns was their imbalanced multiple categories. The “holiday” feature was feature engineered and converted into a binary categorical feature. All holidays were mapped to 1 value and all None values were mapped to 0 value.

After data cleaning and feature engineering, the remaining data consisted of 6 different columns out of which 5 were input features and “traffic_volume” was an output feature.

Data Split and Normalization:

After data cleaning and feature engineering, data was split into three subsets namely train, validation and test. The test and validation split consisted of an equal of 5000 rows and the remaining 30575 rows were in the training set making it nearly 80:10:10 split. The test data split consisted of the last 5000 rows whereas the validation split consisted of 5000 rows before the test set rows.

After data was split into three sets, all the three sets were normalized by mean subtraction and standard deviation division. The normalization was done columns wise for each column. The mean and standard deviation obtained were from train data and were used for normalization for all the three splits.

Window Creation:

After the creation of three splits of data, input and output windows were created from a window creation class obtained from the TensorFlow tutorial. The final output of the window object created using the window creation class was the TensorFlow dataset (TFDS) for train, test and valid data splits. Each of the TensorFlow datasets consisted of a tuple of input tensors and output tensors. The input tensors were of shape (batch_size, input_window_size, features) and the output tensors were of shape(batch_size, output_window_size, units). During the creation of the test split Tensorflow Dataset, shuffling was not used to ensure correct predictions generation while submitting.

Model Training:
A total of six models were proposed in this study. A Simple RNN based model was used as a baseline for comparison of a different model. The remaining five models included the following architectures:

1. Bidirectional RNNs
2. LSTM
3. Stacked LSTM
4. Bidirectional LSTM
5. Hybrid 1D CNN- LSTM

The bidirectional RNNs model included consisted of a Bidirectional RNN as compared to a Simple RNN model which was used as a baseline. The LSTM model consisted of the LSTM layer replaced with the RNN layer in the baseline model. The stacked LSTM model consisted of two LSTM layers stacked over each other. The bidirectional LSTM model was similar to the Bidirectional RNN with an RNN layer replaced with an LSTM layer. The last hybrid model consisted of a 1D Convolution layer over a single LSTM layer.
Each of the models was trained for a total of 20 epochs on the training dataset and evaluated on the validation set. Adam optimizer was used in all the model compilation with a learning rate of 0.01.

Comparative Study:
After training and evaluation of all the six models. All the models performed better than the baseline model. However, the Stacked LSTM model performed the best with the least Mean Absolute Error on both the validation and test set. Fig 9 represents the comparison of the performance of all six models on the test and validation set.

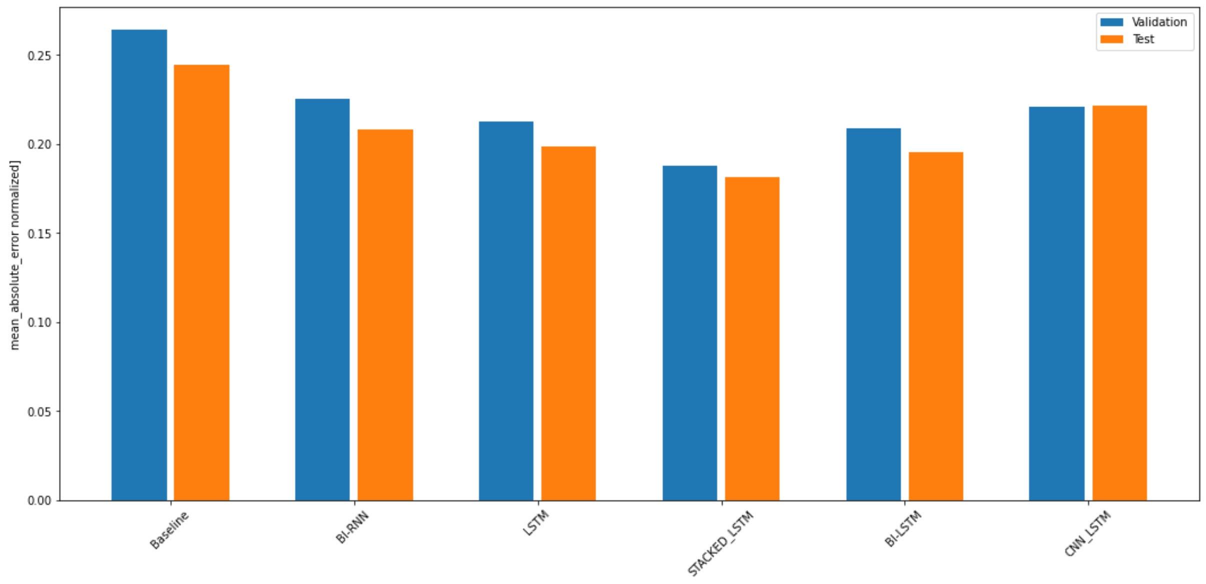

Fig 9 Model Performance Comparison

The second-best performing model was the normal LSTM model followed by BI-LSTM. Surprisingly Bidirectional RNN were performing better than the proposed Hybrid model of 1-CNN and LSTMS. Table 1 contains the mean absolute values of all the model's performance on the validation set.

Model Name	Validation Set MAE
BASELINE	0.2638
Bidirectional RNNs	0.2251
LSTMs	0.2126
STACKED LSTM	**0.1878**
Bidirectional LSTM	0.2084
Hybrid (1D-CNN + LSTM)	0.2209

Table 1: Model's Performance on Validation Set

Submission:

For the submission to Kaggle, the best performing model was fine-tuned for 200 epochs with a learning rate of 0.001. The predictions generated by the model were deformalized by multiplication with the standard deviation, followed by addition with the mean value. The denormalized predictions were then saved in a submission.csv file to submit to the competition.