Model Summary

1. Vector AutoRegressive (VAR) model

**Methodology:**

5 Target variables: number of fatalities, drunk drivers, pedestrians killed, vehicles and persons involved

Used correlation to confirm that these influence each other and hence can use the VAR model to forecast all five variables.

Did the diff and used the ADF (Augmented Dickey Fuller) statistical Test to confirm stationarity of the time series variables.

The coefficient of correlation between two values in a time series, called the autocorrelation function (ACF) was used to identify seasonality in our time series data.

The ACF plot, a bar chart of the coefficients of correlation between a time series and lags of itself, was used to pick the points that are statistically significant.

Partial autocorrelation (PACF), a statistical measure that captures the correlation between two variables after controlling for the effects of other variables, was used to confirm the statistically significant lags.

Decomposed the three parts of a time series – the trend, seasonality and residuals for all 5 variables.

Then came the traditional instantiate, fit, predict and evaluate.

**Combinations attempted:**

- Monthly, weekly and daily time series.

- For monthly and daily, used 3 Targets. For weekly, used all 5.

- Did test size of 5% in monthly, 10% in weekly and daily for train\_test split.

- Monthly needed one diff to make stationary. Weekly and daily were already stationary.

**Model Evaluation:**

**Weekly**: (showing just for fatals)

Baseline RMSE: 142.8255

Training RMSE: 79.9684

Test RMSE: 136.6682

Test RMSE was close to baseline and much worse than Training. Very overfit.

**Daily**: Test RMSE (28) is worse than Baseline (26)!

Best model was monthly.

Used a Train Test Split of 5% since forecasting works better for small time periods and our monthly time series for 11 years has only 132 data points. Using maxlags during fitting, the order of the timeseries (number of lags used in the model) is 13. There are no other hyper-parameters to tune, except to ensure that the time series is stationary.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Baseline RMSE** | | | **Test RMSE** | | | **Baseline MAPE** | | | **Test MAPE** | | |
| fatals | drunk\_dr | peds | fatals | drunk\_dr | peds | fatals | drunk\_dr | peds | fatals | drunk\_dr | peds |
| 761.98 | 190.04 | 190.04 | 503.66 | 139.41 | 67.09 | 0.20 | 0.14 | 0.23 | 0.14 | 0.14 | 0.08 |

\*\*Monthly Model Evaluation:\*\*

The test RMSE is substantially better than the baseline RMSE, for all 3 variables. For fatals, the test RMSE of 503 is 34% better than the baseline of 761.9.

The Mean Absolute Percent Error (MAPE) which shows the percent error indicates that our model’s forecast for fatals is off by 14% on an average month or 503.

The forecast for the peds is the most accurate, its off by 8% or 67 peds per month, in an average month.

Chart

Description automatically generated

\*\*Interpretation\*\* :

The forecast does a pretty good job in following the actual curve except where the data spikes suddenly.

2. RNN

Methodology:

Target variable: number of fatalities,

Predictors: drunk drivers, pedestrians killed, vehicles and persons involved

Recurrent Neural Networks are deep learning models used to solve problems with sequential input data. I used the same five variables in the RNN model, as all the inputs are related to one another, to enable easier comparison between the two models.

Similar to linear regression models, and unlike the time series, these variables are like the X variables used for predictions and the target variable is just the fatals.

I used a small test size of 2% during the train test split and the Keras TimeseriesGenerator to feed 16 sequences (batchsize) of length of 3 (and 5) through our model. Our validation data set is small (only 81 rows) so chose a small batch size of 16. The length is the number of lag observations to be used in the input portion of each sample.

LSTM (long short-term memory) and GRU (Gated Recurrent Units) can alleviate the vanishing gradient problem which is a known shortcoming of RNNS especially with long time series extending over hundreds of periods. Along with Regularization using Dropout layers and EarlyStopping for optimization, I was able to get to a reasonably performing predictive model.

The daily time periods spanning 11 years has a total of 4018 observations, of which the validation data of 2% is 81 observations.

Tried various models with varying sequence of length size and LSTM and GRU layer with regularization, different number of hidden layers and nodes and Dropout.

**\*\*Model Evaluation:\*\***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Baseline RMSE** | **Training RMSE** | **Test RMSE** | **Baseline MAPE** | **Training MAPE** | **Test MAPE** |
| 23.25 | 14.69 | 15.32 | 0.15 | 0.12 | 0.10 |

The RMSE and MAPE confirm the graphs. The Training RMSE is substantially better than baseline and the Test RMSE is only marginally higher than the Training RMSE.

This indicates that our model’s forecast for fatalities is off by 10% on an average day or 15.

Chart, line chart

Description automatically generated

\*\*Interpretation:\*\*

The predictions are quite good, more so in the first few days. Over the total 81 days of predictions, the predictions don't quite match the peaks. In fact, towards the end they overshoot the data considerably.

Chart

Description automatically generated

#### \*\*Interpretation:\*\*

The graphs are almost perfect! The testing RMSE is marginally higher than the Training RMSE, showing the model is not overfit and should do a good job on predicting new unseen data.

**Comparison between both models:**

**VAR**: Testing RMSE: 503/30=16, Testing MAPE: Fatals: 0.14, Peds: 0.08

**RNN**: Testing RMSE: 15.32, Testing MAPE Fatals: 0.10

Both did quite well, though RNN is lightly better on fatalities.

VAR gets to forecast on more than one variable and did a really good on forecasting pedestrian fatalities.

**Best Params:**

**RNN:**

2 GRU hidden layers of 16 and 8 nodes

Dense hidden layer with 8 nodes

Output layer (1 node, regression, relu activation)