Data Science and Visualisation Techniques applied on Bus Search Requests and its Correlating Booking Data

Subtitle

Bachelor Thesis

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science in Engineering

to the University of Applied Sciences FH Campus Wien
Bachelor Degree Program: Computer Science and Digital Communications

Author:

first name surname

Student identification number:

Number

Supervisor:

Title first name surname

Date:

dd.mm.yyyy

Declaration of authorship:

I declare that this Bachelor Thesis has been written by myself. I have not used any other than the listed sources, nor have I received any unauthorized help.

I hereby certify that I have not submitted this Bachelor Thesis in any form (to a reviewer for assessment) either in Austria or abroad.

Furthermore, I assure that the (printed and electronic) copies I have submitted are identical.

Date: Signature:

Abstract

(E.g. "This thesis investigates...")

Kurzfassung

(Z.B. “Diese Arbeit untersucht...”)

List of Abbreviations

ARP	Address Resolution Protocol
GPRS	General Packet Radio Service
GSM	Global System for Mobile communication
WLAN	Wireless Local Area Network

Key Terms

GSM
Mobilfunk
Zugriffsverfahren

Contents

1 Introduction

2 Info

2.0.1 Research Question

What Neural Networks are suitable to predict future bus bookings and which attributes of bus search requests can be utilized to create an analytical dashboard.

Research Question BA1: How to create a theoretical framework that covers basic DS aspects and visualization techniques in order to develop an analytical web-based dashboard

2.0.2 notes

Improve add targeting, by search hours

Marketing for potential "ausflugziele" along the often search connections

the averages

2.0.3 General

Basic Idea/Content:

- Explain available dataset, data structure, how data is gathered
- Explain what techniques are used to clean the base dataset
- Interdisciplinary - explain why certain KPIs or models are choosen and are applied onto the dataset
- Prediction Model - explain the technologies, methods etc. used to create a ML based prediction model (To improve Yield Management). Maybe two models, Supervised Learning and unsupervised learning
- Data Clustering and other KPIs + applied statistical models (e.g. Clustering, LR), Heatmaps etc.
- Visualisation techniques used to display results of the applied statistical models
- Maybe? Short chapter about technical setup of the Dashboard

In general i would appreciate some general feedback what else could/should be descriped within this thesis. I think the Prediction Model will be the main aspect of this thesis (How it is created + which techniques are used, how is the performance when comparing prediction to actual booking data etc.)

2.0.4 Lit Research

[1] - provides also a lot of useful references to other papers that can be used

[2] - ML

[3], [4], [5], [6] - Tensorflow, ML etc.

[7], [8] - interdisciplinary to provide context which KPI's etc are chosen etc.

3 Available Dataset

This chapter focuses on explaining and analysing the available data. The data is analyzed for Business Intelligence (BI) purposes as well as on metrics that can be used to create predictions. Whereas BI [?] focuses on historical data and aims to support managers to make decisions traditional methods like predictive analytic asses potential future scenarios using advanced statistical methods [?].

3.1 Data Origin

The available dataset is gathered from a website that provides a service to find and book buses for individual journeys. This service is currently available in Austria, Germany, Switzerland and Lichtenstein. The buses itself are offered in real time by various different bus companies. Offers can vary in price which is based bus calculations which may vary from operator to operator. The data is stored in a relational database. Since the service also provides the possibility to directly book a bus, booking and corresponding user meta data is available as well.

3.2 Data Structure

The service launched in March 2017 therefore booking data is available back to this date. Tracking the search requests was introduced in October 2020. The request table itself keeps track of 40 attributes but not all of them host valuable information that could be analysed therefore only the ones which can be analysed are listed and explained below:

- `task_id` - PK (incremented value)
- `createdAt` - At which time the search request was made.
- `accountId` - Not empty when the user is currently logged in
- `amountSearchResults` - How many buses can be offered
- `containsTripCompany` - If the user wants to stop at a certain company during the trip
- `distanceInMeters` - Distance between departure and destination place
- `durationInSeconds` - Duration of the trip
- `pax` - Amount of passengers
- `taskFrom_address` - Departure address
- `taskFrom_lat and lng` - Latitude and Longitude of the departure
- `taskTo_address` - Destination address

- taskTo_lat and lng - Latitude and Longitude of the destination
- taskFrom_Time - Desired departure time
- taskTo_Time - Estimated arrival time
- cheapestPrice_amount - The cheapest price for a bus
- bus_id - The operator with the cheapest bus
- city - From which city the request was made
- country - From which country the request was made

Whenever a booking is made the correlating data is stored within a booking table. As the booking table contains sensitive data which is not scope of the analysis, so only three attributes are used:

- createdAt - At which time the booking was made.
- company_id - FK used to identify who received the booking
- task_id - FK used to link the booking to an search request
- taskTime_from - Indicates the day of departure
- basePrice_amount - The price the customer has to pay

3.3 Data Cleansing

During this process the available data is investigated for irregularities that cause distortion when applying statistical models.

Search requests are tracked whenever a user opens the service and searches for a certain connection. Given that behaviour it may occur that a user searches for the same connection within a short time window. This behaviour results in the need of de-duplication to avoid bias. To filter out duplicates the attributes ipHash, createdAt, taskFrom_address and taskTo_address. A search request is considered as non duplicate whenever the timespan between equal entries is larger than one hour. To pre-process the data the following logic is applied once //todoChange:

```
    query = '''
    DELETE t1
    FROM search_requests_clean t1
    INNER JOIN search_requests_clean t2
        ON t1.taskFrom_address = t2.taskFrom_address
        AND t1.taskTo_address = t2.taskTo_address
        AND t1.ipHash = t2.ipHash
        AND t1.createdAt > t2.createdAt
        AND t1.createdAt - t2.createdAt <= %s
  '''

    timespan = 3600  # 3600 seconds  - 1 hour
    cursor = connection.cursor()
    cursor.execute(query, (timeframe,))
    connection.commit()
```

//todo more explanation The logic above compares all entries based on the attributes mentioned above removes equal entries that are within a timespan of 1 hour.

Regarding validation and norming the available data present in both tables no actions are required due to fact that attributes that do not meet their defined data types are not stored in first place.

3.4 Data Augmentation

Starting from March 2020 countries like Austria, Germany, Switzerland and Lichtenstein had to put travel restrictions into effect due to the ongoing Covid19 pandemic
citeHere. This travel restrictions impacted the gathered booking data as those restrictions forbid travelling.

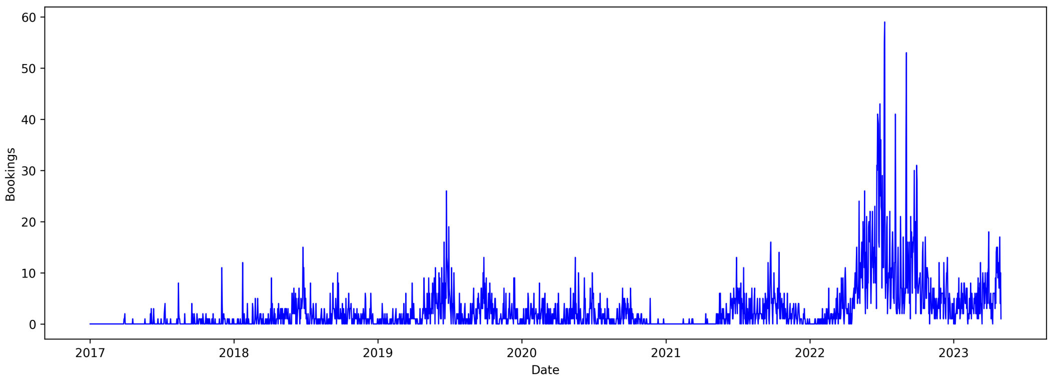

Figure 3.1: Bookings over time - [source:author]

Figure 3.1 highlights the drop of bookings starting from March 2020 until June 2022. The extreme spikes which can be observed during summer 2022 is related to two events taking place in Spielberg in Austria. Both the Formula 1 as well as MotoGP utilized the service to organise the bus shuttles. All buses that were operating as shuttles were booked via busfinder. As busfinder To achieve reliable results when utilizing this data for a time series forecasting ML model this time period needs augmentation. When analysing the chart 3.1 an continuous growth of bookings is visible until 2021. One way to augment the data
citehere is calculate the average growth during this time span. To substitute the distorted data the current data is replaced by the value of the previous year. This value is then multiplied by the average growth. Furthermore missing timestamps throughout the whole time series are added with a value of zero. The following logic is applied to the data frame:

```
df = db.get_booking_data()
average_growth = df['bookings'].pct_change().mean()
substitute_corona = pd.date_range(start='2020-03-01', end='2022-05-01', freq='D')
df['date(createdAt)'] = pd.to_datetime(df['date(taskTime_from)'])
df = (df.set_index('date(taskTime_from)')
      .reindex(pd.date_range('2018-01-01', '2023-05-01', freq='D'))
      .rename_axis(['date(taskTime_from)'])
      .fillna(0)
      .reset_index())

df.set_index('date(taskTime_from)', inplace=True)

for date in substitute_corona:
    year_ago = str(date - relativedelta(years=1)).split(" ")[0]
```

```
    val = int(math.ceil(df.loc[year_ago]['bookings'] * (1+average_growth)))
    df.loc[str(date).split(" ")[0]] = val
```

The average growth per anno is around 30%. After applying the logic the data set looks the following:

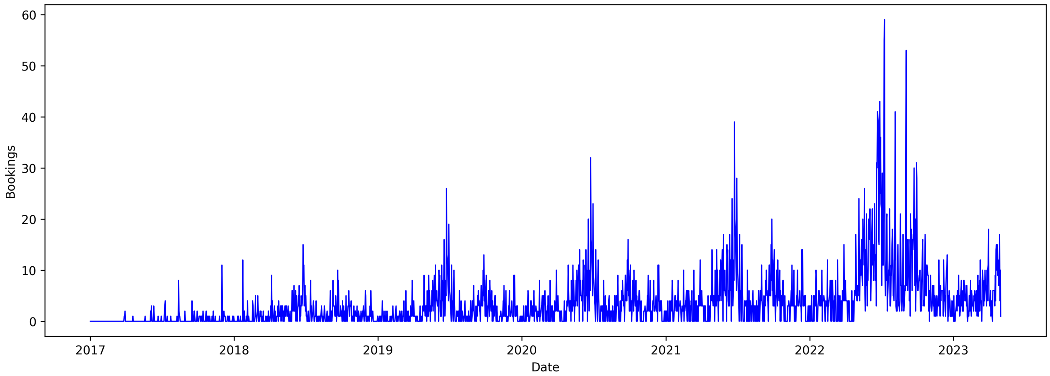

Figure 3.2: Augmented Data Set - [source:author]

The impact of this augmentation in terms of prediction accuracy is compared in chapter **??**.

4 What insights can be gathered?

This section focuses on which potential information from the available dataset can be extracted and utilized for a reporting dashboard. Furthermore the dataset will be analysed for metrics that can support and improve the current yield management.

4.1 Improving the Yield Management

In general Yield Management (YM) describes the way how limited resources like hotel rooms, seats within an air plane or available buses are assigned to customers by leveraging the highest possible revenue. American Airlines claims that by utilizing YM they are able to increase their revenue by 500 million dollars per year [9]. Before integrating YM a few considerations about the characteristics of teh provided service need to be made otherwise YM might actually cause a decrease in revenue. The following characteristics are suitable for the utilization of YM:[9]

- Storing surplus resources can either be costly or unattainable.
- Whenever future demand is uncertain commitments need to be done.
- It is possible to differentiate between customer segments
- A single unit can be used to provide different services
- The company is not legally limited in their actions of selling a certain service or not.

As YM is already in place at busfinder one of the characteristics mentioned above comes along with a high uncertainty. Although commitments for uncertain future demand are made the ability to predict future bookings would further improve the YM. Being able to predict future bookings during ordinary market conditions (e.g. no travel restrictions in place) those estimates directly can be used to influence factor of capacity management - How many buses are available? This directly influences the pricing strategy because a higher demand results in a higher price.

Both Artificial Neural Networks (ANNs) and their usage for time series forecasting further evolved over the past years. Additionally libraries like tensorflow reduce the complexity of developing ANNs. Hence this method is a suitable solution to predict future bookings. Therefore the basics of ANN and the development of two different models are explained in chapter 5.

4.2 Averages

As averages may seem trivial they still can provide valuable information. By comparing averages over time trends in the market become visible. However their usage should always been in combination with additional statistical measures. When analysing the dataset the following attributes could indicate market trends:

- `pax`

- distanceInMeters

By looking on the average PAX over a certain time period the metric indicates weather the number of passengers increase, decreases or stalls over time. This information along with additional statistics can assist an operator in their future planing when it comes to their fleet management. As the average in this case indicates the demand for required seats a bus should have. For example the major part of an operator's fleet are buses with 90 seats but his average PAX is around 60 which is decreasing considerations about buying smaller buses can be made. This would improve the cost-efficiency as smaller buses are cheaper, consume less petrol an maintenance costs are lower. Furthermore a decrease or increase of the average travelled distance can be used to decide weather or not electric buses might be an alternative. As the example stated above applying the average on those parameters without any filters in place do not provide any significant information. Therefore the averages are used together with additional characteristics gathered from the data set like the grouping of certain attributes.

4.3 Grouping Data

One part of the statistical analysis is data grouping. One reason why data is grouped is to simplify complex data structures. Furthermore it enables the possibility to summarize certain characteristics present within the data set. Another benefit of grouping data is that it might reveal potential relationships. Additionally grouping can be used to improve predictive models as demonstrated in section5.6. Analysing the given data set reveals the following attributes offer valuable insights when grouping is applied to them:

- `taskFrom_lat/lng` and `taskTo_lat/lng`
- `createdAt`
- `taskFrom_time`

4.3.1 Geographical Grouping

As latitude and longitude are numerical described using numerical values their usage for geographical grouping is preferred over string values which are used for attributes like `taskFrom_address` and `taskTo_address`. Furthermore mathematical operations on coordinates like calculate the distance between two starting points allow us to modify how the data is grouped. Therefore the geographical data is utilized to group search requests by their departure and destination location. This provides the bus operator insights about popular routes. As bus operators determine maximal travel distances to departure places depending on their logistical base this data might reveal connections with high frequency. Depending on the additional distance the operator might need and it's current utilization an operator might decide to add an exception for this specific region to allocate additional bookings. Furthermore this information could be used to influence the pricing strategy for routes with high demand. Geographical grouping in combination with the attribute `amountSearchResults` can be applied to improve the offered service. Whenever `amountSearchResults` is equal to zero no buses were offered for a certain request. As the grouping might reveal high demand for certain connections bus operators could consider to supply those connections as there are no competitors in this region. This results in a higher utilization of the operators bus fleet.

4.3.2 Grouping by Date

Grouping search requests by date reveals valuable information that can be used for marketing purposes. Grouping request on an hourly basis reveals information about daytimes with high or low user frequency. This fact can be utilized to optimize potential ad campaigns. Grouping bookings by departure date reveals dates with high demand. Furthermore this metric is used to train the prediction models described in chapter 5. As this information indicates the availability of buses on the given day. Furthermore seasonal trends become visible.

4.4 Conversion Rates

The implementation and visualisation of topics discussed in this section are carried out in chapter 6
The carried out analysis does not claim for completeness. As there are several insights that can be gathered from the requests. The data set is analysed for metrics that provide insights that support operators as well as busfinder in their decision making and to further improve their service.

5 Predicting Future Bookings

The knowledge of potential future bookings provide useful insights when it comes to yield management. Yield management in general describes controlling price and capacity control in a simultaneous ways [10]. Therefore those predictions can be used to support bus operators in their pricing strategy. This chapter focuses on creating two prediction models utilizing different techniques based on the data that is available. Both models are implemented using python and the following libraries.

- `matplotlib`[1] - used for plotting
- `pandas`[2] - used for data manipulation
- `tesnorflow`[3] - provides ML models
- `keras`[4] - Neural Network library

As there are various models available a literature review was conducted to figure out which models fit the purpose of time series forecasting. It turns out that the most promising NN that can be utilized for time series prediction are either Convolutional Neural Networks (CNN) or Recurrent Neural Networks (RNN) especially Long Short-Term Memory (LSTM)[11][12][13][14].

5.1 Backpropagation

As both of the models LSTM and CNN use back propagation (BP) for its training the basic concepts of the algorithm are explained in this section. To understand the logic of backpropagation a few terms need a detailed explanation:

Gradient

The gardient also called gradient descent is an algorithm that is used to optimize the loss function within backpropagation. That means that the gradient descent indicates by how much the weights and biases need to be adjusted in order to reduce the actual error value which is the result of the applied loss function. [15]

Bias

The bias is an additional parameter used in each neuron of a NN. It is used to directly influence the activation function to offset the results either to the negative or positive direction. When looking at the sigmoid function without any bias in place where `x` correlates to the input value and `w` indicates the used weight:

$$\sigma(x) = \frac{1}{1 + e^{-(w*x)}} \tag{5.1}$$

[1] https://matplotlib.org/
[2] https://pandas.pydata.org/
[3] https://www.tensorflow.org/
[4] https://keras.io/

When looking at figure 5.1 the weights only influence the steepness of the function but won't shift it along the x axes.

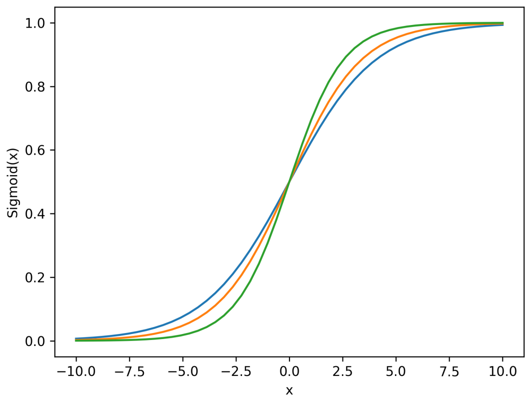

Figure 5.1: Sigmoid function with different weights and no bias - [source:[16]]

To shift the function along the x axes the sigmoid function is adapted with the bias b value:

$$\sigma(x) = \frac{1}{1 + e^{-(w*x+b)}} \tag{5.2}$$

By adding the bias value as constant to the sigmoid function can be shifted along the x axes as shown in 5.2.

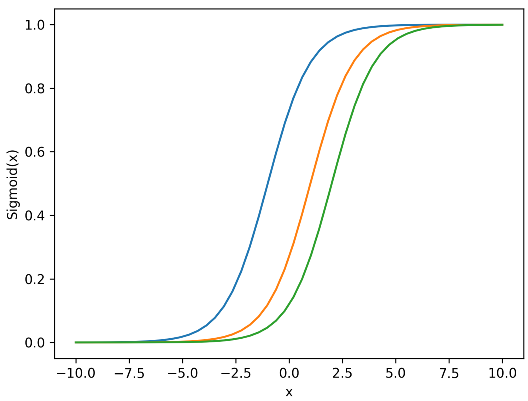

Figure 5.2: Sigmoid function with same weights but different bias - [source:author]

The bias therefore is utilized to directly influence the result of the activation function and whether or not a certain neuron is activated or not.

Activation Function

Activation functions introduce non-linearity characteristics to NN. This function is applied to the output of a neuron and decides weather or not a neuron is activated or not.[17]. In combination with the descent gradient this function enables the NN to learn complex patterns within a training set. Orignially the sigmoid function [15] was used as activation function for NN but as of today multiple other functions like softmax, Tanh, ReLu emerged [17].

Phases

Backpropagation follows an iterative process. At the beginning there is the feed forward pass. During this phase the input data is passed through all layers. Each hidden layer applies a linear function to create certain weights those outputs then are fed to the activation function. Depending on how many hidden layers are used within the NN the output of the activation function is used as input for the next neuron. At the end the predicted outputs by the model are compared with the actual outputs of the training data. This comparison is evaluated through a loss function. At this step the actual learning process of the model starts by calculating the gradient of the loss function considering the output of the NN. After that the back pass is initiated. Along this phase the gradients of each previous layers are multiplied with a local gradient it's weights which results in a gradient in respect to the layer's inputs. After receiving all gradients based on the network's input data, all weights and biases are updated and optimized to reduce the result of the loss function. The steps forward pass, loss calculation, back pass, and the updates of weight and biases are repeated to improve the network in an iterative way. [15] Figure 5.3 demonstrates the logic of the backpropagation algorithm. Whereas wu represents the updated weights after each iteration.

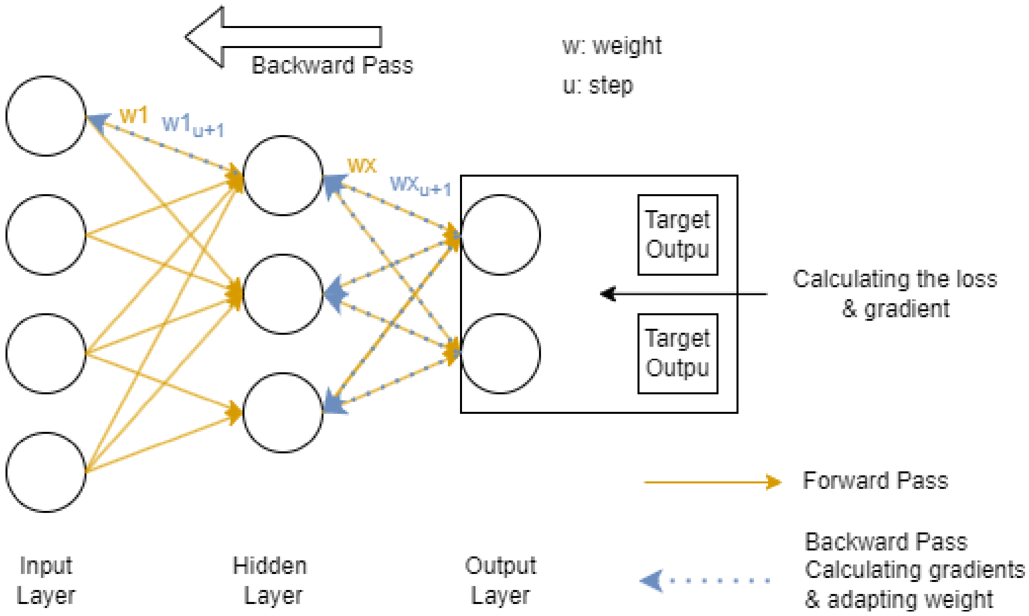

Figure 5.3: Simplified logic of the backpropagation algorithmus - [source:[15]]

5.2 The Models

Both models CNN and RNN/LSTM can be used for time series forecasting. To create accurate prediction models a basic knowledge about models functionality is required. Therefore this section explains the components of each NN as well as the approaches those models follow.

5.2.1 LSTM

LSTM is an RNN and was invented by [18] in 1997. Until today this NN is widley used for time series forecasting and provides reliable results for short as well as long term predictions [19]. LSTM have so called memory cells which are responsible to store the state of data.

Whenever information arrives at a memory cell its outcome is defined by refreshing the cell state with the newly arrived information. LSTM utilizes gates to control a cells state by either including or excluding information [20]. The gates are called:

- input gate - data selection and storage for upcoming state
- forget gate - data selection and storage which will not be used for the upcoming state
- output gate - sets information within the state that is send to the output

Those gates are created by combining sigmoid functions. The results of this gates are values ranging from zero to one. A result of zero indicates the cell to not pass any infomration whereas values close to one indicates the cell to pass all information. The LSTM Module or Repeating module consists of four NN layers which interact together as shown in Figure 5.4:

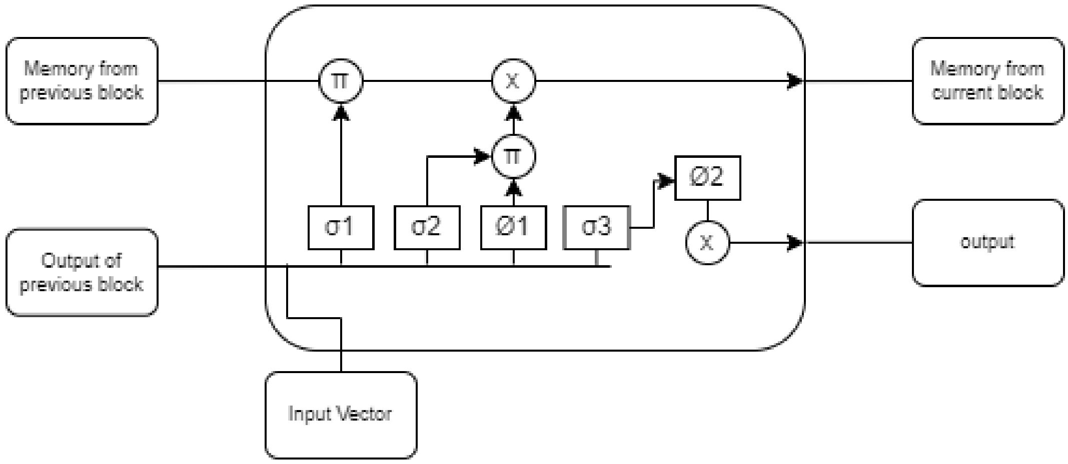

Figure 5.4: Repeating LSTM Module - [source:[16]]

In total the repeating model has 3 gate activation functions which are named σ_1, σ_2, σ_3 and shown in figure 5.4. Furthermore σ_1 and σ_2 act as output activation functions too. The cell state is illustrated using a blue line which starts at St-1 which indicates the previous memory block to St representing the current memory block. The amount of information that is passed is regulated by layer σ_1 using the following function:

$$cf_t = \sigma_1(W_cf * [O_t - 1, x_t] + b_cf)[16] \tag{5.3}$$

Furthermore two network layers are used to store new information to the cell state. Therefore sigmoid layer σ_2 chooses the values which are updated by utilizing the following formula:

$$l_t = \sigma_2(W_1 * [O_t - 1, x_t] + b_l)[16] \tag{5.4}$$

Layer ϕ_1 or `tanh` is created by using new candidate values. This layer outputs a vector by utilzing the following formular:

$$\widetilde{S}_t = tanh(W_s * [O_t - 1, X_t] + b_s)[16] \tag{5.5}$$

The last step includes combination of both states 5.4 and 5.5 which is added to the state. Also the state is reconditioned by applying: [16]

$$S_t = cf_t * S_t1 + I_t * \widetilde{S}_t - 1[16] \tag{5.6}$$

The reason why a LSTM model is used for this purpose is that a standalone RNN is challenging to train due its characteristics. As Back propagation is used for RNN's problems like vanishing-gradient occur. The gradient in general can be understand as a computed value through all time setps which in the end used to update parameters of the RNN. The vanasihing-gratdient over time results in information decay. By implementing an LSTM module this problem can be solved. [21]

Bidirectional LSTM

Bidirectional LSTMs are able to look in both directions past and future. This is achieved by processing the available data into both directions. Therefore those models make use of bidirectional layers. Those layers split up the used neurons into two directions. [22] This provides more information to the network as the model is now capable if storing the forward state as well as the backward state. Resulting in potentially more accurate results [23].

5.2.2 CNN

CNN's follow the concept of NN consisting of multiple layers. The scope of application for this kind of network reaches from computer vision problems to time-series forecast modelling. Whereas data provided for image classification is structured in multi dimensional arrays (matrices), data used for time-series forecasting is provided via one dimensional arrays.[24] A CNN provides different types of layers . Those layer types are called pooling layer (PL), fully connected (FC), Convolution layer (CL) and flatten layer (FL). The connection of those layers are demonstrated in figure 5.5.

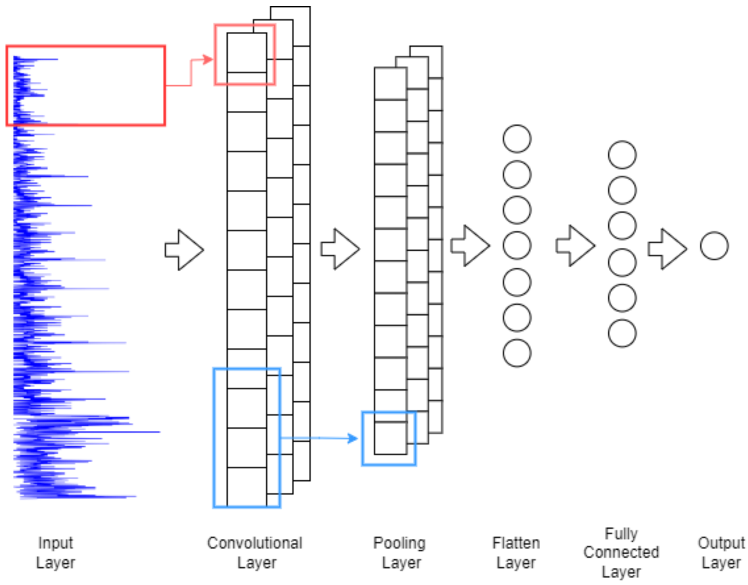

Figure 5.5: One Dimensional CNN Structure - [source:[25]]

CNN is based on convolution which is a linear operation that multiplies input data with convolution filters. Those filters which are also called kernels correlate to a set of weights [25]. The kernel values are created during the learning process and are optimized from the NN utilizing back propagation. Furthermore this layer is utilized to detect features within the given one dimensional array. Those features are stored within a feature map and are calculated by applying convolutions on the input data. One crucial parameter to detect proper features is the size of the kernel. The kernel size can be understand as a number of weights that are multiplied with the input data. After each multiplication the sequence is shifted along the input data. Each shift during this process produces one output which is stored in the feature map. The following example demonstrates how this process is done.[26]

```
Input Data: [4,7,10,43,20,10] e.g. number of bookings per day.
Kernel: [0.5,0.25,0.2] Kernel size = 3
1st Multiplication: 4*0.5 + 7*0.25 + 10*0.2 = 5.75
2nd Multiplication: 7*0.5 + 10*0.25 + 43*0.2 = 14.6
...
Output sequence [5.75, 14.6, ...]
```

The second operation that is used within the CNN is called activation function. This non-linear function is utilized to detect complex relationships between variables and are applied onto the feature map. As of today multiple functions like ReLU, Sigmoid and softmax can be used as activation functions [27].

The pooling layer as shown in figure 5.5 is deployed within a NN to diminish the size of feature maps. To reduce the size pooling operations like average pooling, max pooling or sum pooling can be applied. Applying one of those operations results in less computational efford.[28]

The activation function is also part of the fully connected layer. This layer applies the activation function onto the feature map and enables the model in combination with back-propagation to learn complex connections between features. Furthermore this layer operates on the already flatten feature map and outputs a 2d vector.[26]

The flatten layer transforms two dimensional input data into one dimensional input vectors. It's output is used to provide outputs to the fully connected layer. Over-fitting can be caused whenever all features are used in the flatten layer therefore a dropout layer can be set in place. This layer cancels out neurons during the training process of a NN which reduces the model's size.

5.3 Overfitting and Underfitting

One problem that can occur when utilizing NN for predictions is over-fitting or under-fitting of the training data. Both scenarios result in a poor performance of the trained model.

5.3.1 Overfitting

Overfitting describes the phenomenon that the model is not able to improve its problem solving capabilities after a certain period of training. There are multiple reasons for the occurrence of overfitting. One reason for example is a inaccurate or unbalanced training set. This leads to the fact that the NN produces wrong connections during its training. Whereas the results for the training set are accurate the problems occur during the validation phase because the model learned wrong characteristics. [29]

5.3.2 Underfitting

On the other hand underfitting arises when the model is not capable of identifying the traits of the training set and therefore struggles to achieve matching its target values. This results in a high loss values. Reasons for underfitting are caused by a lack of trainable parameters as well as a NN model with a simple architecture in terms of hidden layers.

In section 3.3 and 3.4 actions were taken to avoid both overfitting and underfitting.

5.4 Loss Function

The loss function is one crucial element as they evaluate the accuracy of the produced outputs from a NN. This is achieved by calculating the difference between the predicted value and the actual value provided by the test dataset. Supervised learning deals with two different problems which is either a classification problem i.e. is the animal on the picture a cat or with regression problems which deal with i.e. predicting future bookings. Both of those problems use different loss function.[30] As this section focuses on solving a regression problem a brief overview about available loss functions and their characteristics are given.

Function	Characteristics
Square loss	Sensitive to outliers (Model tends to focus on those outliers whereas accuracy for normal values decline)
Absolute loss	Outliers do not influence the model as severe as compared to square loss
Huber	Combination of square loss and absolute loss - Outliers do not influence the accuracy of results and learning from smaller errors can still be done in a efficient way
Log-cosh	Similar to Huber when it comes to its characteristics. Does not handle large errors well because the gradient tends to stay constant.
Quantile loss	Extends absolute loss and provides prediction intervals. Utilizing a punishment system for overestimated and underestimated samples.
ϵ-insensitive	Focuses on samples with large prediction errors

Table 5.1: Loss functions and their characteristics - [source:[30]]

By looking at the characteristics of the augmented data set shown in figure 3.2 it is clear to see that the dataset itself has got outliers repeating themselves every year. To avoid a strong focus on those peaks both models are initial trained utilizing the Adam loss function.

5.5 Optimize Function

To optimize a NN's parameters an optimizer function is required. This function updates parameters like weights based on the results provided by the loss function [31]. Since both NN's described in this section make use of backpropagation for their training a literature review was conducted to figure out which optimizer is keen to deliver the most accurate results. By inspecting the advantages and disadvantages proposed by these works [31][**?**][32] it turns out that the algorithm ADAM is a valid choice. This algorithm is characterized by achieving faster convergence compared to other algorithms. Furthermore Adam provides a decent performance for datasets with meager features.

5.6 Implementation

Both models follow the same workflow when it comes to their implementation. The workflow is highlighted in figure 5.6:

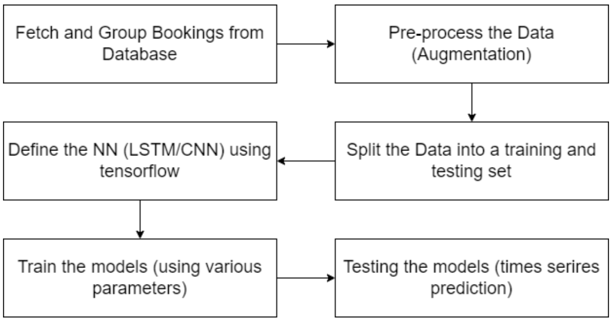

Figure 5.6: Workflow of the implementation - [source:[author]]

As each booking correlates to one entry within the database table the data those entries need to be grouped on a daily basis. Therefore the following sql query is executed:

```
import mysql.connector
import logging
import pandas as pd

def get_booking_data():

       sql = ("SELECT count(taskFrom_time) as bookings, date(taskFrom_time) "
           "from bookings_2 "
           "WHERE date(taskFrom_time) <= DATE('2023-05-01') and date(
    taskFrom_time) >=      DATE('2017-01-01') "
           "Group By date(taskFrom_time) order by date(taskFrom_time) asc")

    res = pd.read_sql(sql, connection)
    return res
```

Once the data is retrieved from the database it is directly converted to a `pandas dataframe`. As mentioned in chapter 3.4 data augmentation is necessary to compensate the lack of data during COVID19 3.4. Corresponding to the workflow described in figure 5.6 the data now needs to be separated into a training and test set. The training set is used to train the model whereas the test set is used to display how the trained model performs. Therefore the trained model is used to predict the values for timestamps used in the test set. By reviewing other scientific works that deal with time series forecasting [26],[25],[24],[21],[16],[20] the most accurate results are achieved by using ranges from 80% to 90% for training and depending on the range for training data a range of 20% to 10% for test data are recommended. The initial split used for both models correlates to 90% training data to 10% test data as visualized in 5.10 . Therefore the following code is applied to split the data:

```
training_end = pd.to_datetime('2022-10-31')
#total range 2017-01-01 - 2023-05-31 = 2342 days
train = df[:training_end]
# 2017-01-01 - 2022-10-31 = 2129 days ~ 90%
test = df[training_end:]
# 2022-10-31 - 2023-05-01 = 213 days ~ 10%
```

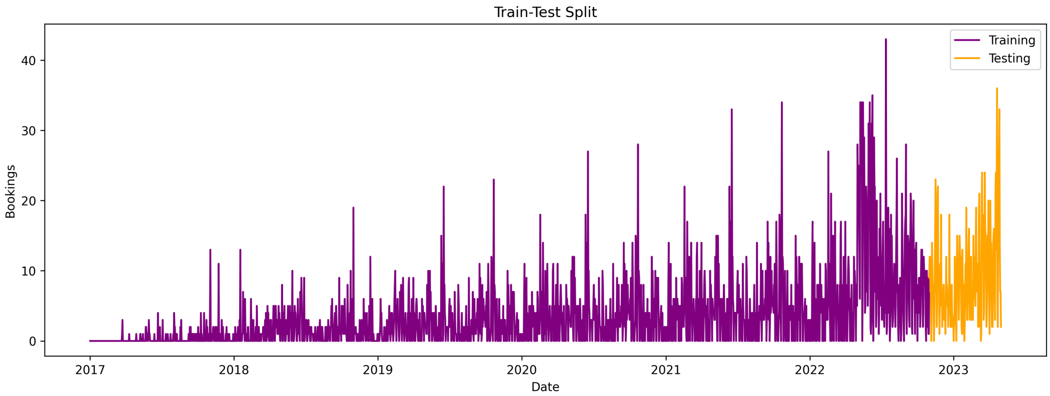

Figure 5.7: Visualized Training-Test split (90%-10%) - [source:[author]]

With the training and test data in place the next step is to implement both models (LSTM, CNN) by utilizing the `tensorflow` library. To reduce computation costs and to increase training efficiency the training data is further processed.

```
1 WINDOW = 20
2 bookings_training_90 = tf.data.Dataset.from_tensor_slices(train.values)
3 bookings_training_90 = bookings_training_90.window(WINDOW + 1, shift=1,
    drop_remainder=True)
4 bookings_training_90 = bookings_training_90.flat_map(lambda x: x.batch(WINDOW +
    1))
5 bookings_training_90 = bookings_training_90.map(lambda x: (x[:-1], x[-1]))
```

On line 2 the code above transforms the training data into a `TensorSliceDataset`. This data format grants access to tensorflows data API which supports the user to manipulate the data further. As this is time serires data the model itself is fed with data limited to certain ranges. The constant `WINDOW` indicates a range of 20 days which is used for training. The function `flat.map()` is now used to flatten the dataset. As `from_tensor_slices()` creates a single tensor for each entry in a window `flat.map()` combines those windows to a single tensor holding the windowed data. Line 5 prepares the data and splits each window into features and target values. As the learning process described in section **??** involves multiple inputs that are used to predict one output the `.map()` function prepares each windowed tensor by using the interval from `x to x-1` to predict `x-1`. Both models LSTM and CNN are trained with the prepared data explained in during this section.

5.6.1 Implementation of LSTM

The code itself required to setup a LSTM model only requires a few input parameters. Therefore it is crucial to understand the meaning behind the input parameters as well as how they can influence the training results of the model itself. The following code is used to initialize the model:

```
#define the model
lstm_booking_prediction_model = Sequential([
    Lambda(lambda x: tf.expand_dims(x, axis=-1), input_shape=[WINDOW]),
    Bidirectional(LSTM(128, activation='tanh',recurrent_activation='sigmoid' )),
    Dense(units=128, activation='relu'),
    Dropout(0.4),
    Dense(1)
  ])
```

When utilizing LSTM for time series predictions a certain input format for data is needed. Therefore the Lambda function on line 3 is required to reshape the dimension of the used input data. `Bidirectional()`[5] actually represents a wrapper for the actual layer used for this model. In this case its holding additional states and is used to created a Bi-LSTM model as described in 5.2.1.
`LSTM()`[5] contains the logic actual logic as described in 5.2.1. Furthermore the parameter `unit=128`[5] can be understand as the number of neurons used within this layer. Furthermore this model offers different kind of activation functions. By default the hyperbolic tangent (tanh)[33] is used. Whereas the `recurrent_activation` defines which functions are used for the actual gates within the module as described in 5.2.1. Whenever creating a stacked LSTM model, which means it makes use of at least two LSTM layers the paramater `return_sequences` must be set to true. Otherwise the layer's output results in a 2D tensor output which only provides information about the last timestep. This format cannot be passed on to the next LSTM layer.
`Dense()` is used to implement fully connected layers whereas `units` correlate to the amount of neurons used for this layer The last `dense(1)` layer is used to reduce the number of outputs to 1. The way this layer works is explained in section 5.2.2. Additional the model needs to be compiled. Therefore the following code is required:

```
#compile the model
lstm_booking_prediction_model.compile(
    loss=Huber(),
    optimizer=Adam(),
    metrics=['mae']
  )
```

The Parameter `loss` sets the loss function which is utilized to evaluate the models performance. The reason why `Huber` is used is explained in section 5.4. Futhermore the model also requires an optimizing algorithm. This algorithm is assigned by using the `optimizer` parameter. Due to the results of a literature review explained in section 5.5 Adam turned out to be the most promising candidate for this purpose. To observe the capability of learning of a given model the metric mean absolute error (mea) can be used.

The next step is to start the training of the model as demonstrated below:

```
#start to train the model
lstm_history = lstm_booking_prediction_model.fit(
    bookings_training_90,
    epochs=100,
    verbose=1,
    use_multiprocessing=True
  )
```

The function `fit()` requires parameters like the data used for training (`bookings_training_90`) as well as how many epochs are processed. One `epoch` covers the process of iterating through the entire training dataset and performing forward and backward propagation as well as updating the model's parameters based on the chosen optimization algorithm.
When training the model using the initial parameters described during this chapter the following results are achieved:

[5] `https://www.tensorflow.org/api_docs/python/tf/keras/`

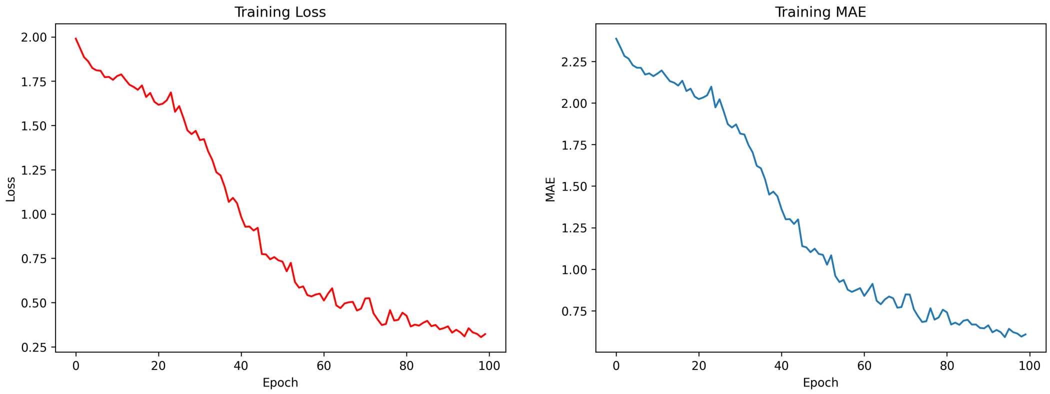

Figure 5.8: Decrease of MAE and Trainig loss for 100 epochs - [source:[author]]

The training loss in general indicates the performance of the model based on the training set. It is generated through the output of the applied loss function. Low values basically mean that the model is able to catch up patterns within the training set but could also indicate overfitting. This causes the model to just memorize the training set rather than learning actual patterns. Furthermore the loss indicates the amount of required epochs for the model to learn. Whenever the loss stalls or is starting to increase again the training should be stopped. The Mean Absolute Error (MAE) is another metric to observe the ability of the model to learn. The MAE is obtained by calculating the mean of actual values minus the predicted values. This result is divided by the number of observations.[34] Similar to the training loss a lower value indicates the performance of the model but the MAE needs to be interpreted in context with the range the predictions are made for. For example a MAE of 1 for predictions made in ranges between 1-2 is interpreted as poor performance, whereas a MAE of 100 for predictions made in a range between 100.000 and 1.000.000 indicates a proper result.

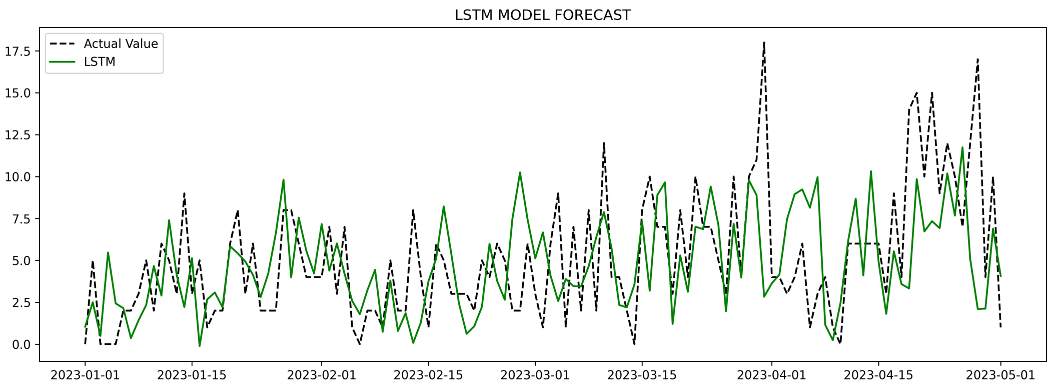

Figure 5.9: Predictions on Test set from 01-01-203 to 01-05-2023 - [source:[author]]

By looking at figure 5.11 its clear to see that the model's performance in terms of accuracy is insufficient. Investigating the training loss and MAE shows that the final MAE of around 0.5 is acceptable in context of the prediction range ranging from 1 to 17. This indicates that the model tends to overfitting as the final training loss is low as well. Section **??** investigates strageties on how to improve the model's performance.

5.6.2 Implementation of CNN

Similar to the implementation of LSTM the implementation of a CNN does not require much code. Furthermore some parts of the code use the same parameters. Those parameters are not explained again. To initialize a CNN model the following code is required:

```
#define the model
cnn_model = Sequential([
    Lambda(lambda x: tf.expand_dims(x, axis=-1), input_shape=[WINDOW]),
    Conv1D(filters=512, kernel_size=3, activation='relu'),
    Conv1D(filters=512, kernel_size=3, activation='relu'),
    GlobalAveragePooling1D(),
    Flatten(),
    Dropout(0.3),
    Dense(512, activation='relu'),
    Dropout(0.4),
    Dense(1)
])
```

`Conv1D()` is used to add one one dimensional CL to the model. Those Layers work as described in 5.2.2. The `filter` parameter is equivalent to the number of neurons this layer is going to use. Furthermore the kernel's size as explained in 5.2.2 is determined by the parameter `kernel_size`. The only difference left in comparison to the LSTM layer is the FL 5.2.2. This layer is added by utilizing the `Flatten()` parameter. When it comes to compiling and fitting the model the CNN makes use of `Huber()` for its loss function and `Adam()` for its optimization function contrary to the implementation of the LSTM model. Furthermore both models are initially using 100 epochs for their first training.

The first training results produced by the CNN model look the following:

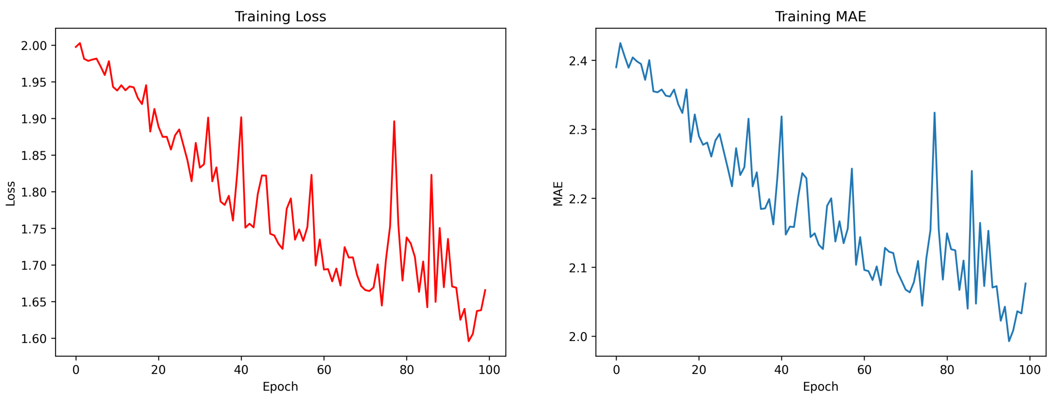

Figure 5.10: Decrease of MAE and Trainig Loss for 100 epochs - [source:[author]]

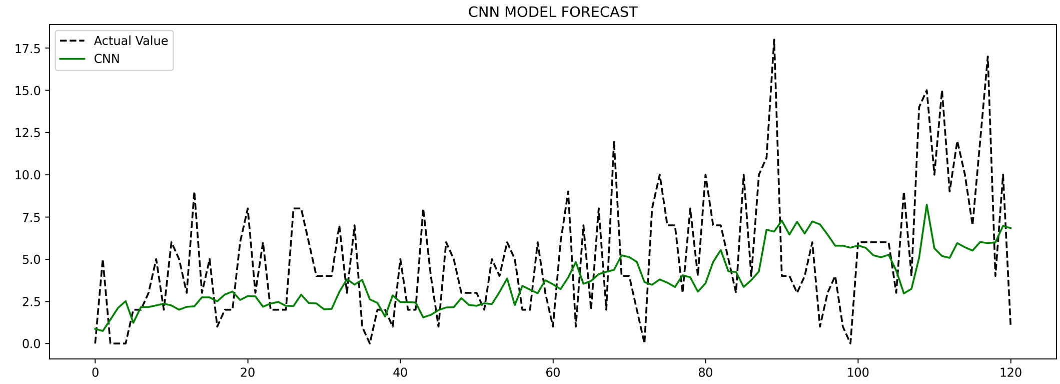

Figure 5.11: Predictions on Test set from 01-01-203 to 01-05-2023 - [source:[author]]

5.7 Improving the models

??

5.8 Model accuracy

Having a look at the model performance accuracy (comparing predictions of the model with already available data) , explain potential twerks that have been applied to the model itself to achieve a higher level of accuracy.

6 Implementation of Averages and Grouping

6.1 Technical Setup

Explain the basic setup and used technologies for used for the analytical web based dashbaord

6.2 Applied Statistical Models

explain which attributes also provide additional information that can be gathered from the dataset, which models were applied (algorithms)

6.3 Visualisation techniques

which plots etc (and why) are used to display the gathered information

Bibliography

[1] S. V. Mahadevkar, B. Khemani, S. Patil, K. Kotecha, D. R. Vora, A. Abraham, and L. A. Gabralla, “A review on machine learning styles in computer vision—techniques and future directions,” *IEEE Access*, vol. 10, pp. 107 293–107 329, 2022. 2

[2] K. Pahwa and N. Agarwal, “Stock market analysis using supervised machine learning,” in *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, 2019, pp. 197–200. 2

[3] A. GÃ©ron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow : concepts, tools, and techniques to build intelligent systems*, third edition. ed., 2022. 2

[4] A. Géron, “Praxiseinstieg machine learning mit scikit-learn und tensorflow : Konzepte, tools und techniken für intelligente systeme,” 2018. 2

[5] P. R. Gulhane and T. S. Pradeep Kumar, “Tensorflow based website click through rate (ctr) prediction using heat maps,” in *2018 International Conference on Recent Trends in Advance Computing (ICRTAC)*, 2018, pp. 97–102. 2

[6] A. Dehghan-Banadaki, T. Taufik, and A. Feliachi, “Big data analytics in a day-ahead electricity price forecasting using tensorflow in restructured power systems,” in *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2018, pp. 1065–1069. 2

[7] A. Wannes and S. A. Ghannouchi, “Kpi-based approach for business process improvement,” *Procedia Computer Science*, vol. 164, pp. 265–270, 2019, cENTERIS 2019 - International Conference on ENTERprise Information Systems / ProjMAN 2019 - International Conference on Project MANagement / HCist 2019 - International Conference on Health and Social Care Information Systems and Technologies, CENTERIS/ProjMAN/HCist 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050919322215 2

[8] P. R. Martins de Andrade and S. Sadaoui, “Improving business decision making based on kpi management system,” in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2017, pp. 1280–1285. 2

[9] S. Netessine and R. Shumsky, “Introduction to the theory and practice of yield management,” *INFORMS transactions on education*, vol. 3, no. 1, pp. 34–44, 2002. 7

[10] K. Donaghy, U. McMahon, and D. McDowell, “Yield management: an overview,” *International Journal of Hospitality Management*, vol. 14, no. 2, pp. 139–150, 1995. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0278431995000133 10

[11] L. Wang, J. Chen, W. Wang, R. Song, Z. Zhang, and G. Yang, “Review of time series traffic forecasting methods,” in *2022 4th International Conference on Control and Robotics (ICCR)*, 2022, pp. 1–5. 10

[12] M. A. Istiake Sunny, M. M. S. Maswood, and A. G. Alharbi, "Deep learning-based stock price prediction using lstm and bi-directional lstm model," in *2020 2nd Novel Intelligent and Leading Emerging Sciences Conference (NILES)*, 2020, pp. 87–92. 10

[13] C. Hou, J. Wu, B. Cao, and J. Fan, "A deep-learning prediction model for imbalanced time series data forecasting," *Big Data Mining and Analytics*, vol. 4, no. 4, pp. 266–278, 2021. 10

[14] J. Deng and P. Jirutitijaroen, "Short-term load forecasting using time series analysis: A case study for singapore," in *2010 IEEE Conference on Cybernetics and Intelligent Systems*, 2010, pp. 231–236. 10

[15] W. Ertel, "Grundkurs künstliche intelligenz : Eine praxisorientierte einführung," 2021. 10, 11, 12, 27

[16] J. Kumar, R. Goomer, and A. Singh, "Long short term memory recurrent neural network (lstm-rnn) based workload forecasting model for cloud datacenters," *Procedia Computer Science*, vol. 125, pp. 676–682, 01 2018. 11, 13, 17, 27

[17] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, "Activation functions in deep learning: A comprehensive survey and benchmark," 2022. 11

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, p. 1735–1780, nov 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735 12

[19] K. Moharm, M. Eltahan, and E. Elsaadany, "Wind speed forecast using lstm and bi-lstm algorithms over gabal el-zayt wind farm," in *2020 International Conference on Smart Grids and Energy Systems (SGES)*, 2020, pp. 922–927. 12

[20] M. A. Istiake Sunny, M. M. S. Maswood, and A. G. Alharbi, "Deep learning-based stock price prediction using lstm and bi-directional lstm model," in *2020 2nd Novel Intelligent and Leading Emerging Sciences Conference (NILES)*, 2020, pp. 87–92. 13, 17

[21] S. Bodapati, H. Bandarupally, and M. Trupthi, "Covid-19 time series forecasting of daily cases, deaths caused and recovered cases using long short term memory networks," in *2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA)*, 2020, pp. 525–530. 14, 17

[22] S. Yang, "Research on network behavior anomaly analysis based on bidirectional lstm," in *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, 2019, pp. 798–802. 14

[23] S. Prakash, A. S. Jalal, and P. Pathak, "Forecasting covid-19 pandemic using prophet, lstm, hybrid gru-lstm, cnn-lstm, bi-lstm and stacked-lstm for india," in *2023 6th International Conference on Information Systems and Computer Networks (ISCON)*, 2023, pp. 1–6. 14

[24] A. P. Wibawa, A. B. P. Utama, H. Elmunsyah, U. Pujianto, F. A. Dwiyanto, and L. Hernandez, "Time-series analysis with smoothed Convolutional Neural Network," *Journal of Big Data*, vol. 9, no. 1, p. 44, Apr. 2022. [Online]. Available: https://doi.org/10.1186/s40537-022-00599-y 14, 17

[25] I. Čavor and S. Djukanović, "Vehicle speed estimation from audio signals using 1d convolutional neural networks," in *2023 27th International Conference on Information Technology (IT)*, 2023, pp. 1–4. 14, 17, 27

[26] S. Guessoum, S. Belda, J. M. Ferrandiz, S. Modiri, S. Raut, S. Dhar, R. Heinkelmann, and H. Schuh, "The short-term prediction of length of day using 1d convolutional neural networks (1d cnn)," *Sensors*, vol. 22, no. 23, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/23/9517 15, 17

[27] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into Imaging*, vol. 9, no. 4, pp. 611–629, Aug. 2018. [Online]. Available: https://doi.org/10.1007/s13244-018-0639-9 15

[28] M. M. Taye, "Theoretical understanding of convolutional neural network: Concepts, architectures, applications, future directions," *Computation*, vol. 11, no. 3, 2023. [Online]. Available: https://www.mdpi.com/2079-3197/11/3/52 15

[29] H. Zhang, L. Zhang, and Y. Jiang, "Overfitting and underfitting analysis for deep learning based end-to-end communication systems," in *2019 11th International Conference on Wireless Communications and Signal Processing (WCSP)*, 2019, pp. 1–6. 15

[30] Q. Wang, Y. Ma, K. Zhao, and Y. Tian, "A Comprehensive Survey of Loss Functions in Machine Learning," *Annals of Data Science*, vol. 9, no. 2, pp. 187–212, Apr. 2022. [Online]. Available: https://doi.org/10.1007/s40745-020-00253-5 16, 28

[31] A. Omar, F. Mohamed, M. Mohammed, and B. Fouad, "The commonly used algorithms to optimize a neural network in supervised learning: Overview, and comparative study," in *2021 International Conference on Digital Age and Technological Advances for Sustainable Development (ICDATA)*, 2021, pp. 31–38. 16

[32] C. Desai, "Comparative analysis of optimizers in deep neural networks," 10 2020. 16

[33] A. Namin, K. Leboeuf, R. Muscedere, H. Wu, and M. Ahmadi, "Efficient hardware implementation of the hyperbolic tangent sigmoid function," 06 2009, pp. 2117 – 2120. 19

[34] W. Wang and Y. Lu, "Analysis of the mean absolute error (mae) and the root mean square error (rmse) in assessing rounding model," *IOP Conference Series: Materials Science and Engineering*, vol. 324, no. 1, p. 012049, mar 2018. [Online]. Available: https://dx.doi.org/10.1088/1757-899X/324/1/012049 20

List of Figures

List of Tables

Appendix

(Hier können Schaltpläne, Programme usw. eingefügt werden.)