Höhere Technishe
Bundes - Lehranstalt
Graz – Gösting (BULME)
Ableitung für Elektrotechnik

DIPLOMARBEIT

Visitor Prediction Modeling

For Shopping Centers

Thi Ly Tran 4AKIFT

ausgeführt im Schuljahr: 2022/2023

Betreut von: DI Mag. Dr. Gerald Senarclens de Grancy

Abgabevermerk:

Datum: 22.09.2023 übernommen von:

Statutory declaration

I hereby declare in oath that I have written this thesis independently and without outside help. I have not used any other aids than those specified, as well as passages taken from the sources used, either verbatim or in terms of content, have been marked as such. The work has not been submitted to any other examination board in the same or a similar form, nor has it been published.

Graz, 08.09.2023

Thi Ly Tran

Acknowledgements

I would like to express my sincere gratitude to INVENIUM Data Insights Gmbh for providing me with the opportunity to work on this project. I would also like to extend my thanks to my supervisors, Dr. Gerald Senarclens de Grancy, Dr. Florian Fuchs and Dr. Nguyen Thanh Binh for their invaluable guidance and feedback throughout the thesis. Lastly, I would like to thank my family and friends for their constant support and encouragement during the thesis work.

Abstract

This thesis demonstrates two main approaches of the time series forcasting problem, specifically the XGBoost model from machine learning and the SARIMAX model from statistical modeling, to forecast the number of visitors in some shopping centers in Austria.

To derive the predictive model, the dataset from the Donau Zentrum shopping center is utilized. This dataset, provided by Invenium Data Insights GmbH, includes historical visitor data since 8.2.2020, as well as weather data, and holiday data. Based on various comparisons between XGBoost and SARIMAX, the XGBoost model is selected as the preferred choice for visitor forecasting. The evaluation results indicate that the XGBoost model, trained on data without lockdowns and optimized, achieves a accuracy of 91% for visitor forecasting. Additionally, when training the XGBoost model on lockdown data, there is a decrease in accuracy. Nevertheless, it is important to note that the model still performs significantly well, with an accuracy of 85%. Therefore, this model is considered a contingency option in case of future lockdowns.

The model of Donau Zentrum is then applied to other shopping centers. The accuracy depends on specific center data, but it persists a prediction ranging from 84% to 91%. Common features that impact visitor count across all shopping centers include seasonal variations. Additionally, each shopping center has its own specific features. Interestingly, the influence of weather is relatively small and is observed in only two out of the five shopping centers.

The findings provide valuable insights for resource management and decision-making in the retail industry, enabling shopping center managers to make informed decisions based on accurate visitor forecasts.

Keywords Forecasting visitors in shopping centers · Machine learning · XGBoost · SARIMAX · Time Series Analysis

Kurzfassung

Diese Arbeit zeigt zwei Hauptansätze zur Prognose von Zeitreihenproblemen auf, insbesondere das XGBoost-Modell aus dem Machine Learning und das SARIMAX-Modell aus der statistischen Modellierung, um die Besucherzahlen in Einkaufszentren in Österreich vorherzusagen.

Um das Vorhersagemodell abzuleiten, wird der Datensatz des Donau Zentrums verwendet. Dieser Datensatz, bereitgestellt von Invenium Data Insights GmbH, enthält historische Besucherdaten seit dem 8.2.2020 sowie Wetterdaten und Feiertagsdaten. Basierend auf verschiedenen Vergleichen zwischen XGBoost und SARIMAX wird das XGBoost-Modell als bevorzugte Wahl für die Besucherprognose ausgewählt. Die Evaluierungsergebnisse zeigen, dass das XGBoost-Modell, das auf Daten ohne Lockdowns trainiert und optimiert wurde, eine hohe Genauigkeit von 91% bei der Vorhersage von Besuchern erreicht. Zusätzlich gibt es eine Verringerung der Genauigkeit, wenn das XGBoost-Modell auf Lockdown-Daten trainiert wird. Dennoch ist es wichtig anzumerken, dass das Modell immer noch eine recht gute Leistung mit einer Genauigkeit von 85% aufweist. Daher wird dieses Modell als Notfalloption für zukünftige Lockdowns betrachtet.

Das Modell des Donau Zentrums wird anschließend auf andere Einkaufszentren angewendet. Die Genauigkeit hängt von den spezifischen Einkaufscenter-Daten ab, liegt jedoch im Bereich von 84% bis 91%. Gemeinsame Merkmale, die sich auf die Besucherzahl in allen Einkaufszentren auswirken, sind Saisonalität. Darüber hinaus verfügt jedes Einkaufszentrum über seine eigenen spezifischen Merkmale. Interessanterweise ist der Einfluss des Wetters relativ gering und wird nur in zwei der fünf Einkaufszentren beobachtet.

Die Ergebnisse liefern wertvolle Einblicke für das Ressourcenmanagement und die Entscheidungsfindung in der Einzelhandelsbranche und ermöglichen es Management der Einkaufszentren, fundierte Entscheidungen auf der Grundlage genauer Besucherprognosen zu treffen.

Schlüsselwörter Vorhersage von Besuchern in Einkaufszentren · Machine Learning · XGBoost · SARIMAX · Zeitreihenanalyse

Contents

Chapter 1

Introduction

Due to the high usage of smartphones, it is nowadays possible to analyze anonymized movement streams of cell phone users through passively collected cellular signaling data. This makes it possible, for example, to determine the number of visitors to shopping malls or large events.

Accurately predicting the number of visitors in shopping centers has many advantages for managing resources and planning operations effectively. First, it helps schedule staff members in the right way, so there are enough people to serve customers during busy times and fewer staff members during quieter periods. This ensures good service and makes operations run smoothly. Secondly, it aids in maintaining optimal inventory levels and organizing maintenance schedules. Additionally, it enables strategic planning of marketing campaigns to attract customers at the right times. Finally, it helps keep the shopping center safe by identifying busy areas and assigning enough security personnel. Overall, accurate visitor prediction improves staff management, inventory control, maintenance planning, marketing effectiveness, and safety measures, leading to better operations, customer satisfaction, and profits.

The aim of this thesis is to develop a predictive model for forecasting the daily number of visitors in shopping centers. This task falls under the domain of time series forecasting. A time series represents a sequence of observations of a variable over time, in this case, the daily visitors. To address the time series forecasting problem, two main approaches are investigated: statistical modeling and machine learning.

Statistical methods include classical techniques such as autoregressive integrated moving average (ARIMA), exponential smoothing, and simple moving average. After experimenting these techniques, it was determined that the ARIMA archived the best result. Additionally, it was observed that the historical data exhibited weekly seasonality without trend. Consequently, the SARIMA model (Seasonal ARIMA) is applied. Moreover, exogenous variables were incorporated to enhance the model's accuracy. Therefore, seasonal autoregressive integrated moving average with exogenous variables (SARIMAX) was chosen as the representative model for statistical modeling.

Machine learning is a branch of artificial intelligence that focuses on developing algorithms that can learn patterns and make predictions from data. To achieve the goal of forecasting the visitor count, the supervised learning approach in machine learning is suitable. In supervised learning, models are trained using labeled examples to understand the relationship between input features and target outputs. In this case, the target variable is a continuous numerical value, specifically the number of visitors, making it a regression problem. For regression tasks, where the goal is to predict continuous values, the eXtreme Gradient Boosting (XGBoost) algorithm is a powerful and widely-used machine learning technique. XGBoost offers several advantages, such as its ability to handle complex relationships, scalability, and high performance. Therefore, XGBoost was selected as the representative model for machine learning in this project.

The original visitor data includes nine shopping centers in Austria. However, due to the project's objectives and time limitations, this thesis concentrates on sufficient datasets from five selected shopping centers for model development. Specifically, the chosen shopping centers are Shopping City Seiersberg in Graz, PlusCity in Pasching, Donau Zentrum in Vienna, Lugnercity in Vienna, and Shopping Center Süd in Vienna.

To derive the predictive model, the dataset from the Donau Zentrum shopping center is utilized. The evaluation results show that the XGBoost model, after optimization, achieves a root mean squared error (RMSE) value of 6080 on the test set compared to the baseline model with default parameters, which has an RMSE value of 7952. The mean absolute percentage error (MAPE) value on test set calculated without holidays is 0.09, indicating a model accuracy of 91%. Based on various comparisons between XGBoost and SARIMAX, the XGBoost model is selected as the preferred choice for visitor forecasting.

Furthermore, when training the XGBoost model using data without lockdowns, the model has the lower RMSE values (6080 versus 10690) on the test set compared to using data with lockdowns. However, it is worth noting that the XGBoost model trained on lockdown data still performs reasonably well, with a MAPE of 0.15, indicating a model accuracy of 85%. Therefore, this model is considered as a contingency option in case of future lockdowns.

The model of Donau Zentrum is then applied to other shopping centers. The accuracy depends on specific center data, but it persists a prediction ranging from 85% to 91%. The analysis of feature importance reveals that certain time-related factors have a significant impact on the number of visitors, such as holidays, December, Christmas Eve/New Year's Eve, Fridays and Saturdays. The influence of weather is relatively small and is observed in only two out of the five shopping centers.

The organization of this thesis is as follows: Chapter 2 provides foundational knowledge for the subsequent chapters. Chapter 3 focuses on the processing and exploration of the dataset used in this thesis. The prediction models are presented in Chapter 4 where XGBoost and SARIMAX models are investigated in Sections 4.1 and 4.2, respectively, and their comparison in given in Section 4.3. Chapter 5 presents the visualization of the results obtained from the prediction models. Finally, Chapter 6 concludes the thesis, discussing the limitations of the models considered in this study and providing an outlook for future investigations.

The digital version of this thesis is available at this address:
https://github.com/LyBaoMinEmi/HTL_Thesis

Chapter 2

Preliminary

In this chapter, I introduce some techniques that are used for both models. Special techniques using for each model will be demonstrated in chapter 4.

2.1 Split dataset and cross-validation

- **Split dataset correctly**
  When fitting a machine learning algorithm to a dataset, it is common practice to divide the data into three distinct parts for different purposes [28]:
  - Training Set: This subset of the data is used to train the model.
  - Validation Set: This subset is employed to optimize model parameters.
  - Test Set: The test set serves as an unbiased measure of the final model's performance.

  The following diagram provides a visual explanation of these three different types of datasets:

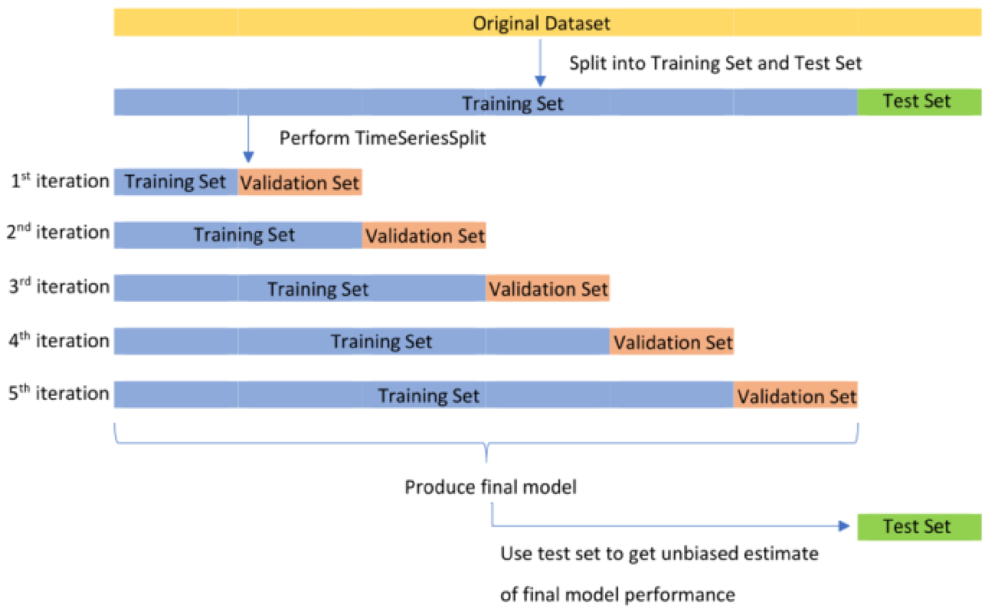


Figure 2.1: Split data using TimeSeriesSplit

- **Choosing an appropriate split ratio**
  As a general guideline, a 70:30 or 80:20 ratio is often used for small to medium-sized datasets, while a 90:10 or 85:15 ratio is more appropriate for larger datasets. However, there is no fixed rule for determining the ratio. In this thesis, considering that the dataset has 14 months and the model needs to be trained for at least 12 months, a split ratio of 85:15 for the training and test sets is reasonable.

```python
ratio = 0.85
train_size = int(len(df_xgb) * ratio)
train, test = df_xgb[:train_size], df_xgb[train_size:]
```

- **TimeSeriesSplit**
  Cross validation is an important technique in machine learning used to evaluate a model's performance on new, unseen data. It involves splitting the data into multiple subsets, training the model on one subset, and then evaluating it on another. This process is repeated several times, with each subset serving as a training and test set. The results from each iteration are averaged to provide an overall measure of the model's performance. Cross-validation is also commonly used to determine the optimal hyperparameters for a machine learning model [19].

  In the case of Time Series problems, the data is collected in chronological order, and thus the validation of the model must be done sequentially. This means that I cannot use traditional cross-validation techniques like K-Fold cross-validation because it assumes that the data is independent and identically distributed. Therefore, I need a specialized technique for Time Series data known as Time Series Split.

  The Time Series Split is a cross-validation technique specifically designed for time series data. It allows for the division of the data into training and validation sets in a sequential approach. This means that the model is trained on past data, then used to predict future values, and the accuracy of these predictions is evaluated on the corresponding actual data points. This method ensures that the validation set is always in the future compared to the training set [24]. The figure in 2.1 provides a visual representation of this approach.

- **Number split**
  It is important to ensure that the test set and validation set have the same size to make a fair comparison of the errors. When using TimeSeriesSplit with n_splits = 5, the dataset is split into 5 folds. For each fold, the data is divided into two parts: the training set and the validation set. The following code demonstrates how to determine the number of splits for TimeSeriesSplit and apply it to the dataset:

```python
from sklearn.model_selection import TimeSeriesSplit
n_splits = math.ceil(ratio/(1-ratio)-1)
tscv = TimeSeriesSplit(n_splits=n_splits)
```

2.2 Autocorrelation

Both autocorrelation function (ACF) and partial autocorrelation function (PACF) can provide valuable insights into the behaviour of time series data [23].

The ACF calculates the correlation between the time series and its own lagged values. It quantifies the linear relationship between the current observation and its past observations at different time lags.

On the other hand, the PACF is similar to the ACF but focuses on capturing the direct correlation between the current observation and its past observations, while eliminating the influence of intermediate lags. It helps to identify the direct relationship between the current observation and its past observations without the influence of other intermediate lags.

The following code shows how to plot the ACF and PACF:

```python
import matplotlib.pyplot as plt
import statsmodels.api as sm
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(20, 8))
sm.graphics.tsa.plot_acf(df[['visitors']], lags=30, ax=ax1)
sm.graphics.tsa.plot_pacf(df[['visitors']], lags=30, ax=ax2)
plt.show()
```

2.3 Feature selection

Feature selection is the process of choosing a subset of important features from a larger set of features in a dataset. The goal of feature selection is to identify the most informative and discriminative features that have a significant impact on the predictive ability of a machine learning model [6]. By selecting relevant features, feature selection can offer several advantages, including improved model performance, reduced overfitting, enhanced interpretability, and decreased computational complexity.

There are many techniques for selecting the optimal features for a machine learning model. In this thesis, I have experimented with different techniques and have found that forward selection is effective for the problem.

Forward selection is a wrapper method technique that involves iteratively adding features to the model. This method can be computationally expensive. Wrapper methods can be illustrated with the following graphic:

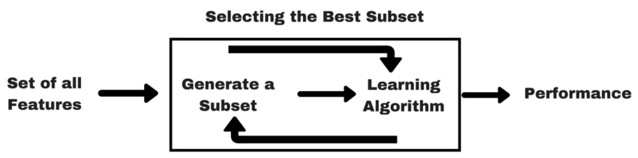


Figure 2.2: Wrapper method, see [6]

The forward selection process starts with an empty feature set, and in each iteration, the algorithm evaluates the model's performance by adding one feature at a time. The best-performing feature is selected and added to the feature set, and the process continues until the desired number of features is reached or no further improvement is observed.

2.4 Evaluation metrics

To build and deploy a generalized model, it is necessary to evaluate the model on different metrics that assist in optimizing performance, fine-tune it, and obtain a better result. Here are some evaluation metrics used for this regression problem, as described in [2]:

- **Mean absolute error (MAE)**
  mean absolute error (MAE) is the average of the absolute differences between the predicted and actual values. It measures the average magnitude of the errors in a set of predictions, without considering their direction. Mathematically, MAE can be expressed as follows:

  $$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

  where y_i is the actual value of the i-th sample, $\hat{y}_i$ is the predicted value of the i-th sample, n is the sample size.

  ```
  from sklearn.metrics import mean_absolute_error
  print("MAE",mean_absolute_error(y_test,y_pred))
  ```

- **Root mean square error (RMSE)**
  RMSE is the square root of the average of the squared differences between the predicted and actual values. It measures the standard deviation of the errors in a set of predictions, and gives more weight to large errors. The formula for RMSE is:

  $$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

  ```
  from sklearn.metrics import mean_squared_error
  print("RMSE",mean_squared_error(y_test,y_pred, squared = False))
  ```

- **R squared (R2)**
  R squared (R2) is the one minus sum of squares of residual divided by the total sum of squares. It ranges from 0 to 1, with higher values indicating better fit. R2 provides a measure of the proportion of variance explained by the model, and can help compare the performance of different models. The formula for R2 is:

  $$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum\limits_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum\limits_{i=1}^{n}(y_i - \bar{y})^2}$$

  where $\bar{y}$ is the mean of y_i values.

  ```
  from sklearn.metrics import r2_score
  print("R2", r2_score(y_test,y_pred))
  ```

- **Mean absolute percentage error (MAPE)**
  MAPE is the average percentage difference between the predicted values and

the actual values. The MAPE frequently employed as a loss function in regression problems and for evaluating models due to its intuitive interpretation regarding relative error [27]. The formula for calculating MAPE is as follows:

$$\text{MAPE} = \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right|$$

The model makes predictions for all days, including holidays when the shopping centers are closed. Consequently, the actual values on holidays are equal to 0, resulting in an infinite value for MAPE. To address this issue, the MAPE used in this thesis is calculated only for days that are not holidays.

```
import numpy as np
def mape_without_holidays(true,pred):
  true = np.array(true)
  pred = np.array(pred)
  non_zero_indices = np.nonzero(true)
  true_list = true[non_zero_indices]
  pred_list = pred[non_zero_indices]
  mape = round(mean_absolute_percentage_error(true_list,pred_list)
      ,2)
  return mape
print('MAPE* ',mape_without_holidays(y_test, y_pred))
```

The main metric used for regression problems depends on the specific problem and its requirements. However, in general, RMSE and R2 are more commonly used than MAE and MAPE, because they provide more information about the quality of the predictions and the goodness of fit of the model. RMSE is often used as the main metric, as it is more sensitive to large errors, which are usually more important in practical applications. In this thesis, the metric RMSE is used as the main metric to evaluate the goodness performance of the model.

Chapter 3

Data processing and exploring

In this study, I utilize three datasets: historical visitor data of shopping centers, weather data related to the areas where each shopping center is located and data containing the dates of public and school holidays. In this chapter, I will process, explore the data, merge these datasets together and then extract some relevant features before proceeding to train the model.

The original visitor data includes nine shopping centers in Austria. However, due to the project's objectives and time limitations, this thesis concentrates on sufficient datasets from five selected shopping centers for model development. Specifically, the chosen shopping centers are Shopping City Seiersberg in Graz, PlusCity in Pasching, Donau Zentrum in Vienna, Lugnercity in Vienna, and Shopping City Süd in Vienna. From now until chapter 6, I exclusively utilize the data of Donau Zentrum without explicitly mentioning it. In chapter 6, I will discuss the results in more detail for different shopping centers.

Although our primary focus is predicting the number of visitors on non-holidays, it's essential to include holiday data in our model. This is because even on holidays when visitor counts are zero, valuable seasonal patterns and temporal relationships may impact adjacent days. Therefore, retaining holiday data ensures the continuity and realism of our dataset, ultimately leading to more accurate predictions.

3.1 Visitor data

The received historical visitor data ranges from 02.11.2019 to 01.03.2023. During this period, there are four lockdown periods across Austria and an additional one in Vienna, as documented in [3]:

- 1. Lockdown from 16.03.2020 to 30.04.2020
- 2. Lockdown from 17.11.2020 to 06.12.2020
- 3. Lockdown from 26.12.2020 to 07.02.2021
- 4. Lockdown from 22.11.2021 to 11.12.2021
- Vienna Lockdown from 01.04.2021 to 03.05.2021

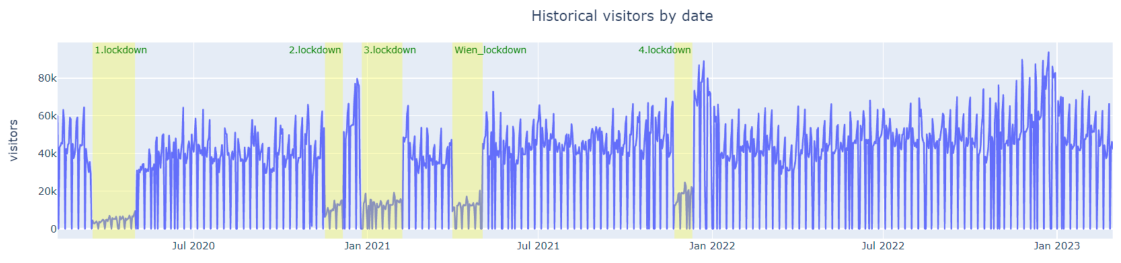


Figure 3.1: Historical visitor data

Due to the unique nature of visitor number during lockdown periods, I will exclude them from the subsequent data processing to ensure the stability of the model. The data processing will focus on the period from 13.12.2021 to 01.02.2023. However, I will later analyze the data including the lockdown periods separately.

The following information represents the descriptive statistics of the number of visitors:

	Visitors
Mean	51790
Median	48800
Min	26900
Max	93800

Table 3.1: Descriptive statistics of the number of visitors calculated without holidays

The visitor count reached its minimum on 24.12.2022 and reached its maximum on 23.12.2022.

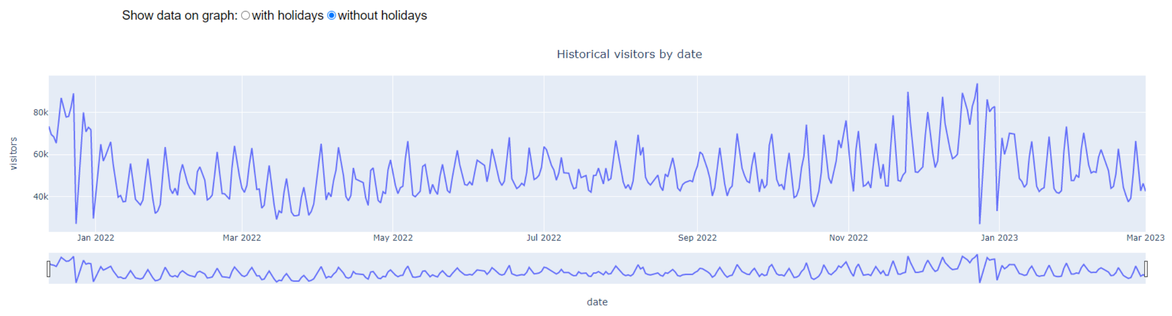

Show data on graph: ○ with holidays ● without holidays

Figure 3.2: Visitor data excluding holidays

As observed in Figure 3.2, the visitor data exhibits a significant drop on 24.12 and 31.12. Consequently, the feature '24,31_12' is considered important for predicting the number of visitors.

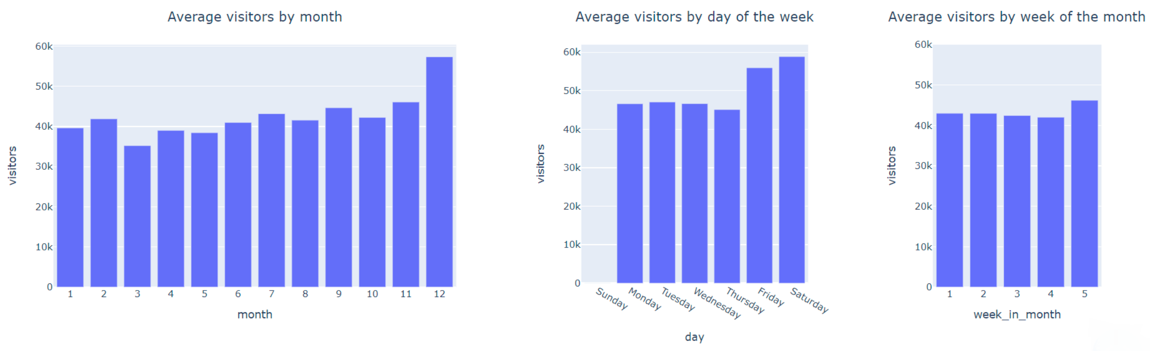


Figure 3.3: Average visitors by month, by day of the week and by week of the month

Based on Figure 3.3, it is observed that visitors tend to visit shopping centers more often in December, and less often in March. The average number of visitors by day of the week indicates a higher footfall on Fridays and Saturdays. Additionally, it is evident that visitors tend to visit more frequently towards the end of the month. Hence, the features 'month_12, month_3, Friday, Saturday, week_of_month_5' hold notable importance in predicting the number of visitors.

3.2 Weather data

Incorporating weather information into machine learning models has been shown to improve their predictive performance [17].

To ensure data consistency, the weather data is filtered out starting from 02.11. 2019, which corresponds to the starting date of the visitor data. This is done because the weather data cover a broader date range compared to the visitor data. However, there is a gap in the weather data from 23.01.2020 to 31.01.2020, which cannot be filled or recovered. As a result, both the weather data and visitor data are aligned starting from 01.02.2020. In addition, there are 3 missing days in the weather data: 23.05.2020, 12.09.2020 and 13.09.2020. To address this, the data from the closest available day is used for these missing days. The weather data for each shopping center is collected from its nearest area. More precisely, the weather data for Shopping City Seiersberg is obtained from Graz, PlusCity from Linz. Regrettably, there is no weather data available specifically for Vienna, hence the weather data from the Eisenstadt area is used for shopping centers in Vienna.

In this thesis, the weather factors that have the potential to impact the number of visitors include temperature, wind speed, and weather description. These factors are specifically taken into account within the time range of opening hours in shopping centers, from 8 a.m. to 9 p.m. The maximum wind speed during the opening hours is 14 km/h, which corresponds to wind force 3, described as a gentle breeze where leaves and small twigs are in constant motion, and light flags are extended. This wind speed does not significantly affect the travel of visitors. However, 'wind' will be considered as a feature when the wind speed exceeds 20 km/h, corresponding to wind force 4, which is described as "raises dust and loose paper, small branches moved" [26]. Based on the experiment conducted, it was found that the weather factors have a minimal impact on the number of visitors. Consequently, the weather factors are refined to provide a clear representation, using the following transformations:

- The 'temp' feature is derived from the maximum temperature during the opening hours. It is transformed into a binary value. A value of 0 indicates very cold or very hot conditions, specifically when the maximum temperature falls below 0°C or exceeds 28°C. Otherwise, a value of 1 is assigned.

- The 'weather' feature is also binary. A value of 0 represents nice weather conditions, specifically when the weather description indicates 'clear sky' or 'few clouds'. Otherwise, a value of 1 is assigned.

3.3 Holiday data

The API of the public holidays are obtained from the website:
https://date.nager.at/api/v3/PublicHolidays/2023/AT

3.4 Merge all data and adding some features

The multiple data sets are then merged together to form a dataframe. The next step involves adding potential features.

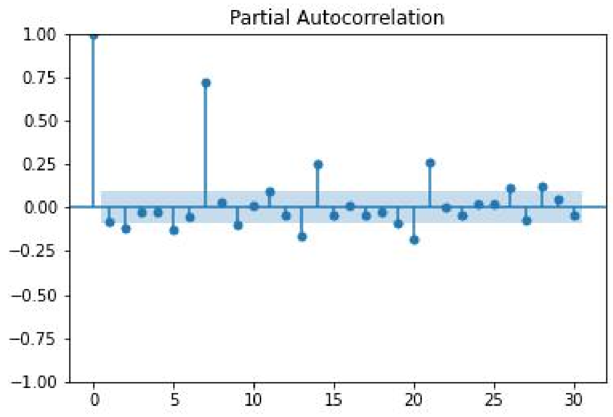


Figure 3.4: Number of daily visitors partial autocorrelation plot

The x-axis represents the lag values, where 1 corresponds to the previous day, 2 represents two days ago, and so on. The y-axis displays the corresponding partial autocorrelation values. From the plot, it is evident that the number of visitors from 7 days ago (lag 7) has a significant influence on the current number of visitors. Consequently, the feature '7_pre_day', representing the number of visitors from 7 days prior, is included in the set of features for predicting the visitor count. To make this adjustment, I need to match the weather data and visitor data, starting from 08.02.2020.

It has been noticed that the number of visitors fluctuates around previous holidays and after holidays, particularly before or after longer breaks. As a result, the following features have been incorporated to predict the visitor count: 'pre_holiday', 'post_holiday', 'pre_long_holiday', 'post_long_holiday', and 'pre_post'.

Feature Name	Description
Monday	Represents Monday as 1, otherwise 0
...	...
Sunday	Represents Sunday as 1, otherwise 0
month_i	Represents the i-th month of the year as 1, otherwise 0 (i = 1..12)
week_in_month_i	Represents the i-th week of the month as 1, otherwise 0 (i = 1..5)
holidays	Represents public holiday or Sunday as 1, otherwise 0
pre_holiday	Represents the previous day of holiday as 1, otherwise 0
post_holiday	Represents the post day of holiday as 1, otherwise 0
pre_post	Represents either pre_holiday or post_holiday as 1, otherwise 0
pre_long_holiday	Represents a day before a long holiday period as 1, otherwise 0
post_long_holiday	Represents a day after a long holiday period as 1, otherwise 0
7_pre_day	Represents the number of visitors from 7 days ago
lockdown	Represents day in lockdown as 1, otherwise 0
school_holidays	Represents day in school holidays as 1, otherwise 0
24/31_12	Represents the date December 24th or 31st as 1, otherwise 0
weather	Represents nice weather as 0 and other weather conditions as 1
temp	Represents hot ($\geq$ 28°C) or cold temperature ($\leq$ 0°C) as 0, otherwise 1

Table 3.2: Explaination of features using to train models

Chapter 4

Prediction Models

In this chapter, I will address the problem of forecasting visitors to shopping centers by applying two effective models: XGBoost and SARIMAX. For each model, I will provide an overview of its theory and explain how it can be applied to my specific problem. I will then evaluate the results obtained from each model. Following that, I will compare the performance of the two models in order to determine which one is better suited for forecasting visitors to shopping centers in the future.

4.1 XGBoost Model

4.1.1 Model description

XGBoost (Extreme Gradient Boosting) is a popular and powerful machine learning algorithm used for regression and classification tasks. XGBoost uses an ensemble of decision trees and gradient boosting to make predictions. It is very popular in data science and has won many competitions in machine learning [4].
Here are some of the advantages of XGBoost [14]:

- Speed and scalability: XGBoost is optimized for speed and scalability, making it a good choice for large datasets.
- Feature importance: XGBoost provides a measure of feature importance, allowing to identify which features are the most important for making predictions.
- Parallel processing: XGBoost supports parallel processing, making it faster and more efficient than other boosting algorithms.
- Regularization: XGBoost includes several regularization techniques such as L1, L2, and elastic net regularization to prevent overfitting and improve generalization performance.

4.1.2 Baseline

Hyperparameters are the configuration settings of a model that can be adjusted to enhance its performance. Finding a baseline in XGBoost means training a model without any hyperparameter tuning or feature engineering to establish a benchmark for the model's performance [10]. This benchmark can be used to evaluate the effectiveness of any improvements made to the model, such as tuning hyperparameters or adding new features.

Index	Metrics	Train sets	Validation sets	Test set
0	RMSE	33	7179	7952
1	MAE	24	5096	4825
2	R2	1.0	0.88	0.89

Table 4.1: Baseline for XGBoost

The table 4.1 shows that the model has quite good results on the validation sets and test set, but it is overfitting with default parameters. I will compare these results to the results after using feature selection and tuning hyperparameters.

4.1.3 Feature selection

I use SequentialFeatureSelector from sklearn.feature_selection to perform forward selection. To work with forward selection, the number of features should be selected first. After experimenting with many numbers of features, I have determined that there are 4 most important features and the best number of features is around 10 in total of 35 features. To determine the optimal number of features precisely, the method GridSearchCV will be applied to choose the best number of features in the range of 4 and 15. The following code explains this process:

```python
from sklearn.pipeline import Pipeline
from sklearn.feature_selection import SequentialFeatureSelector
from sklearn.model_selection import GridSearchCV

# Define the pipeline
pipeline = Pipeline([
('selector', SequentialFeatureSelector(XGBRegressor(),cv=tscv,
    scoring='neg_root_mean_squared_error', n_jobs=-1)),
('xgb', XGBRegressor())])

# Define the parameter grid
param_grid = {'selector__n_features_to_select': range(4, 15)}

# Perform the grid search
grid_search = GridSearchCV(pipeline, param_grid, cv=tscv, scoring='
    neg_root_mean_squared_error',return_train_score=True)
grid_search.fit(X_train, y_train)

# Print the best number of features
print("Best number of features:", grid_search.best_params_['
    selector__n_features_to_select'])
# Get the selected features
selector = grid_search.best_estimator_.named_steps['selector']
best_feature_imp = list(X_train.columns[selector.get_support()])
print("Selected features:", best_feature_imp)
```

After running that code, I will have these results:

```
Best number of features: 9
Selected features: ['Friday', 'Saturday', 'holidays', 'post_holiday', '
    school_holidays', '24/31_12', 'month_3', 'month_12', '
    week_in_month_4']
```

After identifying the list of the most important features, I can obtain the percentage of importance for each feature through the 'feature_importances_' attribute of XGBoost as following code:

```python
import xgboost as xgb
from xgboost import XGBRegressor
xgbr = xgb.XGBRegressor(**params)
xgbr.fit(X_train[best_feature_imp], y_train)
print('Feature importance:',xgbr.feature_importances_)
```

To determine the positive or negative influence of each feature on the target variable, I can use the SHapley Additive exPlanations (SHAP) framework as demonstrated in [20] or a simpler approach like the correlation method . The following code shows how to implement the correlation method:

```python
correlations = X_train[best_feature_imp].corrwith(y_train)
```

I can visualize the feature importance as shown in the following figure 4.1. This figure is a combination of two figures: one that displays the importance of each feature as a percentage, and another that indicates the positive or negative influence of each feature on the target variable. For example, I can see that the feature 'month_12' has a 2.7% effect on visitor count, and it has a positive influence on the visitor count.

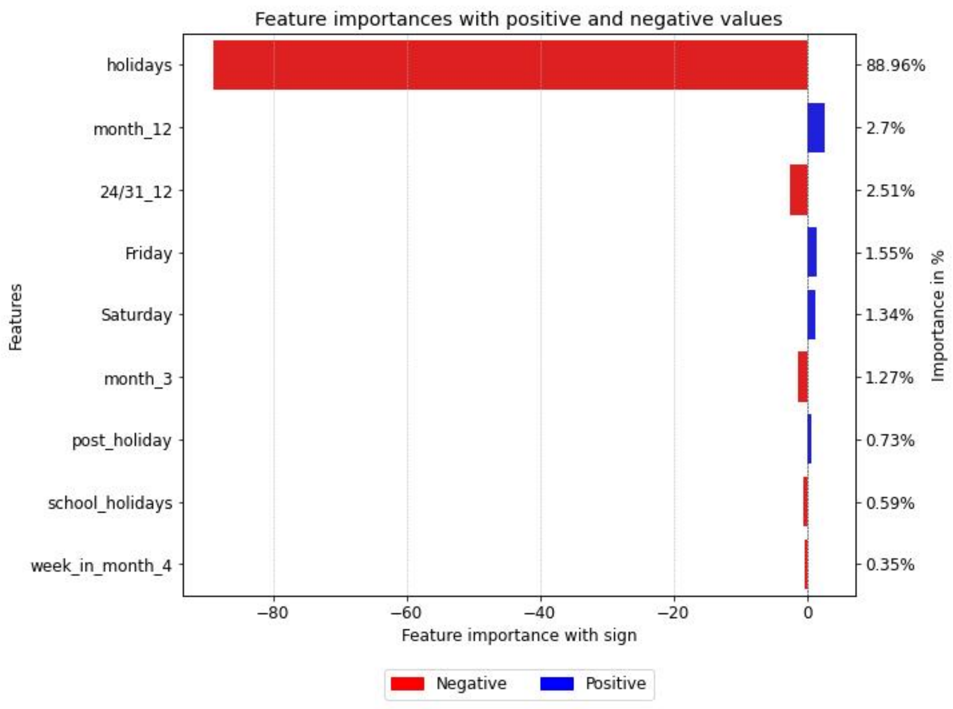


Figure 4.1: Feature importances

The figure 4.1 shows that the 'holidays' feature is the most important, as it has the highest feature importance score. This is clearly because the number of visitors to shopping centers on holidays is 0 and significantly lower than on regular days.

4.1.4 Tuning hyperparameters

Tuning parameters in XGBoost is crucial to achieve the best performance of the model. By adjusting the hyperparameters, I can improve the accuracy of the model, reduce overfitting, and avoid underfitting. Overfitting happens when the model learns the training data too well, and it performs poorly on the test data. Underfitting happens when the model is too simple and cannot capture the complexity of the data, which also results in poor performance on the test data. Therefore, the goal of parameter tuning is to strike a balance between overfitting and underfitting to achieve the best model performance. The figure below illustrates the issue of overfitting/underfitting in machine learning, as shown in [21]:

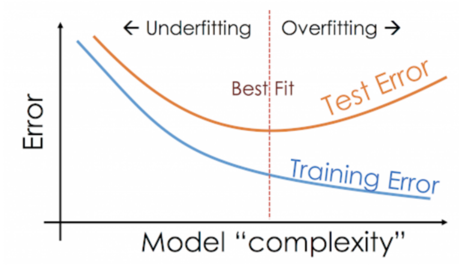


Figure 4.2: Overfitting vs. underfitting in machine learning

In this thesis, I will first apply the technique of 'early stopping' to determine the optimal value of the parameter 'n_estimators'. Afterwards, I will use the approach of GridSearchCV to identify the best combination of XGBoost hyperparameters. This approach will help reduce the execution time of GridSearchCV.

4.1.4.1 Early Stopping

Early stopping is a method to stop the training process when the model starts to overfit the data. Instead of training the model for a fixed number of iterations, early stopping monitors the model's performance on a validation set and stops training when the performance on the validation set stops improving. Early stopping can save computation time and avoid overfitting [11]. Here is a code snippet to implement the technique 'early stopping':

```python
def early_stop(X_train_fold,y_train_fold,X_val_fold,y_val_fold):
  model = XGBRegressor()
  eval_set = [(X_train_fold[best_feature_imp],y_train_fold),
              (X_val_fold[best_feature_imp], y_val_fold)]
  model.fit(X_train_fold[best_feature_imp],y_train_fold,
      early_stopping_rounds=1, eval_metric="rmse", eval_set=eval_set)
  best_ntree = model.best_ntree_limit
  print("\n")
  print("Number of best tree limit: %d" % best_ntree)
  return best_ntree
```

The following figure 4.3 shows an example of using early stopping to avoid overfitting:

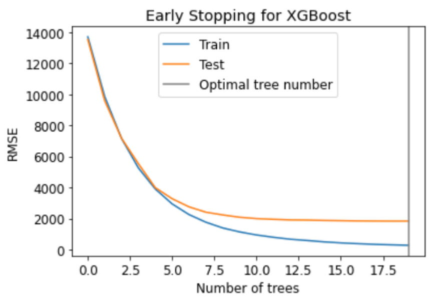


Figure 4.3: Early Stopping at the optimal tree number of 19

The model stops training at the optimal tree number of 19 when the training error continues to decrease, but the test error doesn't improve, indicating that further training is unlikely to result in better performance.

4.1.4.2 GridSearchCv

GridSearchCV is an approach, which exhaustively searches over a range of hyperparameters to find the combination that gives the best performance. GridSearchCV evaluates all possible combinations of parameter values, which can be time-consuming, but it guarantees that the best hyperparameters will be found [22]. In this thesis, I try to tune the most common hyperparameters of XGBoost to help prevent overfitting, improve model performance, and optimize training time, as mentioned in [16]:

- max_depth: Controls the maximum depth per tree. Increasing depth can enhance performance but also raise complexity and overfitting risks. Must be an integer greater than 0. Default: 6.
- learning_rate: Controls the step size during model optimization. A lower learning rate leads to slower computation but can improve the chances of reaching the best solution. Must be between 0 and 1. Default: 0.3.
- n_estimators: Determines the number of trees in the ensemble. This parameter affects the model's complexity and performance. Must be a positive integer. Default: 100.
- colsample_bytree: Randomly selects a fraction of columns for each tree, potentially reducing overfitting. Must be between 0 and 1. Default: 1.
- subsample: Determines the fraction of observations to be sampled for each tree. Lower values help prevent overfitting but may increase the risk of underfitting. Must be between 0 and 1. Default: 1.
- min_child_weight: Specifies the minimum sum of instance weight required in a child or leaf. Higher values make the model more conservative and can prevent overfitting. Can be any integer. Default: 1.
- lambda: Applies L2 regularization on the weights (Ridge Regression), which helps reduce overfitting. Can be any integer. Default: 1.

When using GridSearchCV, trying out all possible combinations of hyperparameter values can take a lot of time, especially with cross-validation. It is therefore a good practice to experiment with different subsets of hyperparameter values to determine a possible range that can be used for the grid search. It saves time and makes the optimization process more efficient.
I determined the optimal value of 'n_estimators' to be 18 after applying the early stopping technique. I will set 'n_estimators' to 18 ($\pm$3) to find the best combination with other hyperparameters. After experimenting with various values, I identified the possible range of hyperparameter values as shown in the following code:

```
arr=np.arange(filtered_best_tree -3,filtered_best_tree+3)
params = {'max_depth': [2,3,4],
        'learning_rate': [0.6,0.7,0.8],
        'n_estimators': arr,
        'colsample_bytree': [1],
        'subsample':[0.7,0.8],
        'min_child_weight': [1,2,3],
        'lambda':np.arange(1,13)}
xgbr = xgb.XGBRegressor()
gs = GridSearchCV(estimator=xgbr, param_grid=params,
            scoring='neg_root_mean_squared_error',cv=tscv,
            verbose=1,return_train_score=True)
gs.fit(X_train[best_feature_imp], y_train)
print("Best parameters:", gs.best_params_)
```

After running that code, I get this result:

```
Fitting 5 folds for each of 3888 candidates, totalling 19440 fits
Best parameters: {'colsample_bytree': 1, 'lambda': 1, 'learning_rate':
    0.7, 'max_depth': 3, 'min_child_weight': 1, 'n_estimators': 16, '
    subsample': 0.7}
```

4.1.5 Result and discussion

4.1.5.1 Without lockdowns

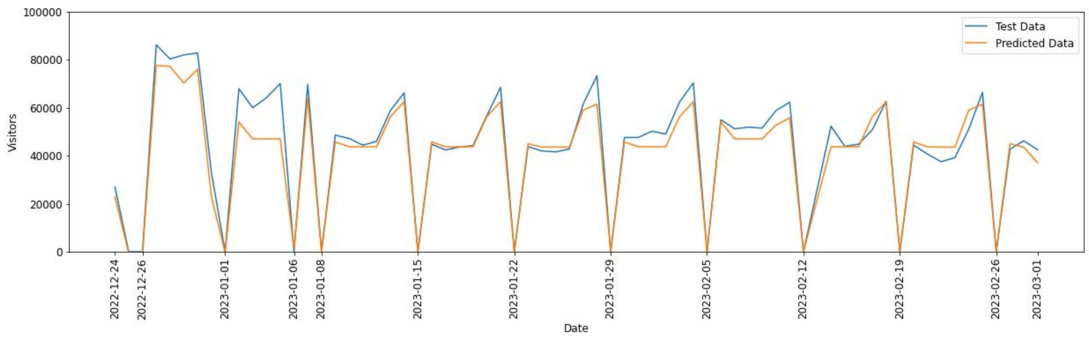


Figure 4.4: Visualisation of actual and predicted values on test set

Index	Metrics	Before improvement			After improvement		
		Train sets	Validation sets	Test set	Train sets	Validation sets	Test set
0	RMSE	33	7179	7952	4363	6285	6080
1	MAE	24	5096	4825	3155	4753	4141
2	R2	1.0	0.88	0.89	0.96	0.91	0.93

Table 4.2: Comparison of results before and after improving for XGBoost

Comparing the result after feature selection and hyperparameter tuning to the baseline before improvement in the table 4.2, I can see a significant improvement in the model's performance on the validation and test sets. The RMSE on the validation set decreases from 7179 to 6285, and on the test set, it decreases from 7952 to 6080. The MAE also decreases on both the validation and test sets. When comparing RMSE on the validation set and test set, a small difference between the two can suggest that the model is performing consistently on both datasets. The R2 score also improved on both the validation and test sets. An R2 value of 0.93 on test set means that 93% of the variance in the target variable is explained by the features in the model. In other words, the model has a good fit and is able to explain a high percentage of the variability in the target variable. In addition, it is worth noting that the optimized model shows no more signs of overfitting on the train set when compared to the baseline. The model before optimization was overfitting, as it performed well on the train set but poorly on the validation and test sets. However, this issue is resolved in the improved model.

Due to the limited number of data points in December, the model fails to capture the pattern during that period. Specifically, the model is trained on data starting from December 13, 2021, and is expected to predict data up to December 24, 2022. Moreover, December exhibits higher fluctuations in the data, which means that the model may not have sufficient information to learn the patterns during this time. Consequently, the higher RMSE compared to MAE on the test set indicates that the model is more sensitive to outliers. The plot of the test data and predicted data in figure 4.4 helps identify specific areas where the model is underperforming, particularly in December and the first week of January. I believe that the model will perform better December in 2023 as it will be trained on more data.

Overall, the results suggest that the feature selection and hyperparameter tuning helped prevent overfitting, improved the model's performance, and optimized training time. The model demonstrated good performance on the validation and test sets, indicating that it can be used to forecast unseen data.

4.1.5.2 With lockdowns

Based on the results above, I can conclude that the model performed well on the unseen data without lockdowns when trained on historical data without lockdowns (14 months). However, it is worth investigating how the model performs when trained on a longer time period (3 years) that includes data with lockdowns. This experiment will help answer the question: Which model will better predict the unseen data without lockdowns, the one built on the shorter data without lockdowns or the one built on the longer data with lockdowns?

The process to train the model on lockdown data is similar to what I did before without lockdowns. The only difference is that I need to add the feature 'lockdown' to the list of total features. The figure 4.5 shows how lockdown impacts the number of visitors to the shopping center.

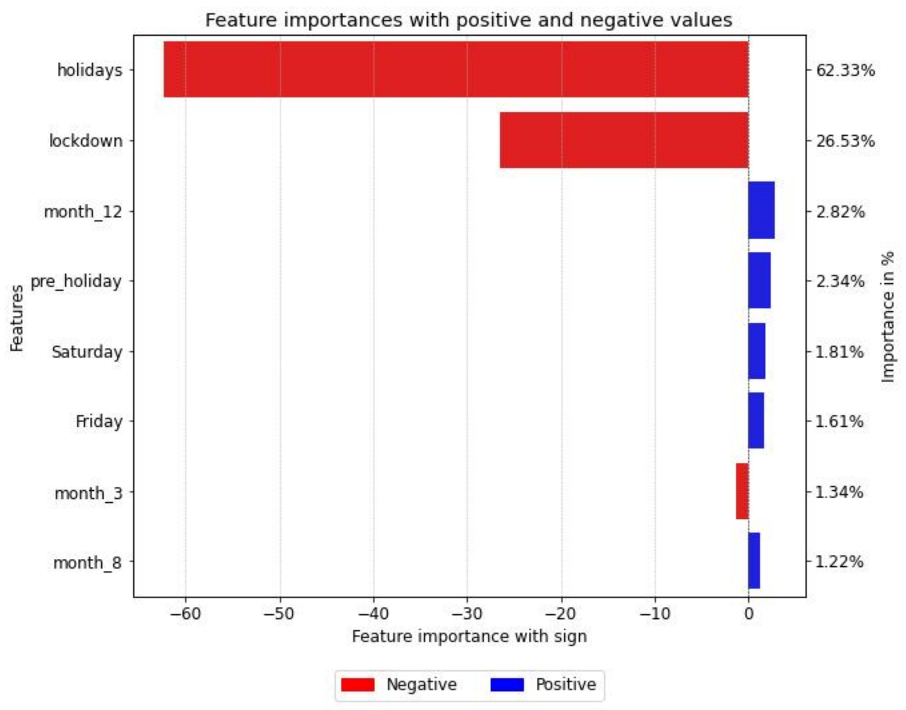


Figure 4.5: Feature importances with lockdowns

From the figure 4.5, I can observe that lockdown has a strong negative impact on the number of visitors to the shopping center.

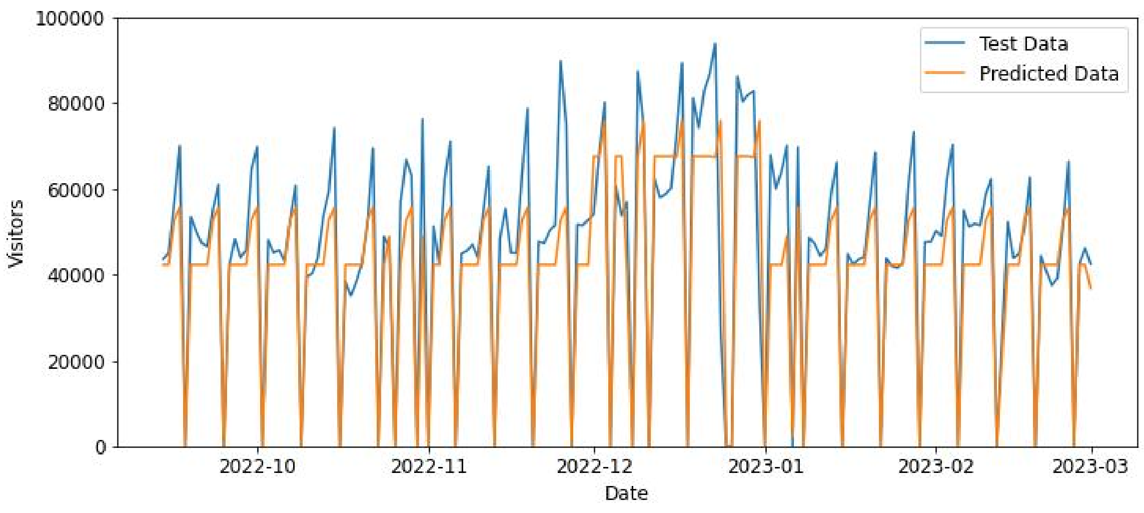


Figure 4.6: Visualisation of actual and predicted values on test set with lockdowns

Index	Metrics	Train sets	Validation sets	Test set
0	RMSE	5998	7175	10690
1	MAE	4003	5075	7144
2	R2	0.92	0.87	0.81

Table 4.3: Result after improvement with lockdowns

When comparing the statistics in table 4.3 with lockdown to those in table 4.2 without lockdown, it is evident that the model's performance is worse when trained on data with lockdowns. Specifically, the RMSE increased from 6080 to 10690 on the test set. To gain a visual understanding, I can refer to figure 4.6, which shows that the model struggles to capture the peaks in the data. This can be explained by the fact that the model is attempting to predict data from Oktober 2022 to February 2023, in this period has no lockdown. However, historical data shows that lockdowns have often occurred during these months, and the number of visitors has shown significant fluctuations during these months, whether there are lockdowns or not. Therefore, to improve the model's performance, it needs to be trained on more data that includes periods without lockdowns.

When calculating the MAPE on test set without holidays, the model trained with lockdowns performs still quite well with an accuracy of 85% compared to 91% when trained without lockdown. Therefore, this lockdown model is considered a contingency option in case of future lockdowns.

4.2 SARIMAX Model

4.2.1 Model description

SARIMAX is a statistical time series forecasting model that stands for Seasonal Autoregressive Integrated Moving Average with Exogenous Variables. It extends the standard ARIMA (Autoregressive Integrated Moving Average) model by adding the ability to handle seasonal data and exogenous variables.

Here is a breakdown of each of the components of the SARIMAX model [5]:

- autoregression (AR): It assumes that the current value of the time series depends on its previous values, with the degree of dependence determined by the order of the AR component.

- integration (I): The I component accounts for the degree of differencing needed to make a time series stationary.

- moving average (MA): It assumes that the current error term is a linear combination of past error terms, with the degree of dependence determined by the order of the MA component.

- seasonal (S): The seasonal component captures patterns that repeat at fixed intervals, such as daily, weekly, or yearly.

- exogenous variables (X): The exogenous variable component allows for the inclusion of external factors that can affect the time series being forecasted.

By combining these components, the SARIMAX model can capture a wide range of time series patterns and relationships, including trends, seasonal effects, and external factors. This makes it a powerful tool for forecasting various types of time series data.

4.2.2 Some specific techniques

4.2.2.1 Augmented Dickey-Fuller test

In time series analysis, it is often necessary to transform the data to ensure that it is stationary before applying statistical models or making predictions. A stationary time series is one where the statistical properties such as the mean, variance remain constant over time. ”In other words, stationarity in time series also means series without a trend or seasonal components”. A stationary time series is easier to model and predict with greater accuracy [12].

The augmented Dickey-Fuller (ADF) test is a statistical test that can be used to determine whether a time series is stationary or not. To perform the ADF test, the adfuller() function in the statsmodels library in Python will be used.

```python
from statsmodels.tsa.stattools import adfuller
def check_stationary(data,maxlag):
  result = adfuller(data,maxlag=maxlag)
  if result[1] < 0.05:
    print("Data is stationary (p_value =",round(result[1],3),").")
  else:
    print("Data is not stationary (p_value =",round(result[1],3),").")
```

4.2.2.2 Bayesian information criterion metric

"Bayesian information criterion (BIC) is a criterion for model selection among a finite set of models" [25]. It provides a way to balance the goodness of fit of a model with its complexity, penalizing models with more parameters. In time series analysis, BIC is often used to compare the fit of different ARIMA models with different values of parameters. The model with the lowest BIC score is considered the best fit for the data.

4.2.2.3 Diagnostics

The plot_diagnostics function in SARIMAX can be used to generate four diagnostic plots that can help assess the quality of the model. Here's a brief explanation of each plot [7]:

- Standardized residual: This plot displays the residuals of the model on the y-axis and the time on the x-axis. A good model will have residuals that are randomly scattered around zero with no obvious patterns.
- Histogram plus estimated density: The histogram displays the actual distribution of residuals, while the orange line represents the kernel density estimation (KDE) curve which is a smoothed version of the histogram. The green line represents a normal distribution. In a good model, the orange line should closely match the green line.
- Normal Q-Q plot: Most of the data points should fall on the straight line, indicating a normal distribution of the residuals.
- Correlogram: This plot illustrates the autocorrelation of the residuals at various lags. A good model will have residuals that exhibit no significant correlation at any lag.

Below is an example of diagnostics which indicates that the model has poor performance.

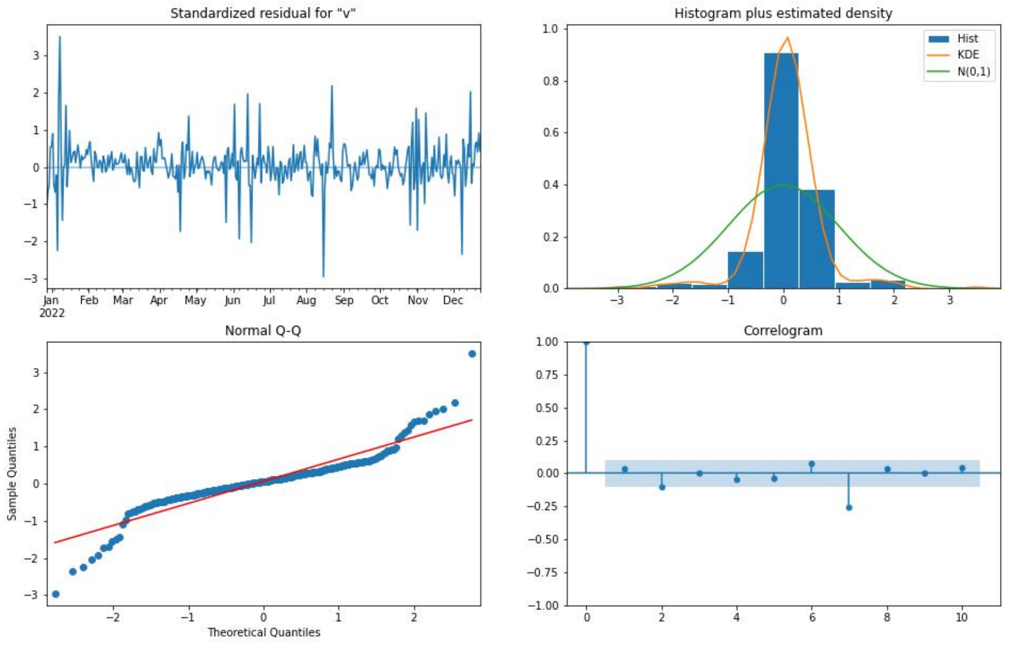


Figure 4.7: Diagnosis indicating poor performance of the model

4.2.3 Parameters

4.2.3.1 Meaning of parameters

Order parameters (p, d, q) and seasonal order parameters (P, D, Q) are key components of the SARIMAX model. The order parameters specify the number of lags used in the autoregressive and moving average components, while the seasonal order parameters specify the number of lags used in the seasonal autoregressive and seasonal moving average components.

The order parameters are typically denoted as p, d, and q, where [8]:

- p represents the order of the AR component, which determines the number of lagged observations used to predict the current value of the time series.
- d represents the degree of differencing needed to make the time series stationary.
- q represents the order of the MA component, which determines the number of lagged forecast errors used to predict the current value of the time series.

The seasonal order parameters are typically denoted as P, D, and Q, where:

- P: represents the seasonal autoregressive order.
- D: represents the seasonal difference order.
- Q: represents the seasonal moving average order.

The additional parameter m represents the number of time steps in each seasonal cycle. For example, if the time series exhibits a weekly seasonality, m would be 7.

4.2.3.2 Finding parameters

- **Finding m**
  One way to determine the value of m is to use statistical methods. One common approach is to use autocorrelation plots or partial autocorrelation plots to identify the seasonal patterns in the time series. The lag at which the autocorrelation or partial autocorrelation plot shows a significant peak can give an indication of the length of the seasonal cycle.

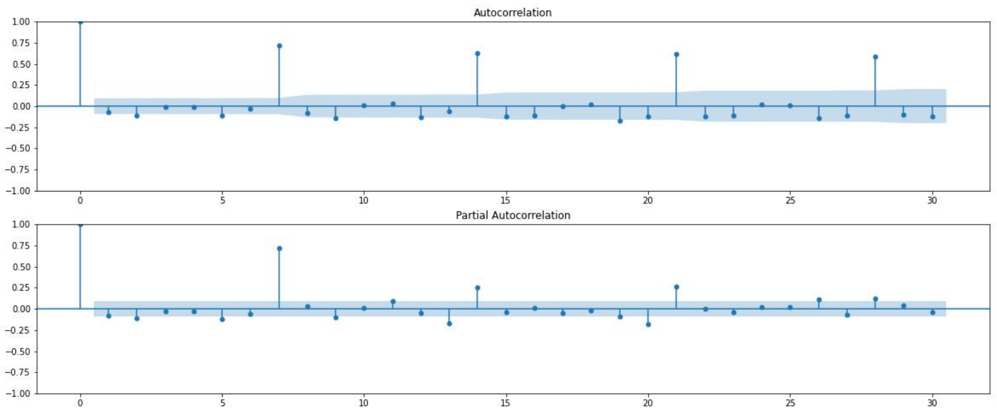


Figure 4.8: Test results for seasonal

  The figure 4.8 shows significant peaks at 7th lag, hence m would be set to 7.

- **Finding d**
  Once the ADF test is performed, the results are used to determine the degree of differencing required to make the time series stationary. If the p-value is less than the significance level (0.05), then the time series can be considered stationary, and no differencing is required (it means, $d = 0$). If the p-value is greater than the significance level, differencing will be applied and re-running the test until the time series becomes stationary [13]. After applying the ADF test on the train sets, I can conclude that the value of d is less than 2 ($d < 2$).

- **Finding D**
  To determine D, the ADF test was conducted on the differenced time series of the train_fold and training sets. The results indicated that the seasonal differencing was sufficient to make both time series stationary, therefore $D = 0$.

- **Finding the best combination of $(p, d, q) \times (P, D, Q, m)$**

  If the values of p or q are larger than the value of m, it indicates that the model is trying to capture short-term dependencies beyond the length of the seasonal cycle. This may result in overfitting and poor model performance. Therefore, it is recommended to set the range of p and q to be less than or equal to the value of m. In this case, since $m = 7$, the maximum value of p and q is 7.

  As the values of P and Q increase, the model becomes more complex and may fit the noise, which can result in poor model performance on unseen data. In this case, the maximum values of P and Q are set to 2.

To find the optimal values of (p, d, q) and (P, D, Q, m), I can use the built-in function *auto_arima*() of the sklearn library. However, this function has some limitations. Alternatively, I can find the best match by iterating through possible combinations of (p, d, q, P, D, Q) based on the metric BIC.

To find the best values of (p, d, q) and (P, D, Q, m), I can use the code below on both the train['visitors'] and train_fold['visitors'] datasets.

```
import itertools
result = []
for p, d, q, P, Q in itertools.product(range(8), range(2), range
    (8), range(3), range(3)):
  try:
    model = SARIMAX(train['visitors'], order=(p,d,q),
        seasonal_order=(P,0,Q,7))
    output = model.fit(disp=False)
    result.append([p,d,q,P,0,Q, output.bic])
  except:
    continue
df_result_train = pd.DataFrame(result, columns=['p','d','q','P','D
    ','Q', 'bic'])
globals()["Train"] = df_result_train.sort_values(by=['bic'],
    ascending=True)
```

I can then merge the resulting dataframes and compute the mean of the BIC values. This approach can help me identify the best combination of parameters.

p	d	q	P	D	Q	mean_bic
1	0	1	1	0	0	4248.28
2	0	4	0	0	2	4313.18
1	0	1	2	0	2	4444.84
6	0	3	2	0	1	4484.43
2	0	2	0	0	1	4554.67

Table 4.4: Finding parameters of SARIMAX

Based on the results from 4.4, I can conclude that the optimal combinations are $(p, d, q) = (1, 0, 1)$ and $(P, D, Q, m) = (1, 0, 0, 7)$.

4.2.4 Exogenous variables

When working with SARIMAX model, the values of exogenous variables should be carefully selected, as the value 0 of an exogenous variable will be ignored during the modeling process. It is important to note that binary exogenous variables should be assigned non-zero values. For instance, the value of the exogenous variable "holidays" should be set to -1 and 1, where -1 represents holidays, and 1 represents non-holidays. To do this, I can use the following code:

```
df_sarimax.holidays = df_sarimax.holidays.map({1: -1, 0: 1})
```

4.2.5 Feature selection

In the SARIMAX model, there is no function to measure the importance of exogenous variables. However, I can look at the summary table of the model, which can be obtained by calling `model_sarimax.summary()`, to get an idea. The absolute value of the z-statistic in the summary table indicates the importance of exogenous variables. Therefore, to find the most important exogenous variables, I can extract the values of the z-statistics and arrange them in descending order.

4.2.6 Result and discussion

4.2.6.1 Without lockdowns

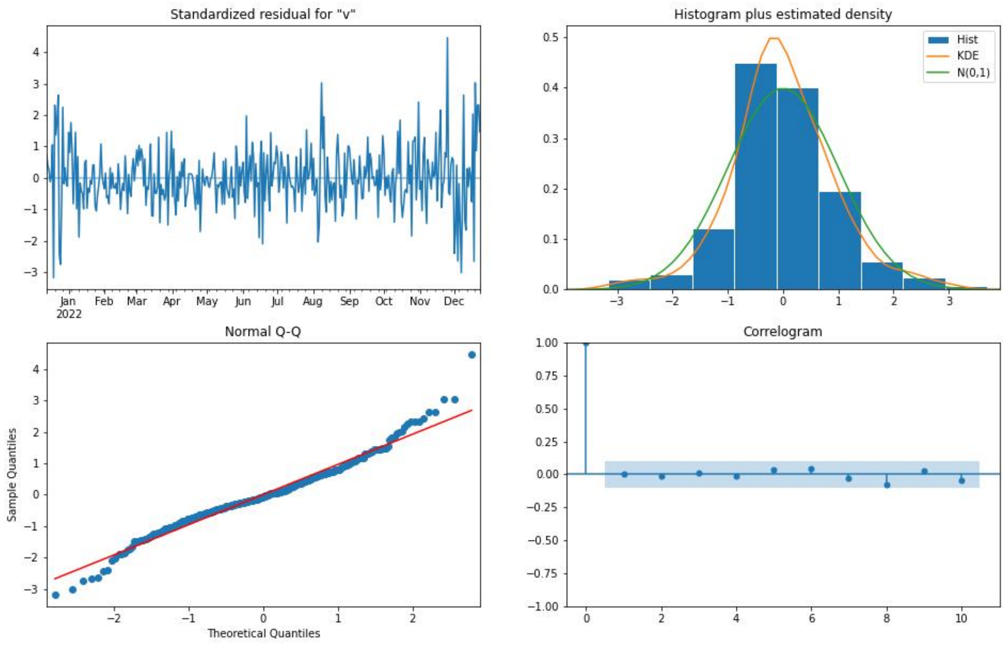


Figure 4.9: Diagnosis of SARIMAX model without lockdowns

I can evaluate the performance of the model by examining the residuals plot. From the residuals plot 4.9, I can observe that the residuals are randomly scattered around zero without obvious patterns. In the Normal Q-Q plot, the residuals fall almost along the diagonal line. Additionally, in the correlogram, there are no significant correlations at any lag. However, the attitude of the blue line does not fit perfectly with the orange line. Ultimately, based on the residuals plot, I can conclude that the model has performed quite well. Let us now examine the performance of the model through the plot 4.10 and the statistical table 4.5:

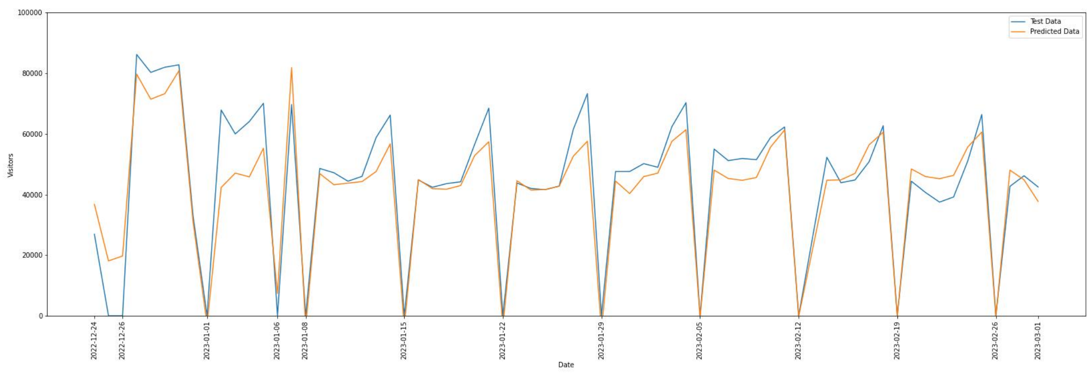


Figure 4.10: Visualization of actual and predicted values on the test set without lockdowns for SARIMAX

Index	Metrics	Train sets	Validation sets	Test set
0	RMSE	6032	7892	7830
1	MAE	4425	5937	5754
2	R2	0.93	0.86	0.89

Table 4.5: Results of SARIMAX model without lockdowns

The RMSE on the validation sets and test set are relatively similar, as indicated in the table 4.5. This suggests that the distribution of data in the test set and validation sets are similar, and the model is able to make accurate predictions on unseen data, specifically on the test set, as it was trained. In addition, the RMSE value on the train set is also close to the RMSE values on the validation sets, indicating that the model is not overfitting. However, there is a noticeable difference of $7830 - 5754 = 2076$ visitors between the RMSE and MAE on the test set, which suggests that the model may not have captured outliers well. This can be also observed in the visualization of actual and predicted values on the test set, as shown in the figure 4.10.
Overall, based on the diagnostics 4.9, the visualization 4.10, and the statistical table 4.5, I can conclude that the model has performed quite well.

4.2.6.2 With lockdowns

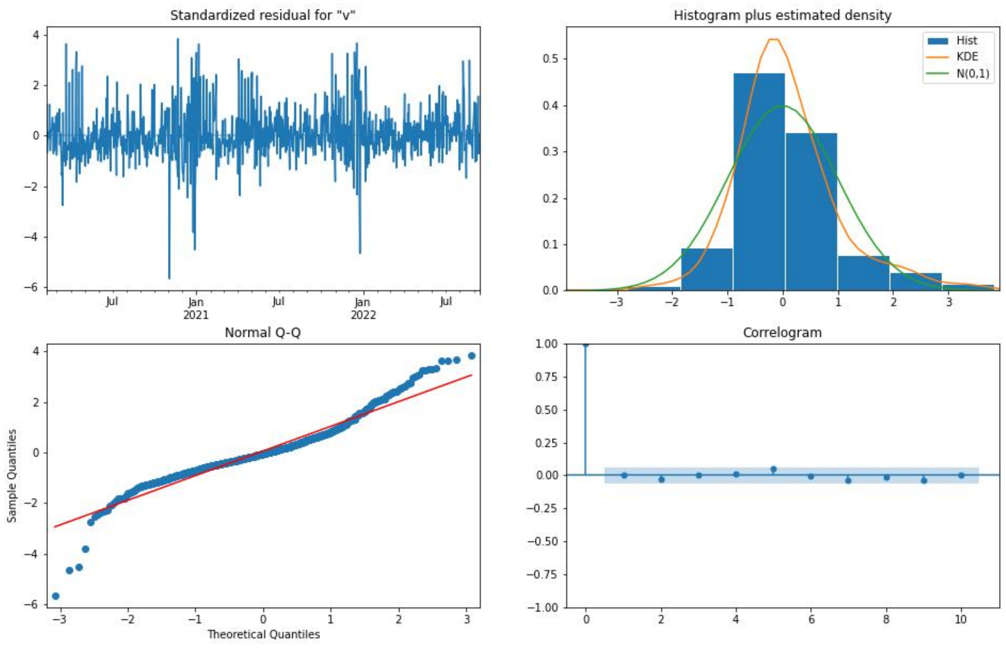


Figure 4.11: Diagnosis of SARIMAX model with lockdowns

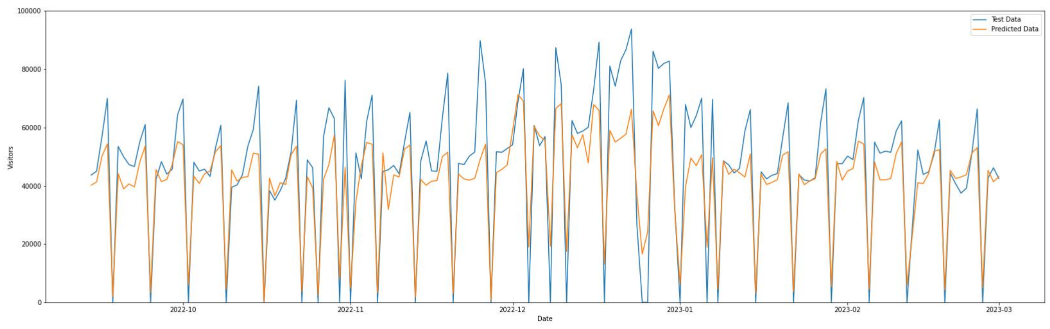


Figure 4.12: Visualization of actual and predicted values on the test set with lockdowns for SARIMAX

Index	Metrics	Train sets	Validation sets	Test set
0	RMSE	7708	9278	11395
1	MAE	5431	6923	8451
2	R2	0.87	0.79	0.78

Table 4.6: Results of SARIMAX model with lockdowns

In Figure 4.11, in the histogram plus estimated density plot, the green line does not closely match the orange line. Additionally, in the Normal Q-Q plot, data points tend to deviate from the red line. These observations indicate that the model's performance is not optimal. This becomes even more apparent when observing Figure 4.12, where the predicted line does not closely align with the actual line, particularly near the peak points.

When comparing the statistics in Table 4.6 to those in Table 4.5, it is evident that the model performs worse when trained on data with lockdowns. The RMSE increased from 7892 to 9278 on the validation sets and from 7830 to 11395 on the test set. Consequently, I can conclude that when predicting data without lockdowns, it is preferable to use a model trained on data without lockdowns, even if there is more data available with lockdowns than without.

4.3 Models comparision

Index	Metrics	SARIMAX			XGBoost		
		Train sets	Validation sets	Test set	Train sets	Validation sets	Test set
0	RMSE	6032	7892	7830	4363	6285	6080
1	MAE	4425	5937	5754	3155	4753	4141
2	R2	0.93	0.86	0.89	0.96	0.91	0.93

Table 4.7: Comparison of results: SARIMAX vs XGBoost

Based on the evaluation metrics for SARIMAX and XGBoost models on the training sets, validation sets, and test set, as shown in the table 4.7, I can make the following comparisons: RMSE and MAE: XGBoost has lower RMSE values and MAE values on all the training sets, validation sets and test set, suggesting better accuracy in predicting the number of visitors during model development. Moreover, R2: XGBoost has higher R2 scores on all the training sets, validation sets and test set, indicating better explanatory power in predicting the number of visitors during model development. Additionally, the model XGBoost used 9 features, while the model SARIMAX used 21 features to predict unseen data. It indicates that the model XGBoost is simpler with fewer features. Overall, it is reasonable to conclude that XGBoost is a better choice for forecasting the number of visitors to the shopping center based on my data.

Why does the XGBoost model have better performance than the SARIMAX model in this problem? Here are 2 potential reasons:

- Model flexibility: XGBoost is a more flexible model compared to SARIMAX. It can capture complex nonlinear relationships and interactions between features, allowing it to learn from the data more effectively. SARIMAX, on the other hand, is a linear model that assumes specific time series properties, such as stationarity and seasonality [1].

- Ensemble learning: XGBoost uses a special method called ensemble learning. It combines many small models (decision trees) to make predictions. By working together, these models can correct each other's mistakes and improve the overall accuracy. SARIMAX, on the other hand, relies on a single model structure.

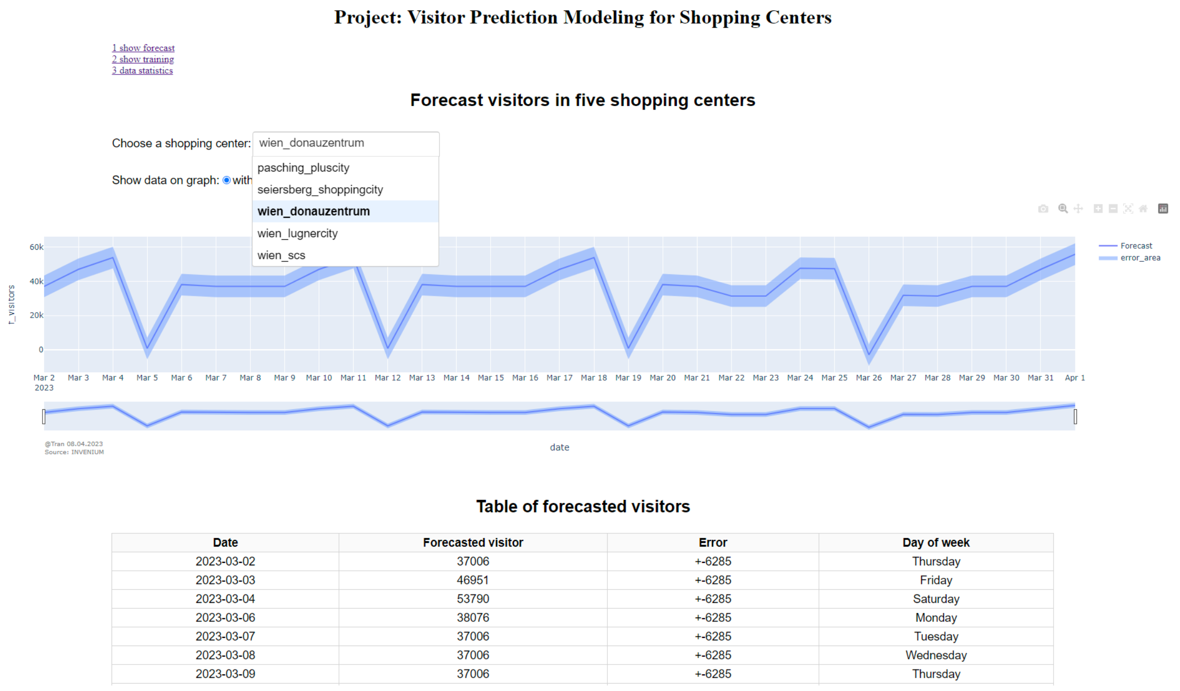
Chapter 5

Visualization

To visualize the data in this thesis, I choose Dash, a Python framework that builds web applications using Flask, Plotly.js, React, and React.js [18]. It enables the creation of interactive and responsive data visualizations.

I divide the visualization content into three pages:

- **Show forecast**
  On this page, users can choose one of five shopping centers to see the forecasted visitors as a chart as well as tabular.



Project: Visitor Prediction Modeling for Shopping Centers

1 show forecast
2 show training
3 data statistics

Forecast visitors in five shopping centers

Choose a shopping center: wien_donauzentrum

Show data on graph: [✓ with]

Table of forecasted visitors

Date	Forecasted visitor	Error	Day of week
2023-03-02	37006	+-6285	Thursday
2023-03-03	46951	+-6285	Friday
2023-03-04	53790	+-6285	Saturday
2023-03-06	38076	+-6285	Monday
2023-03-07	37006	+-6285	Tuesday
2023-03-08	37006	+-6285	Wednesday
2023-03-09	37006	+-6285	Thursday

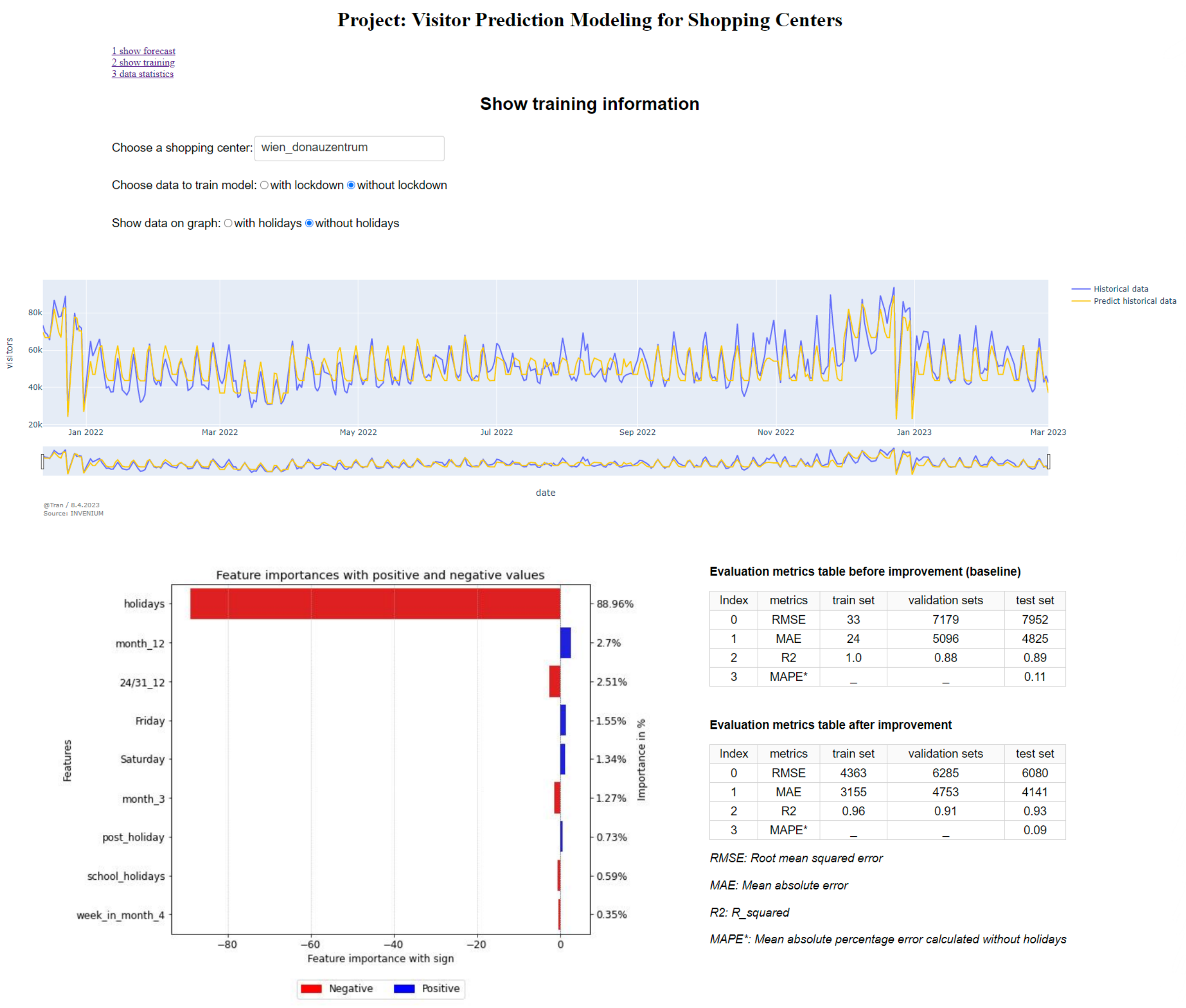
Figure 5.1: Show forecast (screenshot)

  Moreover, users can choose to display data on the graph with or without holidays by selecting one of the two options ’with holidays’ and ’without holidays’.

  The graph shows the forecasted visitors for a one-month period, represented with an error area. Below the graph, there is a table of forecasted visitors that provides more specific values for each day.

- **Show training**
  On this page, users can select one of five shopping centers to view the training

information graphically. Similar to the previous page, users can choose to display the data on the graph with or without holidays. Additionally, users have the option to select the training data, either with or without lockdown, for model training.



Project: Visitor Prediction Modeling for Shopping Centers

1 show forecast
2 show training
3 data statistics

Show training information

Choose a shopping center: wien_donauzentrum

Choose data to train model: with lockdown [✓ without lockdown]

Show data on graph: with holidays [✓ without holidays]

Evaluation metrics table before improvement (baseline)

Index	metrics	train set	validation sets	test set
0	RMSE	33	7179	7952
1	MAE	24	5096	4825
2	R2	1.0	0.88	0.89
3	MAPE*	_	_	0.11

Evaluation metrics table after improvement

Index	metrics	train set	validation sets	test set
0	RMSE	4363	6285	6080
1	MAE	3155	4753	4141
2	R2	0.96	0.91	0.93
3	MAPE*	_	_	0.09

RMSE: Root mean squared error

MAE: Mean absolute error

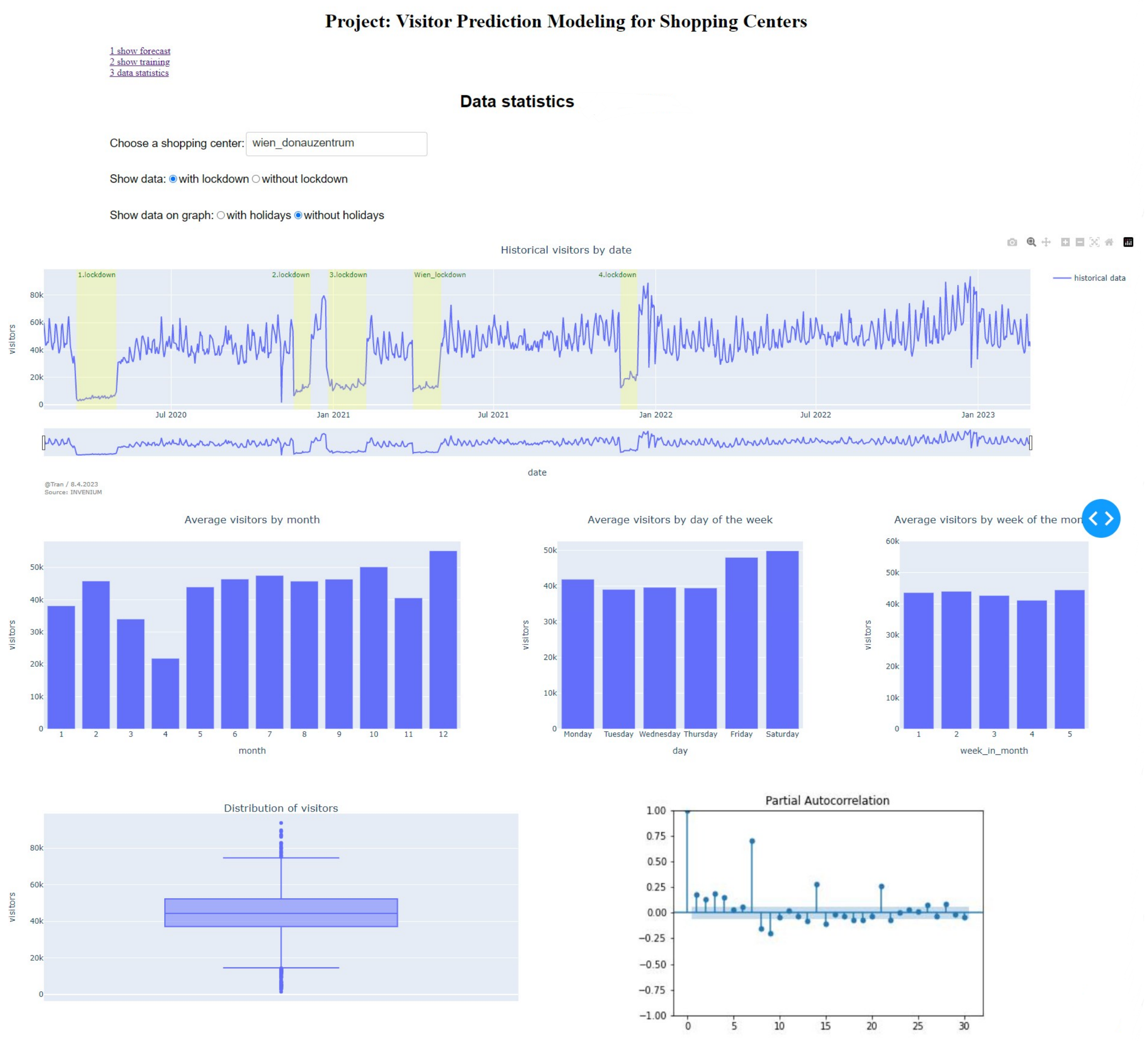
R2: R_squared

MAPE: Mean absolute percentage error calculated without holidays*

Figure 5.2: Show training (screenshot)

The line graph displays the historical data along with the predicted historical data. And then on the left side, there is a chart indicating the feature importances, with positive and negative values. On the right side, there are the evaluation metrics tables before and after improvement.

- **Data statistics**


Project: Visitor Prediction Modeling for Shopping Centers

1 show forecast
2 show training
3 data statistics

Data statistics

Choose a shopping center: wien_donauzentrum

Show data: [✓ with lockdown] without lockdown

Show data on graph: with holidays [✓ without holidays]

Figure 5.3: Data statistics (screenshot)

This page displays statistical information to explore the received data. The line graph shows the historical data. Following that, there are three column charts indicating the traffic by month, by day of the week, and by week of the month. The next graph is the 'Distribution of Visitors', which indicates the minimum, maximum, and mean values of the data. Additionally, the 'Partial Autocorrelation' graph shows the partial autocorrelation of visitors and its lagged values. Lastly, the 'Correlation between Variables' heatmap represents the correlation coefficients between pairs of variables in a dataset.

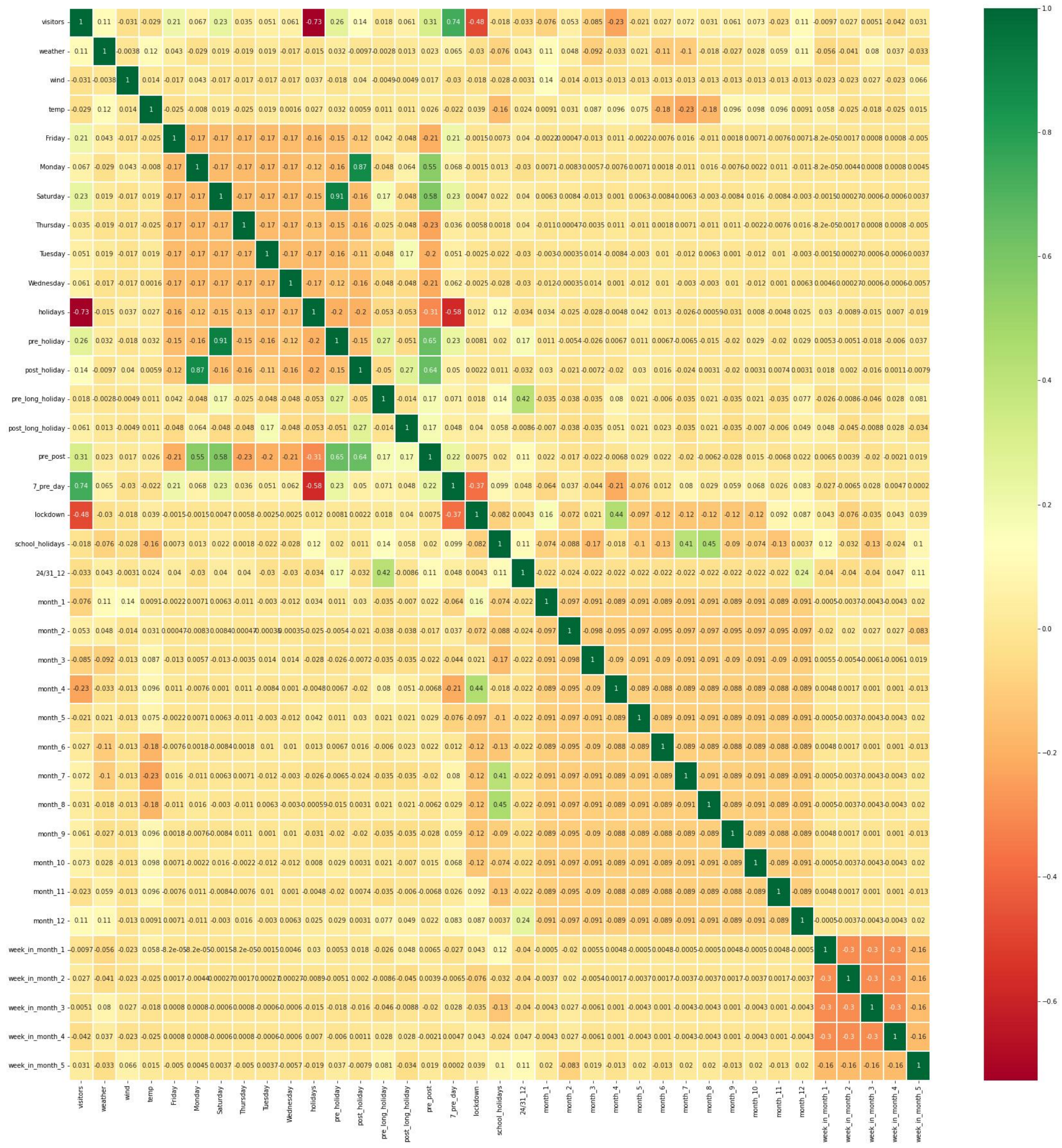


Figure 5.4: Data statistics (continued) (screenshot)

Chapter 6

Conclusion, Limitation and Outlook

6.1 Conclusion

This thesis explores two main methods for the time series forecasting problem: the XGBoost model from machine learning and the SARIMAX model from statistical modeling, and apply them to forecast the number of visitors in five shopping centers located in Austria.

The dataset used includes historical visitor data, weather data, public holidays, and school holidays information.

After conducting various comparisons between the XGBoost and SARIMAX models, the XGBoost model is chosen as the preferred method for visitor forecasting.

The accuracy of the predictions depends on the specific data of each shopping center, but overall, the model achieves good performance, ranging from 84% to 91% accuracy. Common factors that impact the visitor count across all shopping centers include holidays, December, Friday and Saturday. Additionally, each shopping center has its own unique features that influence visitor count. The effect of weather is relatively small and is observed in only two out of the five shopping centers, namely PlusCity in Pasching and Shopping City Süd in Vienna.

Furthermore, when the XGBoost model is trained on data during lockdown periods, there is a slight decrease in accuracy. However, it is important to note that the model still performs reasonably well, ranging from 84% to 89% accuracy. Therefore, this model can serve as a contingency option in case of future lockdowns.

These findings provide valuable insights for resource management and decision-making in the retail industry, empowering shopping center managers to make informed choices based on accurate visitor forecasts.

6.2 Limitations

- **Limitations in weather data availability**
  The received weather data lacks two pieces of information that could be relevant to the visitor count. Firstly, there is no information about the amount of rainfall. Although I can determine if it is raining or not from the weather data description, I cannot determine the intensity of the rainfall. The amount of rain can have an impact on the number of visitors. Secondly, during my research on the three shopping centers in Vienna, I found no available weather

data specifically for the Vienna area. Instead, I had to rely on weather data from Eisenstadt, a city that is 62 km away from Vienna. I believe that having sufficient weather data, including the rainfall amount and weather conditions in Vienna, would provide clearer and more significant insights into the impact of weather on visitor numbers.

- **Limitations in visitor data availability**
  The available visitor data spans a period of 3 years, but it may not be sufficient for building a strong predictive model due to certain limitations. The data contains some challenges that need to be addressed. The first issue is related to the presence of lockdowns, which occurred mostly in April and the end of the year during the initial 2-year period. I only have data for one April and December without lockdowns, and it is noting that the visitor count during these months exhibits significant fluctuations. Consequently, the model did not capture these months accurately. To enhance the model's performance during these months, it would be beneficial to have additional data without lockdowns, starting from at least 2019.

6.3 Outlook

- **Hourly visitor prediction**
  In this project, I aim to predict the visitor count in shopping centers based on historical daily visitor data. However, if I have visitor data available on an hourly basis, I can enhance my predictions by including the feature 'hour' as part of the feature set. Additionally, the accuracy of the model can be increased as the weather factors will be more precise by the hour. By incorporating the hour of the day, I can forecast the number of visitors for each specific hour. This information enables the shopping center to optimize their staffing levels, ensuring appropriate customer service and operational efficiency at different times throughout the day.

- **Effect of promotion**
  Variables related to promotions, such as discounts, sales, or special offers, can have an impact on visitor behavior and attract more people to shopping centers. Therefore, including the 'promotion' feature in the model can potentially improve the prediction of visitor counts.

- **Expanding the research to include additional shopping centers**
  Although the current model achieves high accuracy, it does not provide clear answers regarding why certain features affect one shopping center but not others, or if the determining characteristics of shopping centers play that role. For instance, the weather factor impacts two out of the five shopping centers, but the reasons behind this selective impact are unknown. By expanding the project to include a larger number of shopping centers, I can potentially address these questions. This expanded project will not only provide forecasts for shopping center leaders but also offer insights on how to attract more visitors to shopping centers.

- **Deep learning**
  Deep learning methods have shown great potential in predicting time series data. The neural network architecture learns features from the data

autonomously and captures all non-linear relationships. Additionally, deep learning methods are capable of recognizing and handling patterns such as trends and seasonality. This makes them highly valuable for making accurate predictions [9]. Therefore, it is beneficial to expand the project's direction to deep learning and compare which method yields better results. However, deploying deep learning requires a larger volume of data, more time for training and tuning, and greater expertise in working with deep learning models.

Technical specifications

The following technical specifications are commonly used in predictive models, as seen in [15]. This thesis was conducted using Python 3 (3.8.10) with Jupyter Lab. Therefore, it is necessary to have Python 3 and Jupyter Lab installed on your machine. In addition, the following Python 3 packages must also be installed to run the models:

- numpy (1.23.0)
- pandas (1.4.3)
- matplotlib (3.5.2)
- seaborn (0.12.0)
- sklearn (1.1.1)
- xgboost (1.7.0)
- dash (2.6.2)

Explanation of abbreviations in code

Abbreviation	Explanation
train	Training set
test	Test set
df_xgb	Dataframe for XGBoost
n_splits	Number of splits
tscv	Time series cross validation
true_list	List of actual values
pred_list	List of predicted values
best_feature_imp	Array of best features
xgbr	Extreme gradient boosting regressor
y_train	Array of visitor values in the training set
X_train	The rest of the training set, complementary to y_train
y_train_fold	Array of visitor values in the folded training set
X_train_fold	The rest of the folded training set, complementary to y_train_fold
y_val_fold	Array of visitor values in the folded validation set
X_val_fold	The rest of the folded validation set, complementary to y_val_fold

List of Tables

List of Figures

Abbreviations

ACF	autocorrelation function
ADF	augmented Dickey-Fuller
AR	autoregression
ARIMA	autoregressive integrated moving average
BIC	Bayesian information criterion
I	integration
MA	moving average
MAE	mean absolute error
MAPE	mean absolute percentage error
PACF	partial autocorrelation function
R2	R squared
RMSE	root mean squared error
S	seasonal
SARIMAX	seasonal autoregressive integrated moving average with exogenous variables
X	exogenous variables
XGBoost	eXtreme Gradient Boosting

Bibliography

[1] AAWEG. Time series forecasting: A comparative analysis of sarimax and xgboost algorithms. https://aaweg-i.medium.com/time-series-forecasting-a-comparative-analysis-of-sarimax-and-xgboost-algorithms-7cac99685564, 2023. Accessed: 2023-05-21.

[2] Raghav Agrawal. Know the best evaluation metrics for your regression model. https://www.analyticsvidhya.com/blog/2021/05/know-the-best-evaluation-metrics-for-your-regression-model/, 2022. Accessed: 2023-04-03.

[3] Anonymous. Chronologie der pandemie . https://www.kleinezeitung.at/politik/innenpolitik/6056751/Lockdowns-und-Lockerungen_Chronologie-der-Pandemie, 2021. Accessed: 2023-05-14.

[4] Anonymous. Introduction to xgboost algorithm in machine learning. https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/, 2023. Accessed: 2023-05-18.

[5] Prashant Banerjee. Arima model for time series forecasting. https://www.kaggle.com/code/prashant111/arima-model-for-time-series-forecasting#1.-Introduction-to-Time-Series-Forecasting-, 2021. Accessed: 2023-04-06.

[6] Prashant Banerjee. Comprehensive guide on feature selection. https://www.kaggle.com/code/prashant111/comprehensive-guide-on-feature-selection, 2021. Accessed: 2023-04-03.

[7] Danielle Paes Barretto. Time series part 2: Forecasting with sarimax models: An intro. https://www.jadsmkbdatalab.nl/forecasting-with-sarimax-models/. Accessed: 2023-05-20.

[8] Nana Boateng. Time series analysis methods. https://rstudio-pubs-static.s3.amazonaws.com/303786_f1b99d6b7e9346c4b1488a174bab839a.html. Accessed: 2023-05-20.

[9] Jason Brownlee. Deep learning for time series forecasting. https://machinelearningmastery.com/deep-learning-for-time-series-forecasting/. Accessed: 2023-05-24.

[10] Jason Brownlee. How to get baseline results and why they matter. https://machinelearningmastery.com/how-to-get-baseline-results-and-why-they-matter/, 2014. Accessed: 2023-04-05.

[11] Jason Brownlee. Avoid overfitting by early stopping with xgboost in python. https://machinelearningmastery.com/avoid-overfitting-by-early-stopping-with-xgboost-in-python/, 2016. Accessed: 2023-04-04.

[12] Vijay Kumar G. Statistical tests to check stationarity in time series. https://www.analyticsvidhya.com/blog/2021/06/statistical-tests-to-check-stationarity-in-time-series-part-1/, 2021. Accessed: 2023-04-06.

[13] Vijay Kumar G. Statistical tests to check stationarity in time series. https://www.analyticsvidhya.com/blog/2021/06/statistical-tests-to-check-stationarity-in-time-series-part-1/, 2023. Accessed: 2023-05-20.

[14] Aarshay Jain. Xgboost — complete guide to parameter tuning in xgboost with codes. https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/, 2023. Accessed: 2023-04-04.

[15] Sheridan Kamal. An analysis of machine learning techniques for economic recession prediction. 2021.

[16] David Martins. Xgboost: A complete guide to fine-tune and optimize your model. https://towardsdatascience.com/xgboost-fine-tune-and-optimize-your-model-23d996fab663, 2021. Accessed: 2023-04-04.

[17] Kyle B. Murray, Fabrizio Di Muro, Adam Finn, and Peter Popkowski Leszczyc. The effect of weather on consumer spending. *Journal of Retailing and Consumer Services*, 17(6):512–520, 2010.

[18] Derrick Mwiti. Dash for beginners. https://www.datacamp.com/tutorial/learn-build-dash-python, 2018. Accessed: 2023-05-21.

[19] Manan Parasher. Top 8 cross validation methods!!! https://www.moredatascientists.com/top-8-cross-validation-methods/. Accessed: 2023-05-19.

[20] Spyridon Petsis, Areti Karamanou, Evangelos Kalampokis, and Konstantinos Tarabanis. Forecasting and explaining emergency department visits in a public hospital. *Journal of Intelligent Information Systems*, 59(2):479–500, 2022.

[21] Sharoon Saxena. Underfitting vs. overfitting (vs. best fitting) in machine learning. https://www.analyticsvidhya.com/blog/2020/02/underfitting-overfitting-best-fitting-machine-learning/, 2020. Accessed: 2023-04-04.

[22] Rahul Shah. Tune hyperparameters with gridsearchcv. https://www.analyticsvidhya.com/blog/2021/06/tune-hyperparameters-with-gridsearchcv/, 2022. Accessed: 2023-05-18.

[23] Virenrehal. Interpreting acf and pacf plots. https://spureconomics.com/interpreting-acf-and-pacf-plots/, 2022. Accessed: 2023-05-19.

[24] Abhishek Jha Vladimir Lyashenko. Cross-validation in machine learning: How to do it right. https://neptune.ai/blog/cross-validation-in-machine-learning-how-to-do-it-right, 2023. Accessed: 2023-05-19.

[25] Wikipedia. Bayesian information criterion. https://en.wikipedia.org/wiki/Bayesian_information_criterion. Accessed: 2023-04-06.

[26] Wikipedia. Beaufort scale. https://en.wikipedia.org/wiki/Beaufort_scale. Accessed: 2023-04-03.

[27] Wikipedia. Mean absolute percentage error. https://en.wikipedia.org/wiki/Mean_absolute_percentage_error. Accessed: 2023-05-13.

[28] Zach. Validation set vs. test set: What's the difference? https://www.statology.org/validation-set-vs-test-set/, 2021. Accessed: 2023-04-03.