Report

Introduction

The assignment requires that we perform time series forecasting with a deliberate time window. The time window chosen is 1 hour, as it serves as a good balance point between forecasting usefulness (predicting for next 15/30 minutes might not be as valuable as for slightly longer intervals), and abundance of data (resampling to >1h bins reduces even further our already limited dataset). A Deep Learning technique was used (LSTM) for the forecasting part, and the Loss function to be minimized was RMSE (Root Mean Squared Error).

Main Classes & Methods

dataLoader: contains all processes for loading, cleaning, aggregating and enriching the data.

- cleanData(): Data is explored in terms of NaN values and a plot is exported to the default export path ('./cleaning'), unless stated otherwise. The missing data in our case is less than 1% and therefore is dropped.
- mergeWeatherData(): Weather data is loaded for the city of Lima to provide the grounds of a multivariate time series analysis. The data from routes.csv is resampled to an hourly manner and merged with the weather data to form our final dataset.

clusterAnalysis: contains code for exploring K-means for both source and destination coordinates, performing K-Means with the chosen value of K, and exporting map images with plotted clusters.

- clusterExploration(): Exploration of K for the K-means clustering algorithm in a default range of (5,30) for coordinate data and export of the 'elbow rule' chart.
- clusterData(): Given an optimal value for K, cluster the data based on source and destination request coordinates.
- drawClustersOnMap(): draw the clustered data on a map image of Lima and export it to 'clusterpath' ('./clustering' by default)

TS_Analysis: contains the code required to perform a multivariate Time Series Analysis on the data

- plotTS(): Exports the plots for all TS columns in the data ('requests', 'Temperature', 'WindSpeed' and 'Precipitation' to 'exportpath' (default './TS Decomposition)'.
- TSDecomposition(): for the TS columns mentioned above, a TS Decomposition is performed, in other words an analysis of Observed Values, Trend, Seasonality. The plots are exported in the same folder.
- stationarityTests(): performs the Augmented Dickey-Fuller test to test for unit-root in the TS, thus for stationarity. Results are exported in .csv format in the same folder.

forecastModel: contains the code for the forecasting and exporting the corresponding plots.

- plotTimeSeries(): this only serves as an initial plot of how the data will be split to train and test sets. Exported to 'exportpath' (default is './model')
- preprocess(): removes the TS index, scales data to (0,1), generates the labels based on our data, split to train/test and reshapes the data from [samples, features] to [samples, 1, features], as the expected input shape is for LSTMs through the *keras* library .
- buildModel(): calls defineModel() and if there is already a model (parameters) saved in our disk (thus the process has been rerun in the past), it calls loadModel() otherwise it calls trainModel().
- defineModel(): define LSTM model architecture
- loadModel(): loads the model parameters from disk.
- trainModel(): given a set of hyperparameters, tranModel() trains an LSTM model and saves the parameters to disk for reusability.
- plotTrainHistory(): plots the train/validation loss and exports it to 'exportpath'
- evaluatePlot(): inverse scales the predictions and plots the actual values against the predicted ones by our model. Result is again exported to 'exportpath'

Notes

1. Hyperparameters Explored for the LSTM model include: different architectures (LSTM units and layers), activation functions (relu, leaky-relu, sigmoid), optimizers (adam, rms_prop, adagrad), batch_size (8/16/32/64), epochs (5/10/20/40) and some learning rate experiments. Early stopping was also considered, given the fact that we quickly reach a plateau performance-wise during training. The model does not seem to be very sensitive to most hyperparameters, as was expected due to our limited data.
2. TS does not explicitly have to be stationary for LSTM forecasting (as opposed to other forecasting techniques), but according to literature, stationarity helps even in LSTM learning. Hence the reason of our testing.

Future Work

1. As in most ML/DL tasks, adding more features (e.g. religious/public holidays for Lima, event information, etc) could significantly improve our forecasting performance.
2. It is obvious from our training history that we underfit, which is common in Deep Learning solutions, given that we have fairly limited data. A good alternative to try, should we have more time, would be the Prophet model, developed by Facebook
3. Finally, since K-folds CV is not applicable in its vanilla form to Time Series data, what we could do to completely automate the hyperparameter exploration of our model, as well as monitor overfitting/underfitting during train time, is to set up Walk-Forward cross-validation, as described here.

George Gousios