Session 4: Time Series Analysis

The more difficult models are often quite hard to implement. If you want to train and test such a model on your own dataset, it is easier to look for a good GitHub/Kaggle implementation and try to apply it on your own dataset. It can be used as a blueprint, both for the preprocessing and the modeling steps. Note that it is still important to reason about your data and to change preprocessing and modeling parameters if necessary! But no need to create everything from scratch if there is already a good source available.

The focus of this assignment is to run through a notebook that already contains implementations of different time series models and to apply it on your own dataset. The different questions will guide and 'force' you to understand what is happening in the code. If you manage to answer everything in a correct way, you will gain insight in what is happening and what to change in order to apply it in your own dataset. You will notice that the creation of certain time series models is not so straightforward anymore.

Source: download the notebook on Leho. In the notebook, they work with Tesla stock data. The goal is to apply this code to the Google stock data shown in the demo. This notebook is a slightly updated version from a file on Kaggle.

Goal of the model: forecast Google stock prices

Go through the code and answer the questions below.

Part 1: data collection

Question 1:

```
[4]  def show(data: pd.DataFrame):
         df = data.copy()
         df = df.style.format(precision=3)
         df = df.background_gradient(cmap='Reds', axis=0)
         display(df)

     def highlight_half(data: pd.DataFrame, axis=1, precision=3):

         s = data.shape[1] if axis else data.shape[0]
         data_style = data.style.format(precision=precision)

         def apply_style(val):
             style1 = 'background-color: red; color: white'
             style2 = 'background-color: blue; color: white'
             return [style1 if x < s//2 else style2 for x in range(s)]

         display(data_style.apply(apply_style, axis=axis))
```

What is wrong coding practice here? This shows that you always have to pay attention when using other people's code.

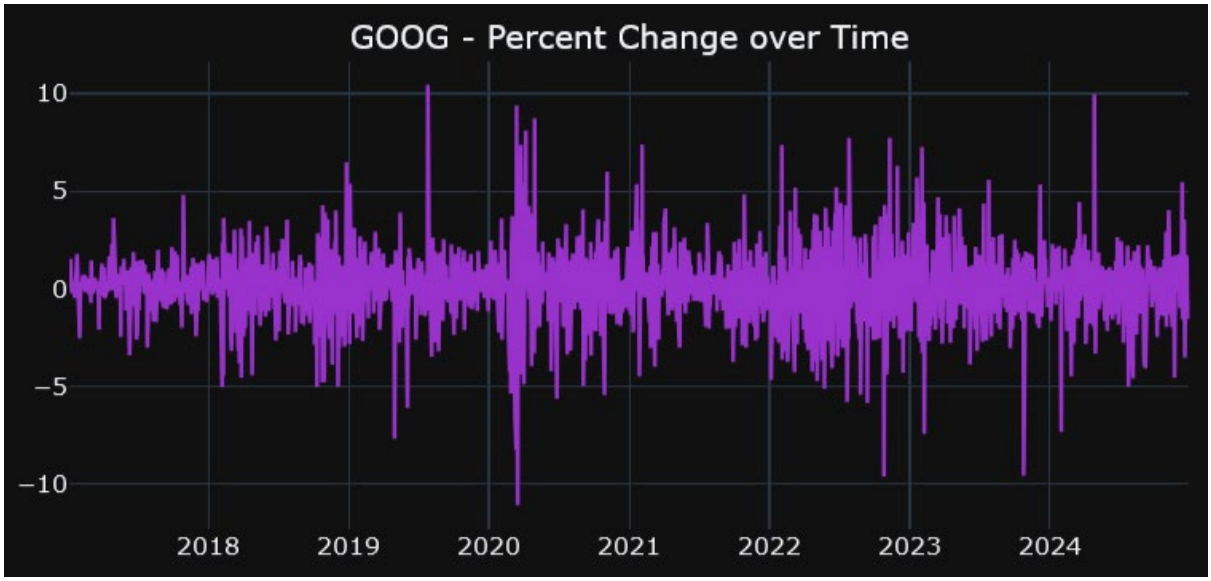
Question 2: Use Google stock data instead of Tesla. You can use the data given in the demo, but in the assignment notebook there is also another way to download this data. Describe how, use the data from 01/01/2017 up until 31/12/2024 and use the same columns as for the Tesla data. Check the meaning of these columns.

Part 2: data visualisation

Question 3: Check the following plot:

Inspect the different moving averages. What happens when they are small or large? Should you use the small or large ones?

Question 4: Describe this plot:



Part 2: data preparation

Question 5: Describe in your own words what happens in the following methods:

- data_mapping
- rnn_train_test_split
- forecast_n_steps + what is the danger when you apply this?

You notice that formatting your data for sequential problems is not that easy anymore. It is quite complex and requires a lot of thinking in order to put everything in the right order. It is easy to make mistakes in this part so please think well before applying code to your dataset.

Question 6: Explain what happens here:

```
# Initializing class
mapping_steps = 32 # ~ 1 months in buisness days
rnn_formater = RNNFormater(df[columns], mapping_steps=mapping_steps)

# Mapping steps
norm_data_mapped = rnn_formater.data_mapping() # n_steps -> y
# print(f'Mapped Normalized data step 0:\n{norm_data_mapped[0]}')
print(f'Normalized data shape: {norm_data_mapped[0].shape}')
```

```
Normalized data shape: (33, 5)
```

```
[23] # Train Test Split
X_train, X_test, y_train, y_test = rnn_formater.rnn_train_test_split(test_percent=0.05)
print(f'Number of time steps for test set: {rnn_formater.test_size}')

print(f'X shape: {X_train.shape}')
# print(X_train[0])

print(f'y shape: {y_train.shape}')
# print(y_train[0])
```

Why is the output of the first cell shape (33, 5)?

Part 3: vanilla LSTM model building

The author proposes the following model:

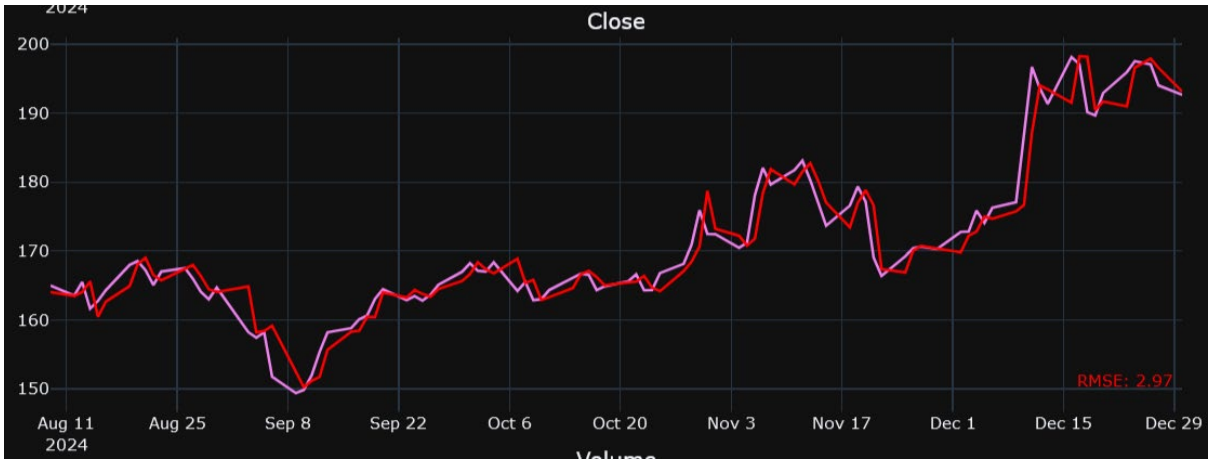
```python
# Vanilla LSTM
model = models.Sequential([
    LSTM(units=80, input_shape=(mapping_steps, len(columns))),
    Dropout(0.05),
    Dense(units=len(columns))
])

# Compiling model
model.compile(optimizer='adam', loss='mae')
model.summary()
```

Personally, I tend to keep the amount of units in the layers a power of two (2-4-8-16…) instead of 80 and avoid dropout for sequential models because it is not guaranteed to give a better result. Consider playing with this on your own data.

The author then creates a lot of different plots using plotly. The details of how to create them are less important for this exercise. More important is to be able to interpret them.

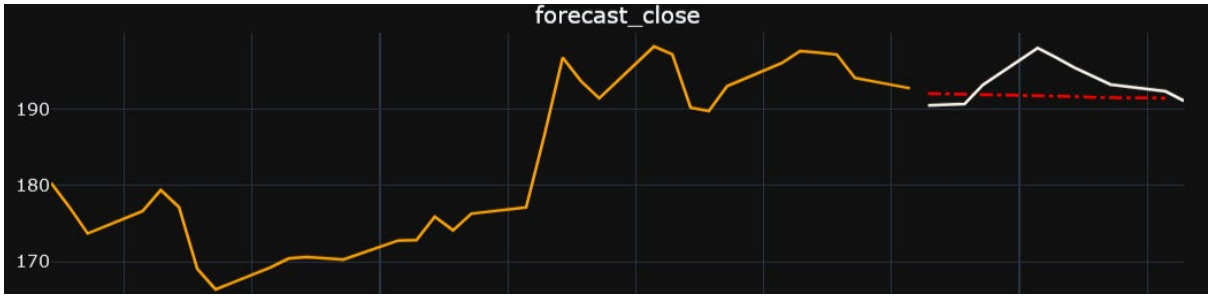
Question 7: describe the course of the training and validation loss. Do you think this is good?

Question 8: the predictions are very good. Below is the example for the close price. Does this mean that we can forecast the price accurately for months in advance now? What should we take into account?



The author then applies the trained model for 10 business days in advance, using forecasted values as new input to forecast other values further into the future.

Question 9: describe the plot below:

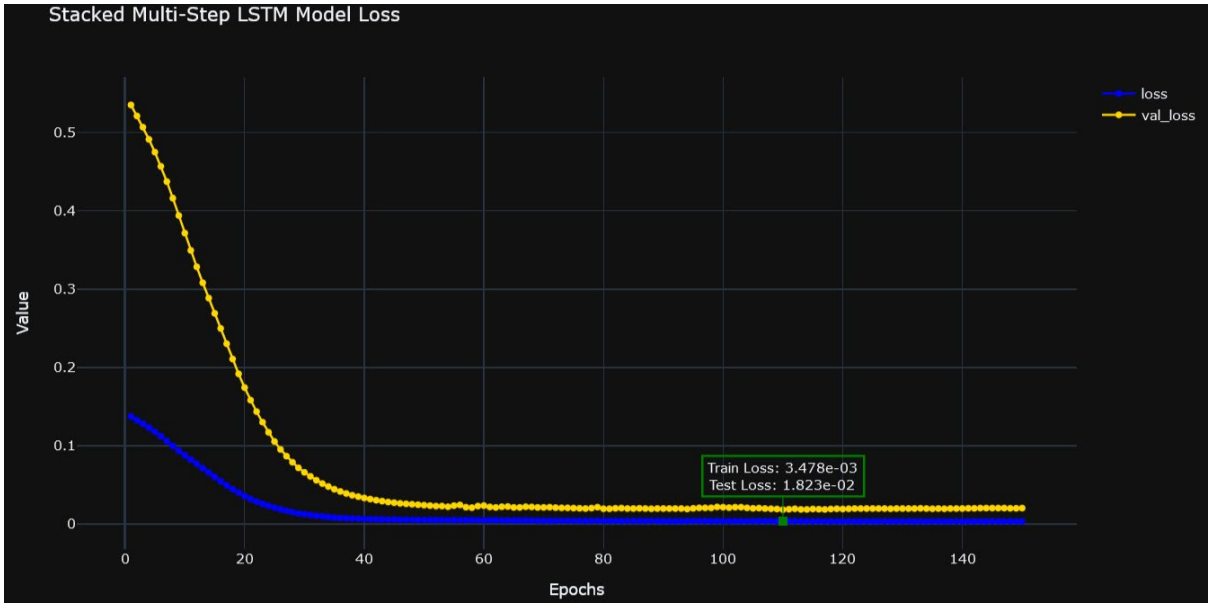


Part 4: Multi-Step Stacked LSTM model building

Question 10: What is the difference between data_mapping, rnn_train_test_split and multi_step_forecast from this model and the previous one in part 3?

The author then describes the use of a Bidirectional LSTM. Personally, I'm not such a fan of using this on time series data. Since most of the time, you do not know the values in the future and those are the exact values that you want to forecast. This is also probably why the author does not mention the bidirectional LSTM anymore in the remainder of the code.

Question 11: discuss the following plot:



If you continue and plot the forecasted values, you notice this for close:

The model is terrible! And no, it's not due to a mistake.

Question 12: Why do you think that the model is bad? Try to find a better forecast. Think about a possible solution.

Question 13: What is the difference between the Multi-Step Stacked LSTM model from this exercise and the sequence-to-sequence model seen in the theory? Which one do you think will perform better?

Possible extension: apply a sequence to sequence network to this data in order to forecast for multiple timesteps. **This part is not mandatory.**

Note that in this exercise, it was easy to apply the code to the Google data instead of the Tesla data, because it was in exactly the same format. In practice, you can also apply this code on whatever forecasting problem for time series that you have. It will be more difficult to put the data in the right format and to select the useful features. You might even need to add new features or add data from other sources. The structure behind the project stays the same. But again, you have to think well at every step and not just copy code from someone else or from ChatGPT.

Source of this exercise:

https://www.kaggle.com/code/guslovesmath/tesla-stock-forecasting-multi-step-stacked-lstm/notebook

Another interesting notebook for time series:

https://www.kaggle.com/code/nicapotato/keras-timeseries-multi-step-multi-output#Part-2:-Forecast-a-multivariate-time-series