Phase IV Report

Smit Soni[1], Siddhant Sahoo[2], Negin Shademan[3], MaHbfoubeh Tajmirriahi[4]

Universität Passau, Passau, Germany
soni01@ads.uni-passau.de[1], sahoo01@ads.uni-passau.de[2], shadem01@ads.uni-passau.de[3], tajmir01@ads.uni-passau.de[4]

Abstract. Mobility solutions have seen a major boost in recent years. Forecasting the number of people and estimating the corresponding demand for public transport has become more crucial as a result. Given the data collected by DB Regio [1] of the tickets sold, we predict if there are 0,1,2 or 3 or more than 3 passengers at a given bus stop. Two different kinds of services are provided by DB Regio [DB], one is the regular route which a bus follows for every trip it makes in the day, whereas there are on-demand routes which are decided if a passenger wants to board, at the bus stop based on the online request. After analysing the data we were provided with, we applied Decision Tree, AdaBoost with Decision Tree, Random forest, Naïves Bayes, and SARIMA models. The results show that AdaBoost performed the best for regular travel while for on-demand travel Decision Tree was the best performer.

1 Introduction

How people move about enables public transportation systems to be improved, which advances the overall decarbonization objective. This is especially valid in sparsely populated rural areas. Predicting ticket sales can be an efficient solution for many problems, including decarbonization, allocating buses with enough seating capacities, increasing the availability at peak hours as well as decreasing the operating costs of the company. A different approach for the same could be to find the busiest bus stops. It often happens that a bus goes empty during a journey while the other is overcrowded.

With the provided data of the top 50 bus stops in and around Passau, Bavaria, we try to predict passenger volume at each bus stop at each hour of the day into three classes: high demand, medium demand, and low demand. The data is gathered by DB Regio [1] which offers two different kinds of bus services, one is the regular and predefined bus routes that a particular bus must follow throughout its journey while the other one is on-demand service. These routes are defined by already knowing prior information about the passenger's start and end stop so that the bus covers only those bus stops which are required by the passenger. When predicting passenger volume and routes of the bus, many other external factors might also affect the travel. For example, if it is raining, less number of people may use a bus, or during holidays passenger volume might see a sudden hike or drop. Hence along with the given data on on-demand travel and regular routes, the weather on a particular day/time should also maybe considered. Apart from these for on-demand travel, to determine the bus routes we also have WOHIN DU WILLST data which contains the queries about the start and end stops that people looked up on the web, which might indicate that a passenger might be demanding stops on those start and end stops. To make predictions even more accurate we also have longitudinal and latitudinal coordinates of bus stops which can be used to determine the distance between any two stops. Also by knowing the location coordinates, one further application could be re-plan the already existing bus routes and hence have efficient yet reliable public transport. Since the two kinds of services are independent of each other, we proceed by building separate models for both the data, taking scores individually from them and then combining to classify a particular bus stop.

2 Related Work

Applications of time-series methods for forecasting bus passenger demand were studied by Cyril et al. [8]. They have used Holt Winter's modelling method. In this work, utilizing information from the inter-zonal buses, additive and multiplicative Holt-Winters models with and without damping have been empirically compared. The study's data came from the Kerala State Road Transport Corporation's (KSRTC) ticket sales transactions, which were kept at the Kerala state's Trivandrum City depot between 2010 and 2013.

They used the patterns obtained from the data to build a model using methods of Autoregressive (AR) models and Autoregressive Conditional Heteroscedasticity (ARCH). The weighted average of past observations is used with the new data points getting higher weightage and older data points getting lower weightage which decreases exponentially as the new data points are added. Smoothing techniques such as Simple Exponential Smoothing (SES), Double Exponential Smoothing (DES), and Holt-Winters' method were used to assign weights to the observations. Holt-Winters' model was used when the data had both trend and seasonality. Mean Absolute Percentage Error was used to evaluate the predicting abilities of four time-series models, and information criteria is used to assess the model's goodness of fit.

Stadler et al. [13] analysed all the bus routes in the rural area of Roding, Bavaria, Germany, and hence predicted passenger demand. The data used was transferred by the bus company employee from the ticket system to excel, which was previously only used for accounting purposes. This data included the type of ticket, zone, departure time, date, and start and end stop. Synthetic data was generated for covering those bus stops and small towns where only a few people boarded the bus. The model was implemented by tuning individual parameters and thus using prefabricated functions. To find out how many people travel to a certain start or end bus stop, an origin-destination matrix was then generated that would also help in determining the busiest bus stops and average demand. On the data, the Auto Correlation Function(ACF) and Partial Auto Correlation Function(PACF) were plotted to find out the Autoregressive(AR) and moving average(MA) components. The authors then generated random data and classified tickets into two categories “purchase” and “no purchase” whereby no purchase might include such tickets which may be student passes or monthly passes. Following on, three different models were tested, Naïve Bayes, Random Forrest, and XGBoost which were trained at a learning rate of 0.001 in 100,000 iterations. The results showed XGBoost being the most accurate while Naïve Bayes being the worst, as it assumes all the features are independent of each other.

3 Data Acquisition and Analysis

3.1 Data collection

The data for the on-demand and regular routes over the year 2019 has been provided by Deutsche Bahn Regional AG (Germany) [1]. The *regular_travel* file consists of 310800 entries and 4 columns and the *on_demand_travel* file contains 438000 records and the same 4 columns. The data collected includes the date, hour, the bus stop where the ticket was obtained, and the number of passengers at that hour for that bus stop. In addition to this, we have the list of names of bus stops along with their longitudinal and latitudinal coordinates, ordered bus route definitions for the regular route and queries from the WOHIN DU WILLST service which includes the location and time of both the source and destination. To determine whether the queries correlate to the actual travel time of the passengers, we will do a proof-of-concept using the null hypothesis to decide whether or not we want to use them in our feature selection. All of these files have been provided in our problem statement.

For external data, Meteostat [2] provides the weather statistics for Passau for the year 2019 on an hourly basis. We plan to take the average temperature in celsius, precipitation in millimetres, and weather condition codes [4] from this data source. We used Rapid’s API key [3] for this purpose which lets us perform 500 requests a month at the rate limit of 3 requests per second.

3.2 Data Decomposition

Time series data generally consists of level, trend, seasonality, and residual components. The general change in direction as time progresses is referred to as a trend for example worldwide population tends to increase with time. Seasonality, on the other hand, is the periodic change in behavior for example toy sales tend to increase during the Christmas season every year. Residual components do not follow any pattern.

To gather insights, we decompose our data into these individual components. There are two types of decomposition models – additive and multiplicative. Additive models have a constant amplitude whereas multiplicative models have an amplitude that increases with time. Pseudo-additive models are a combination of these two and are used when the data has many zeroes.

$$AdditiveFunction = Trend + Seasonality + Residual$$

$$MultiplicativeFunction = Trend * Seasonality * Residual$$

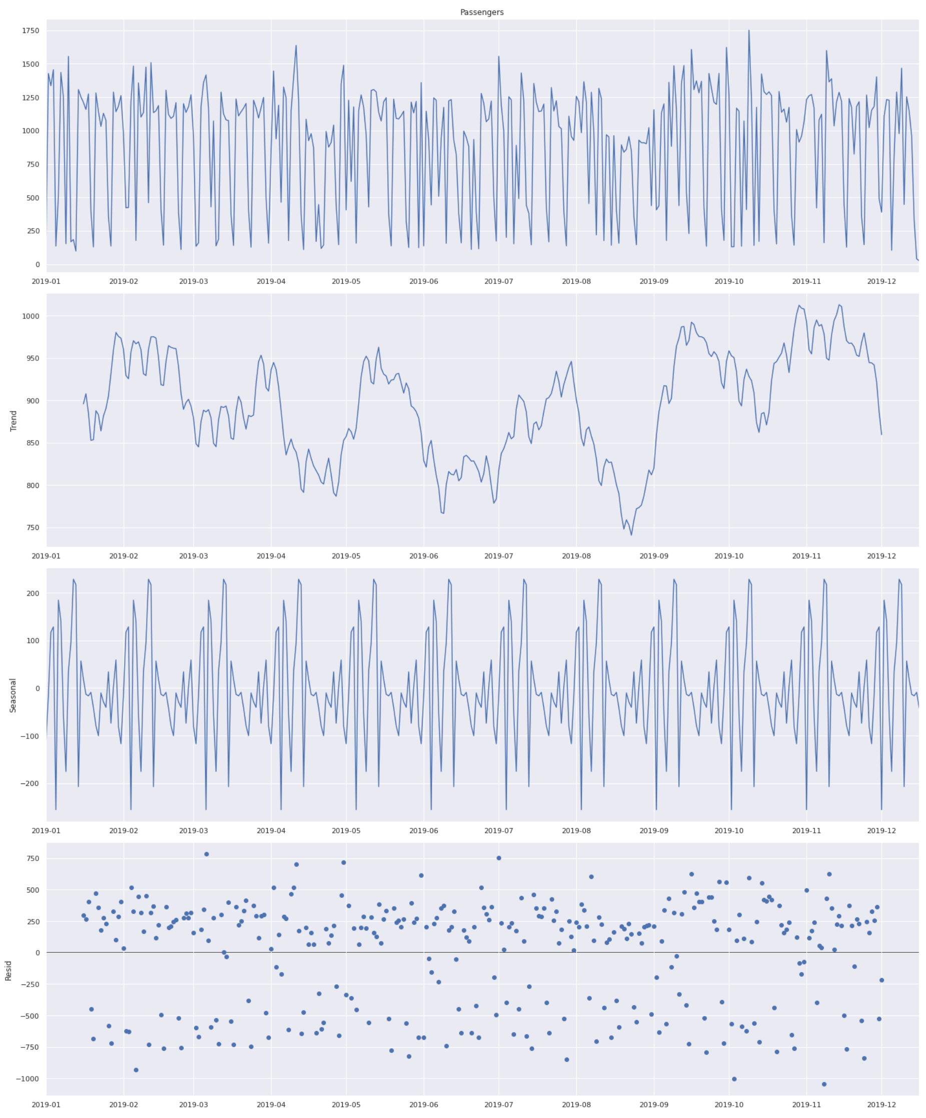

Fig. 1. daily sold tickets for all bus stops. ADF test: p-value = 0.000005, KPSS test: p-value = 0.100000

By breaking the time series into seasonality, and trends, we gain a better understanding of whether statistical or machine learning models will produce better results.

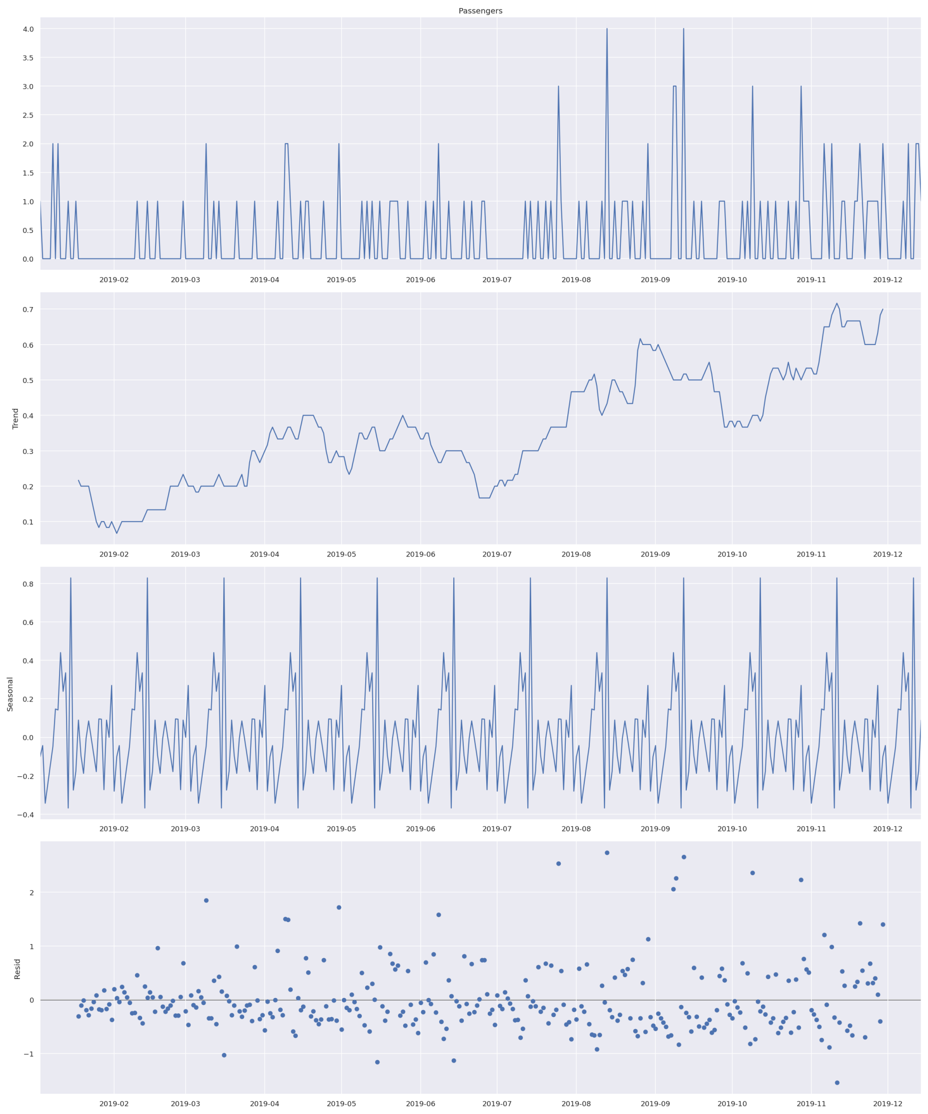

Fig. 2. daily sold tickets for bus stop 5400 Pölzöd. ADF test: p-value = 2.552400e-30, KPSS test: p-value = 0. 0.010000

The existence of some patterns like trends and seasonality that make a time series non-stationary are the factors that affect the selection of appropriate models for better forecasting. For this purpose, we used ADF and KPSS tests as well as seasonality decomposition on daily sold tickets for all bus stops (whole data set) Fig.1, and daily sold tickets for each bus stop separately Fig.2.

KPSS and ADF tests on the whole dataset show that our data set is stationary while looking at the results from the data for bus stop 5400 - Pölzöd, the KPSS test shows that, for this bus stop, we have non-stationary behavior. The p-value of the KPSS test is less than 0.05 and we reject the null hypothesis which means the data for this bus stop is not stationary. Although in some parts of our data set and in the whole time series data set analysis we can see stationary behavior, non-stationary components have been seen in other parts of the data set, as we see for bus stop 5400 - Pölzöd. Therefore, as we are going to apply our models to the whole data set, we consider non-stationary properties of our data set that lead to selecting specific models like the SARIMA model instead of the ARIMA model.

The Autocorrelation Function (ACF) drops quickly for stationary data and for non-stationarity data, it decreases slowly, and the auto-correlation value is large and positive.

3.3 Data Munging

We observed no null values in our data set except for the date of 2019-31-03 where we did not have weather data for the 2nd hour of the day. Mean for the 1st and 3rd hour of the day was used to estimate the values for temperature, rain, and weather condition code.

3.4 Data Analysis

In this section, we look at various graph plots to achieve a better understanding of the distribution of our features. We try to gain an insight into what affects the commuters' decision to travel by bus.

On taking a close look at the distribution of passengers, we can notice in the first graph of Fig.3 that commuters start taking the bus just before 0500 hours, which is understood since that is the time of the first bus. The peak hours are at 0600 hours which might be due to the students and employees who form the main chunk of our users. This is further confirmed by the second graph which shows clearly that the demand for buses drops significantly on weekends. The third graph shows the distribution of travelers throughout the entire year of 2019 and we can notice that it drops in summer may be due to the heat and rises continuously after the fall season.

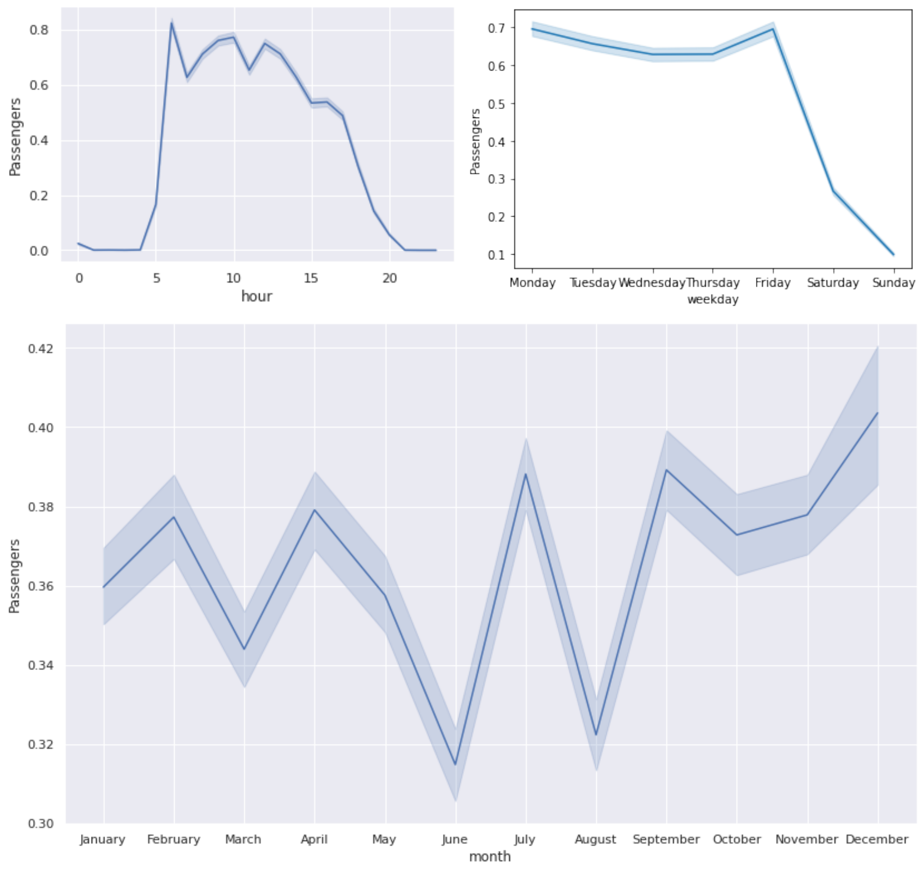

Fig. 3. 1) Number of passengers per time of the day 2) Number of passengers per day of the week 3) Number of passengers in each month

As you can see in Fig.4, we have a large number of records for 0 passengers. We are analyzing our data on an hourly basis, this explains the large number of zero passengers and makes our dataset sparse. This also shows that a large number of bus stops are underutilized on an hourly basis. Bin 3 in the figure is for more than 2 passengers.

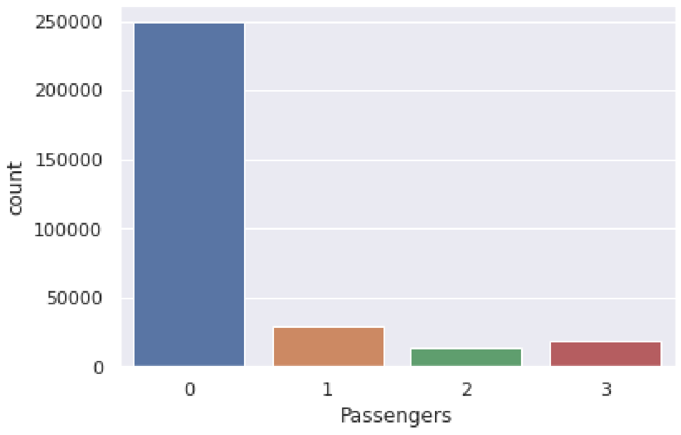

Fig. 4. Passenger count

The box plot in Fig.5 further shows us that passengers have traveled throughout the entire day and thus the distribution is evenly spread over the entire day, giving a mean of noon. Also, after 2100 hours or the 21st hour of the day, very few to no passengers board the public vehicle.

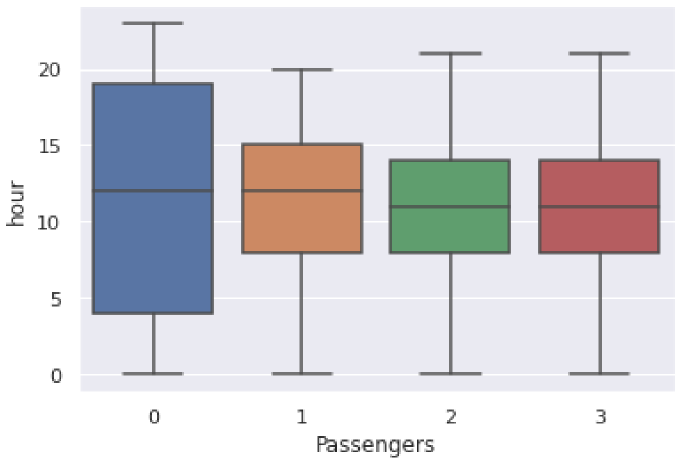

Fig. 5. Passenger vs hour of the day

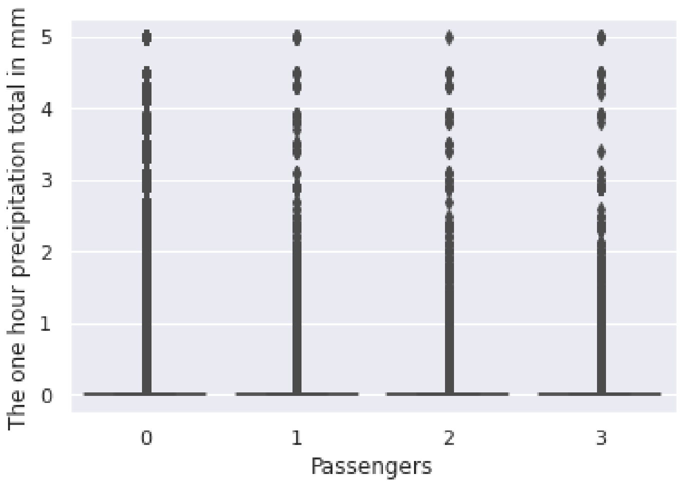

Fig. 6. Passengers at different precipitation levels

In Fig.6, we see that rain has affected the frequency of travelers. As the downpour of rain increases, lesser individuals are motivated to go out of their respective homes and use public buses.

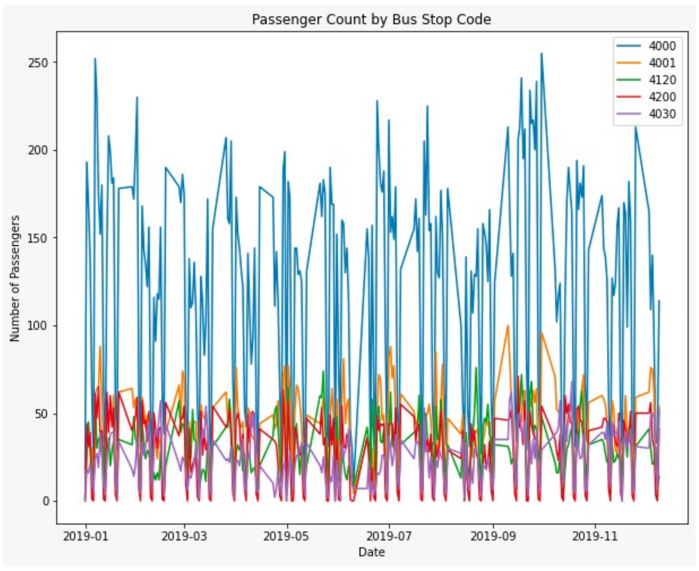

Fig. 7. 5 bus stops with the most number of passengers

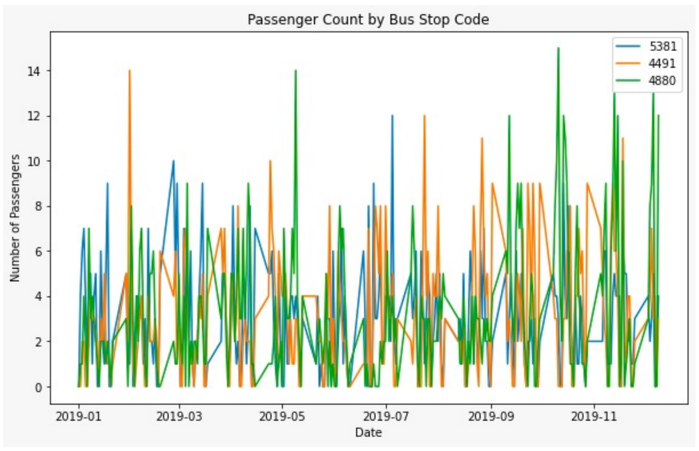

Fig. 8. 3 bus stops with the least number of passengers

From the line plots 7 and 8, it is noticeable that Passau Hauptbahnhof (Hbf) was the busiest bus stop with hundreds of commuters every month. Passau ZOB, was also busy but much less than the Hbf, in fact, none of the other bus stops come even close to the Hbf in terms of passenger volume. On the other hand, Passau Lindau, Tiefenbach, R.-Töpfl-Str. and Franklbach were the lowest in passenger volume.

The correlation matrix in Fig.9 does not show a strong correlation in either the positive or negative direction between our features and the target variable. But a small positive correlation exists between *route_count* and our target feature *Passengers*. Since features are mainly uncorrelated, Naïve Bayes should perform better but we will wait for the evaluation stage to come to a conclusion.

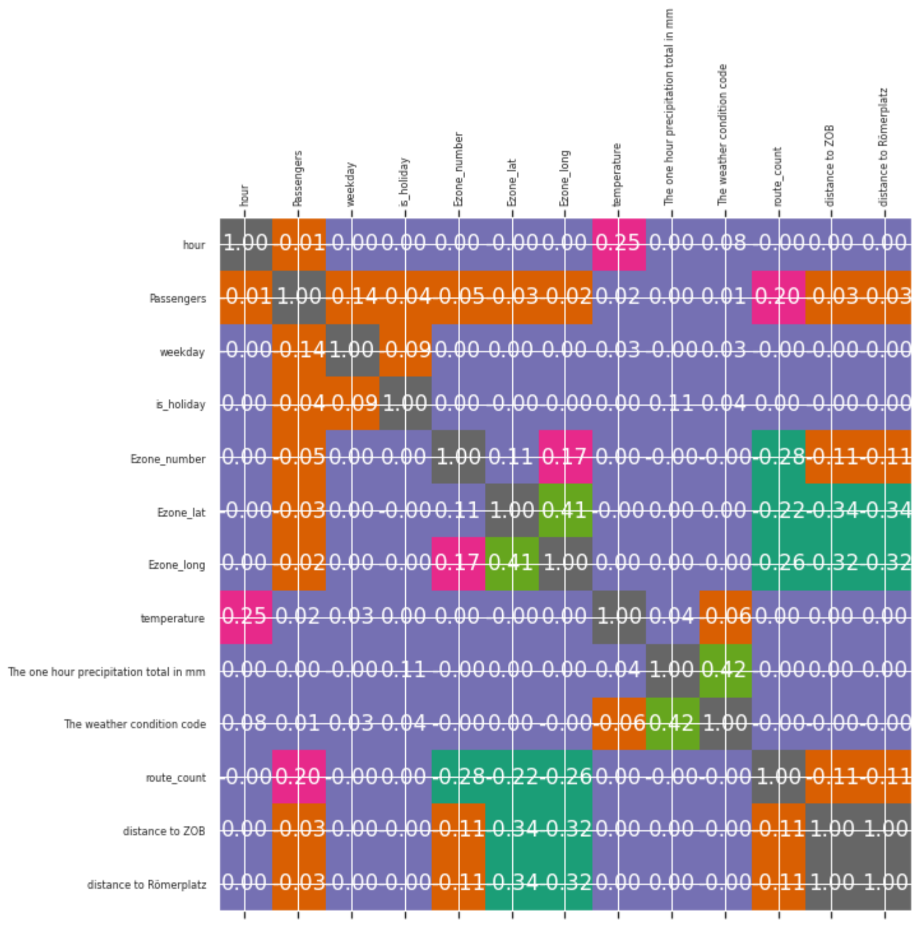

Fig. 9. Correlation Matrix

3.5 Pre-processing

We aim to make hourly forecasts using a prediction window of 12 hours using a persistent classifier as a baseline. We might change the size of the prediction window depending on the performance of our models. We split the data into 3 weeks for training and 1 week for test and will use the error distribution to show how good this is. If we choose a longer timeframe for training, then we might face the issue of omitting some relationships. For example, on choosing a training sample of January to September, the model might not be able to capture the trend of the last few months when there is extreme cold and also a festive season.

We will also study our prediction dependence on past covariates. We might use them at different scales such as daily or weekly aggregates since the normalization of the aggregates of past covariates on an hourly basis will be a challenge.

We could also use backward or forward selection for features. For dimensionality reduction, we use Principal Component Analysis (PCA) with one-hot encoding on the dates. PCA can reduce dimensionality by standardizing the variable to a comparable scale, computing the covariance matrix, and then determining the principal components using eigenvectors. Principal components are just combinations of initial variables which have information compressed within them in decreasing order.

Another method to reduce the number of variables is to eliminate travel of fewer than 8 minutes.[12] This forms the basis for the assumption that a travel time of less than 8 can be covered on foot and does not constitute travel. If required, we might use this theory to reduce the number of data points recorded.

3.6 Feature Engineering

From the files provided by Deutch Bahn, we will need to extract the date, and the hour, determine what day of the week it corresponds to and whether that day was a holiday or not. The *hour* and *weekday* features are categorical and nominal. 0 on a *weekday* would correspond to Monday and 6 would correspond to Sunday respectively. Holidays might affect the passenger frequency which was provided by the holidays' package. The *is_holidays* attribute we used is a categorical feature with boolean values.

The EZone (Einstiegsszone) column of the regular routes raw file needed to be cleaned because it contains a combination of numbers and the name of the bus stops. We extracted the bus stop number as the *Ezone_number* in our code. This acts as a unique identifier for our bus stops.

We also appended a column for calculating the number of times a bus stop comes in all the routes or in other words how many routes have a stop at that particular bus stop. This is recorded in the *route_count* feature. For determining this particular feature, we used the *name_sanitized* as a common column in both the bus stops and regular route definition files using the reasoning that one bus stop can be part of multiple routes. This provides additional information about a specific bus stop.

Another feature that could be helpful is the distance to popular landmarks like the City Center. The thought behind this is that the bus stops tend to be more crowded near these landmarks and should be considered. We used two landmarks - Zentraler omnibusbahnhof or ZOB (the bus stop at the city center) and Römerplatz and recorded the distance in kilometeres. More bus stops were not added because we did not want features that were highly correlated with each other, which would lead to redundancy. The calculation for the distance was done using geodesic distance using the geopy package. Geodesic distance is the shortest distance between two points on a given surface.

4 Classification Methods

Classification problems can be divided into various categories depending on the problem at hand. Furthermore, a variety of methods can be used ranging from statistical and econometric methods to deep learning. The division or subdivision of methods is necessary so that the problem can be better understood and a more appropriate method can be applied rather than following a generic approach.

Machine learning methods can be roughly divided into econometric or artificial intelligence methods and some researchers have combined these two into hybrid models as shown by Hastie et al. [10]. Another division can be based on static or dynamic data. Static, being observations of an object in a specific period, and dynamic, when the observations are recorded in a continuous time frame. We finalized five algorithms that we wanted to try and evaluate their performance.

4.1 Decision Tree

Decision trees are one of the tools that can be used for classification problems. Decision trees consist of internal nodes, which perform a test of one of the attributes, branches, which convey the outcome of the test, and leaves, which represent the outcome of the decision tree. In order to predict the value of a target variable, we train a model by inferring simple decision rules from the data. Different algorithms can be used to build decision trees, all based on greedy approaches. Iteratively, attributes are selected from an empty tree to split the data at each step. Records belonging to the same class are included in the leaf containing the class label. Determining the test condition, choosing the best attribute, and determining the splitting condition are important components of decision trees. To find the best test condition, various metrics are used, such as Gini Impurity Index and Information Gain Ratio.

4.2 Random Forest

Random Forest is an ensemble learning method proposed by Leo Breiman in 2001. The algorithm creates a forest (combination) of decision trees where each tree decides on the random subset of features it receives and the outcome is split into one or more leaf nodes until the final prediction is made. The category which gets the majority vote becomes our predicted class. It is a powerful method in high-dimensional machine learning classification and regression. It works well on large datasets with high dimensionality and can capture both linear and non-linear relationships. It is also able to retain its accuracy even in the absence of certain data. In the case of time series prediction, we need to transform our data set into supervised learning forecasting.

The ensemble method is a technique of making predictions using a bunch of different models. The intuition behind this is to improve flexibility and prediction performance. There can be multiple ways to combine different models but the three main ensemble techniques are bagging, boosting, and stacking. The bagging method trains the different models (decision trees in our case) in a parallel way on different random samples of the training data. The class which gets the majority vote becomes the final prediction.[14].

The ensemble process of Random Forest does feature selection automatically. Random Forest consists of several hundred decision trees which take a random sample of the objects and the attributes. This assures that the trees are decorrelated and that they do not overfit.

4.3 Adaboost with decision tree

Adaptive Boosting (AdaBoost), an ensemble learning technique, is a boosting classifier that was proposed by Yoav Freund and Robert Schapire in 1996[9]. In this iterative algorithm, multiple classifiers are combined to build a classifier with high accuracy. In each iteration, weights are set to the classifiers and the data sample is trained to predict unusual observations accurately while the new weights depend on the past iteration and ensemble formation. Adaboost can employ many instances of the same classifier with various settings. As a result, it is possible to create nonlinear classifiers from a previously linear classifier. We may test using Adaboost if the performance of many tiny decision trees can outperform that of a single decision tree.

Boosting trains the models sequentially where the misclassified data points from the previous step are used by subsequent models to reduce prediction errors. Stacking is another ensemble approach where classifiers are stacked on top of each other. This combines different classifiers into a meta-classifier. Here, it is desirable to have different models as ensemble members, so that they make different assumptions. The predictions of one classifier are fed into another and then finally into the meta classifier.

4.4 Naïve Bayes

Naïve Bayes is a classification method built on the Bayes Theorem and predicated on the idea of predictor independence. Bayes Theorem says that new evidence does not completely determine your beliefs in a vacuum, it should update prior beliefs. Naïve Bayes works well when features are not correlated and hence there is less training data needed. It manages data that is continuous and discrete. Concerning the number of predictors and data points, it is quite scalable. It is quick and may be utilized to make predictions in the present. However, it does not work well when features are correlated.

4.5 SARIMA

The Seasonal Autoregressive Integrated Moving Average model (SARIMA) is introduced by Box et al. (1967)[7] as a modification of the ARIMA model to consider the seasonality of a time series dataset. The SARIMA model has the multiplicative form SARIMA(p,d,q)x(P, D, Q)s (Hipel and McLeod 1994)[11]. The first part shows non-seasonal parameters and the second part shows the seasonal parameters and is calculated by the formula:

$$\varPhi(L^s)\phi(L)\varDelta^d\varDelta_s^D y_t = \theta_0 + \varTheta(L^S)\theta(L)\epsilon_t$$

where, p is the order of non-seasonal Autoregressive,
d is the number of regular differencing,
q is the order of non-seasonal MA,
P is the order of seasonal autoregression,
D is the number of seasonal differencing,
Q is the order of seasonal MA,
s is the seasonal length,
L is the lag operator,
ϵ_t is a Gaussian white-noise process with mean zero and variance σ^2.
$\varDelta^d$ is the difference operator,
$\varDelta_s^D$ is the seasonal difference operator,
ϕ is Autoregressive Parameters,
δ is the Moving Average Parameter,
$\varPhi$ is Seasonal Autoregressive Parameter,
$\varDelta$ is Seasonal Moving Average Parameter.
Robustness is the property that the seasonal part adds to the ARIMA model.

4.6 Overall expectation

The "no free lunch" concept tells us that there is no single algorithm, that performs on average better than other algorithms over all the distributions. Hence, we look at the implementation of the said models and deal with the associated challenges that come along with it.

Scikit-learn provides a classifier module that is straightforward and effective for data scientists who want to utilize the classification models listed above in Python. In later stages, we will do a parameter distribution and permutation to determine feature importance. One of these features will be past covariates and we will decide how many of them we want to use. A single-step forecast can also be used as long as it is accurate. If past covariates have a major influence on the result, we will switch to a multi-step forecast.

5 Experiments

5.1 Evaluation methodology

The objective of any forecasting model is to make prediction with high accuracy. We evaluate the performance of our models through the study of a time series dataset. Classification and Regression are two methods that we can use to make a forecast. As we choose classification algorithms, evaluation metrics like MSE, MAE, and R square are not used here. Also, because k-fold cross-validation is not the appropriate method for a time series dataset, as it sequential and we cannot divide it into folds, then we use a customized splitting method in such a way that for the 12 different folds, each consisting of 3 weeks for training and 1 week for forecasting. We hence plan to use a confusion matrix, accuracy, f1-score, precision and recall with macro and weighted averaged techniques. All these statistical results are shown for each model but the measurement that we used to compare our models is macro average of f1-score because there is not equal support for each classes (imbalance data set) and the weighted average is not a good choice for model comparison. Using statistical tests like p-value and t-value we can easily track a description after fitting the model to be used to evaluate our models. These metrics are the methods that we use to compare different models to choose the most effective classifier for our dataset. Additionally, since we use only the information about 50 bus stops, one of the issues that we can discuss is fairness.

5.2 Supervised Learning

Forecasting time series can be viewed as a supervised learning task. When you reframe your time series data this way, you are able to apply standard linear and nonlinear machine learning algorithms to your problem. We can restructure a time series dataset into something similar to a supervised learning problem given a sequence of numbers. Previous time steps can be used as input variables, and the next time step can be used as an output variable. We will use a 12 hours window in order to transform our data into a supervised learning task. We will also train our models on the dataset without using a window and considering the fact that this is a time-series dataset. This will allow us to get a better understanding of time-series forecasting and the importance of previous time steps.

Decision tree Data from the regular route dataset was used to train a decision tree model using the scikit-learn library. The Gini Impurity index was used as the basis for training the decision tree. The max depth was optimized using the Optuna Framework[6], which automates the search for hyperparameters. This optimization is based on f1-score. The max depth of 18 was chosen by the framework as the best value.

Decision tree with considering previous timesteps Optuna is a framework for python that allows its users to optimize their hyperparameters. It uses both search and pruning strategies to find the best values for the selected hyperparameters. In order to use Optuna only one function is needed which Optuna will try to either minimize or maximize. For our case, the hyperparameter of max depth was chosen to be optimized with this framework.

The model achieved a macro average on f1-score of 0.41 on the test set. The table 1 presents other statistical metrics of this model.

If we take a look at the confusion matrix 10, we can see that 0 is mostly predicted correctly. 1 and 2 have been predicted correctly for the 42% and 34%

	precision	recall	f1-score	support
0	0.97	0.69	0.80	88868
1	0.17	0.42	0.24	9726
2	0.11	0.34	0.16	4397
3	0.36	0.52	0.42	6209
macro avg	0.40	0.49	0.41	109200
weighted avg	0.83	0.64	0.71	109200

Table 1. Decision Tree model

respectively. Almost 30% of time 1 and 2 have been predicted interchangeably. For the label 3+, 52% of time the have been correctly classified.

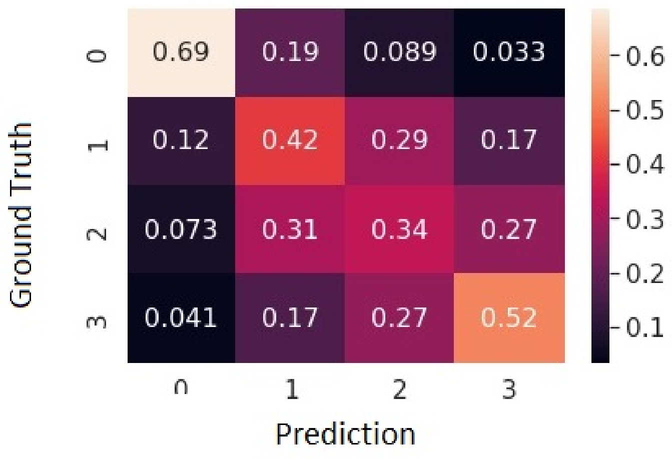

Fig. 10. Confusion matrix of Decision Tree model

The decision tree model was able to identify the most important features that contributed to the predictions. The most important feature based on permutation feature importance was the value of passenger count of 12 hours step before.

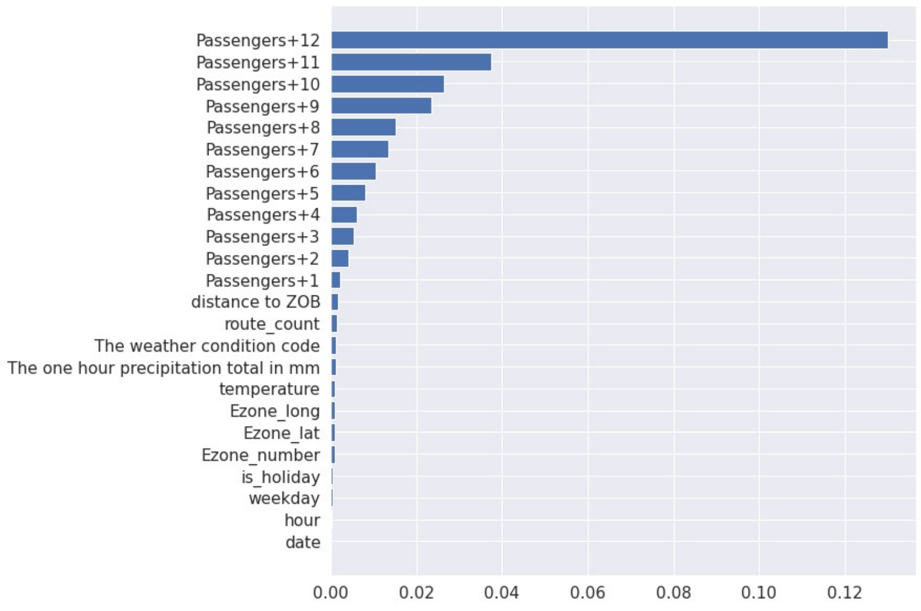

Fig. 11. Permutation Feature Importance of Decision Tree model

Decision Tree without considering previous timesteps Another decision tree model was trained on the dataset without considering previous timesteps. The table 2 presents the results of this model. The most important feature based on permutation feature importance was the distance to ZOB.

	precision	recall	f1-score	support
0	0.85	0.99	0.91	88868
1	0.24	0.01	0.02	9726
2	0.17	0.00	0.00	4397
3	0.51	0.44	0.48	6209
macro avg	0.44	0.36	0.35	109200
weighted avg	0.75	0.83	0.77	109200

Table 2. decision Tree model without considering previous timesteps

If we take a look at the confusion matrix12, we can see that 0 is mostly predicted correctly. But the other labels are almost classified as 0. This means that our model almost always outputs 0.

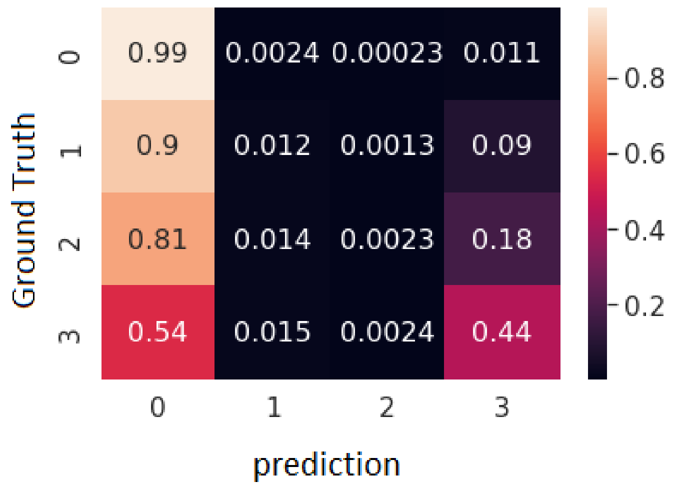

Fig. 12. Confusion matrix of decision tree model without considering previous timesteps

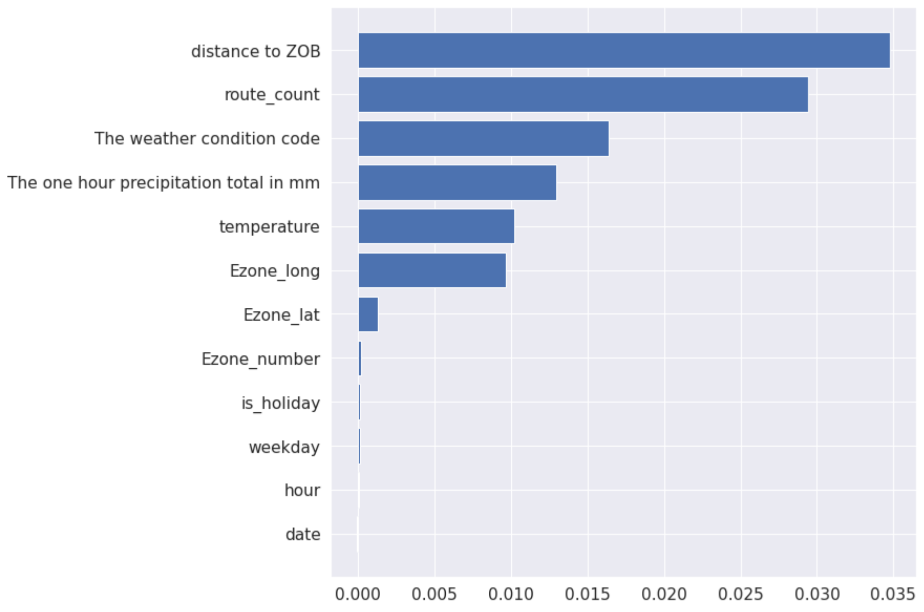

Fig. 13. Permutation Feature Importance of decision tree model without considering previous timesteps

Adaboost Using the AdaBoostClassifier and DecisionTreeClassifier provided by scikit-learn library, an Adaboost model was trained on the regular routes dataset using the decision tree as the base estimator. With the help of the Optuna framework, we optimized 45 estimators and max depth of 3 of the decision trees.

We evaluated this model using various statistical metrics. The macro average on f1-score of the model was 0.43. Other metrics are shown in the table 3.

	precision	recall	f1-score	support
0	0.98	0.70	0.81	88868
1	0.18	0.46	0.26	9726
2	0.12	0.29	0.17	4397
3	0.39	0.62	0.48	6209
macro avg	0.42	0.52	0.43	109200
weighted avg	0.84	0.65	0.72	109200

Table 3. AdaBoost model

If we take a look at the confusion matrix14, we can see that 0 is predicted correctly 70% of the time. 1 has been predicted correctly for the 46%, while 2 has been mistaken for 1 or 3+ almost 70% of time. For the label 3+, 62% of the time they have been correctly classified.

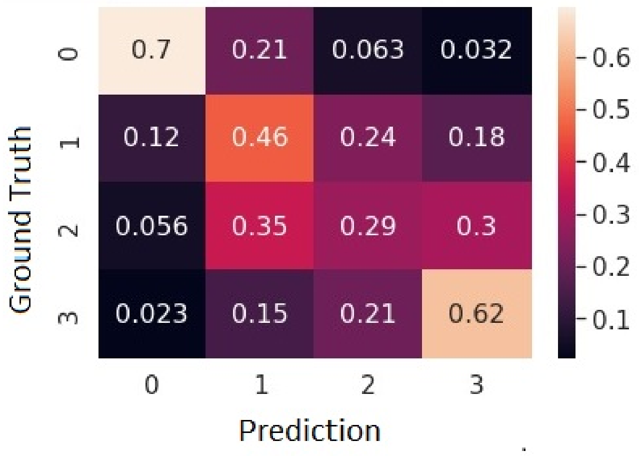

Fig. 14. Confusion matrix of AdaBoost model

It was possible to identify the most important features that contributed to predictions using the Adaboost model. Based on permutation feature importance, the value of passenger count for 12 hours before was the most important feature.

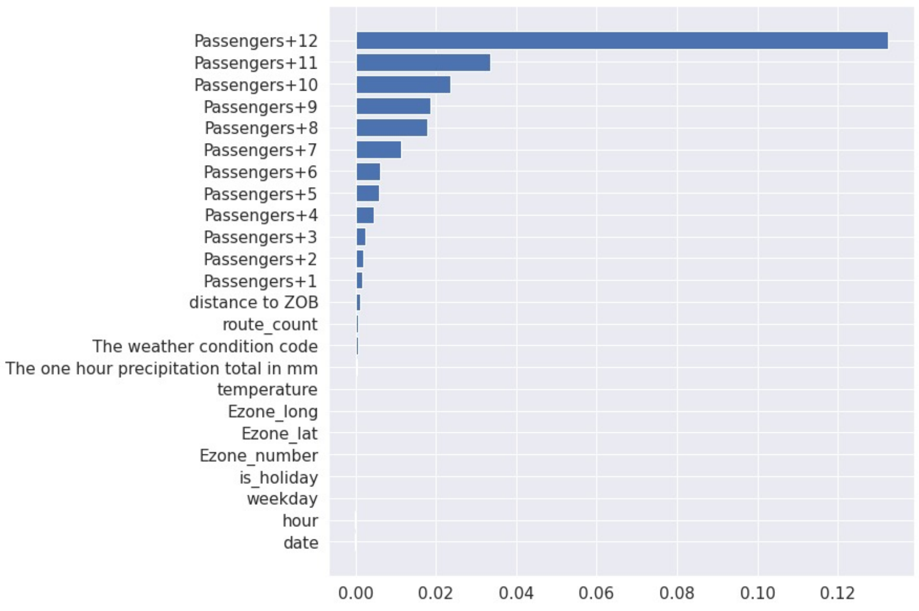

Fig. 15. Permutation Feature Importance of AdaBoost model

AdaBoost with a decision tree is an ensemble method that combines the strengths of a decision tree and boosting algorithm. The decision tree as a base estimator is able to capture non-linearity and interactions between features, while the boosting algorithm is able to improve the model's accuracy by giving more weight to misclassified samples. This is particularly useful for time-series datasets as they often contain complex patterns and interactions between features.

Random forest A Random forest model was trained using the scikit-learn library on the regular routes dataset. For the optimization of hyperparameters, we used the Optuna framework which outputted a model with 71 estimators and a max depth of 10.

The model achieved a macro average on f1-score of 0.42 on the test set. Other statistical metrics were also used to evaluate the performance of our model (table 4)

As we can see from the confusion matrix16, 0 is mostly correctly predicted (almost 70%). In half of cases, 1 has been predicted correctly and 2 has been

	precision	recall	f1-score	support
0	0.97	0.69	0.80	88868
1	0.17	0.51	0.26	9726
2	0.13	0.24	0.17	4397
3	0.40	0.58	0.47	6209
macro avg	0.42	0.50	0.42	109200
weighted avg	0.83	0.65	0.71	109200

Table 4. Random forest model

mistaken for 1 and 3+ of almost 70% of time. There was 58% correct classification of labels 3+.

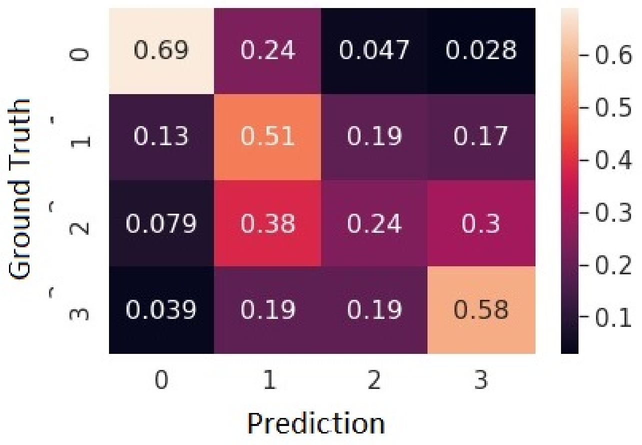

Fig. 16. Confusion matrix of Random forest model

The Random forest model was able to identify the most important features that contributed to the predictions. The most important feature based on permutation feature importance was the value of passenger count of 12 hours step before.

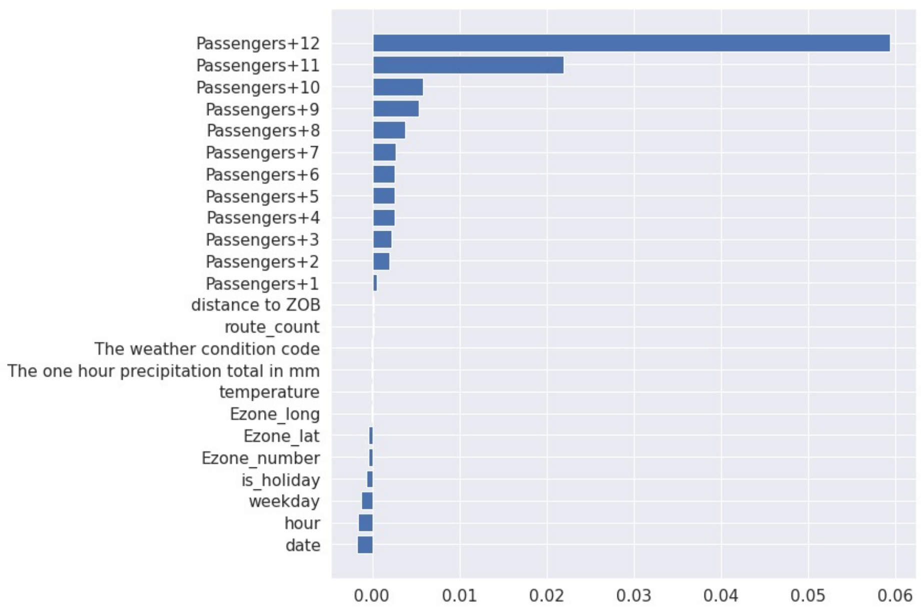

Fig. 17. Permutation Feature Importance of Random forest model

One of the key advantages of Random forest models is their ability to handle large datasets with a high number of features. This is particularly useful for time-series datasets which often have a large number of attributes. Random Forest also provides feature importance, which can be used to identify which variables have the biggest impact on the predictions.

Naïve Bayes We trained two Naïve Bayes models using the scikit-learn library. The model was trained on regular routes and on the on-demand datasets. Both of them are GaussianNB. GaussianNB is a classification algorithm in the Naïve Bayes family. It is called ”Gaussian” because it assumes that the continuous features in the data follow a normal distribution. The normal distribution, also known as the Gaussian distribution, is a probability distribution that is commonly used to model continuous data.

Firstly, we will review the results of the regular routes. This model achieved a macro average on f1-score of 0.35 on the test set. Furthermore, you can find the other statistical metrics in the table 5.

According to the confusion matrix18, 0 is mostly predicted correctly. In most cases, 1 and 2 have been mistaken for 0. There was 53% correct classification of labels 3+.

	precision	recall	f1-score	support
0	0.88	0.88	0.88	88868
1	0.18	0.15	0.16	9726
2	0.09	0.01	0.02	4397
3	0.26	0.53	0.35	6209
macro avg	0.35	0.39	0.35	109200
weighted avg	0.75	0.76	0.75	109200

Table 5. Naïve Bayes model trained on regular routes

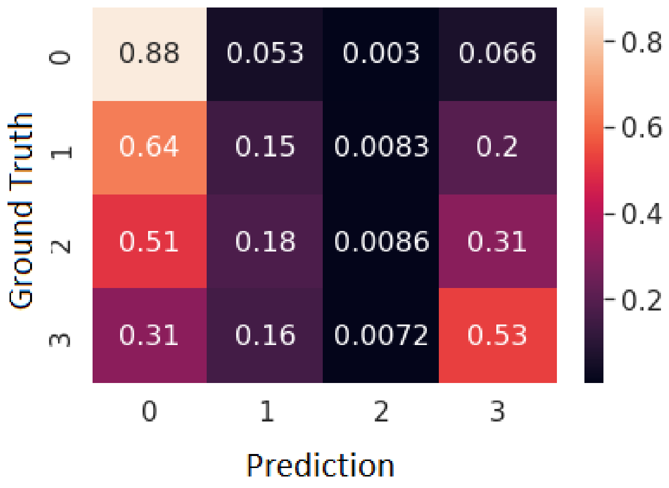

Fig. 18. Confusion matrix of Naïve Bayes model trained on regular routes

Next, we will discuss the on-demand results. The macro average on f1-score of this model was 0.27 on the test set. Additionally, the table 6 contains other statistical metrics.

	precision	recall	f1-score	support
0	0.94	0.90	0.92	88868
1	0.07	0.07	0.07	9726
2	0.03	0.05	0.03	4397
3	0.03	0.14	0.05	6209
macro avg	0.27	0.29	0.27	109200
weighted avg	0.88	0.85	0.87	109200

Table 6. Naïve Bayes model trained on on-demand

The confusion matrix19 indicates that 0 is mostly predicted correctly. However, the other labels are almost classified as 0. Thus, we almost always get 0 as an output from our model.

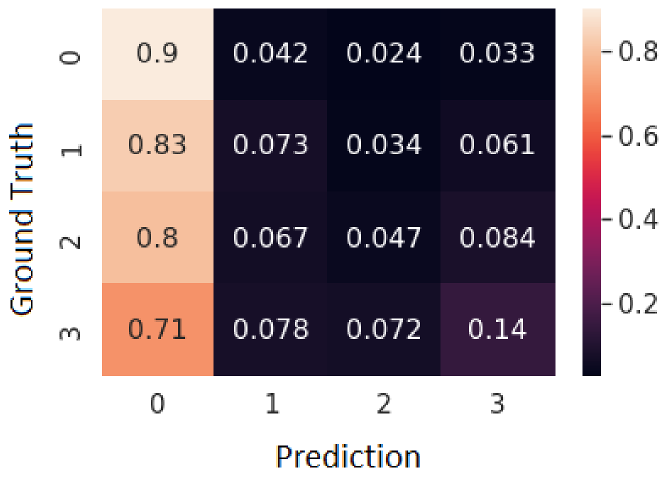

Fig. 19. Confusion matrix of Naïve Bayes model trained on on-demand

Naïve Bayes is a simple, fast, and easy-to-implement algorithm that can be used for classification tasks. It is particularly useful when the dataset has a large number of features, as it makes the assumption of independence between the features. This assumption is often not met in time-series data, where there are strong dependencies between features. However, despite this limitation, the Naïve Bayes model can still perform well on time-series datasets, especially if the dataset is preprocessed to remove dependencies between features.

SARIMA SARIMA, which stands for Seasonal AutoRegressive Integrated Moving Average, is a time-series forecasting model that is used to model and predict future values based on past observations. It is an extension of the ARIMA model, which stands for AutoRegressive Integrated Moving Average, and adds a seasonal component to the model. The model was trained using the SARIMAX function, with the parameters (p,d,q)(P,D,Q)m, where p,d,q are the order of the non-seasonal component and P,D,Q,m are the order of the seasonal component.

In order to get the optimal values for these parameters Auto Arima, a class provided by pmdarima, was used. Auto ARIMA model generates p, d, and q values that are appropriate for the dataset in order to provide better forecasts. As was discussed in previous sections, our dataset is overall stationary but for some of the bus stops, non-stationary behavior can be seen. Therefore, in order to have the best result 2 models were trained to see what difference it makes if we treat our dataset as if it was not stationary.

First, we will discuss the results of assuming the data was stationary. The model has a macro average on f1-score of 0.29. The table 7 shows the model's analysis results.

	precision	recall	f1-score	support
0	0.87	0.94	0.90	88868
1	0.18	0.20	0.19	9726
2	0.09	0.04	0.06	4397
3	0.84	0.00	0.01	6209
macro avg	0.50	0.30	0.29	109200
weighted avg	0.77	0.79	0.75	109200

Table 7. Sarima model with stationary dataset hypothesis

The confusion matrix20 indicates that 0 is mostly predicted correctly. However, the other labels are mostly classified as either 0 or 1.

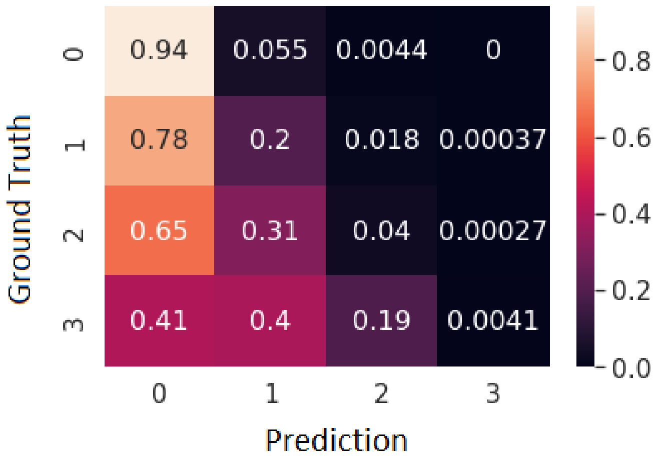

Fig. 20. Confusion matrix of the stationary dataset hypothesis

Second, an analysis for the case of the not stationary dataset is given. This time our model had a macro average on f1-score of 0.30. The results of other statistical metrics are shown in the table 8.

As can be seen from the confusion matrix 21, 0 is mostly predicted correctly. Other labels, however, get mistaken with either 1 or 0.

	precision	recall	f1-score	support
0	0.87	0.94	0.90	88868
1	0.18	0.20	0.19	9726
2	0.09	0.05	0.06	4397
3	0.77	0.03	0.06	6209
macro avg	0.48	0.30	0.30	109200
weighted avg	0.77	0.78	0.76	109200

Table 8. Sarima model with non-stationary dataset hypothesis

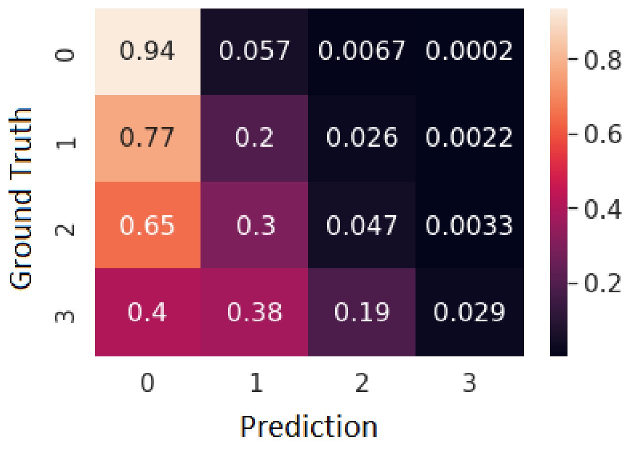

Fig. 21. Confusion matrix of the non-stationary dataset hypothesis

Additionally, as part of our study, we trained another model to determine the difference between multivariate and univariate time series. An analysis of the model trained on multivariate data is presented in the following table. Since training the multivariate model is very time-consuming, we only evaluate our model on data from 1 bus stop. The chosen bus stop is 4000 Passau, Hbf since this bus stop is the busiest bus stop in our dataset. It also is stationary based on KPSS and ADF tests. The macro average on f1-score of the multivariate model was 0.37. Other statistical metrics' results are shown in table 9.

From the confusion matrix22, we can understand that our model similar to the other models classifies the 0 correctly. Labels 1 and 2 will either be correctly classified or classified as the other. Label 3 gets confused by our model with 2.

	precision	recall	f1-score	support
0	0.91	0.82	0.86	836
1	0.15	0.34	0.20	125
2	0.06	0.39	0.11	101
3	0.94	0.19	0.31	762
macro avg	0.51	0.43	0.37	1824
weighted avg	0.82	0.50	0.54	1824

Table 9. Sarima multivariate model

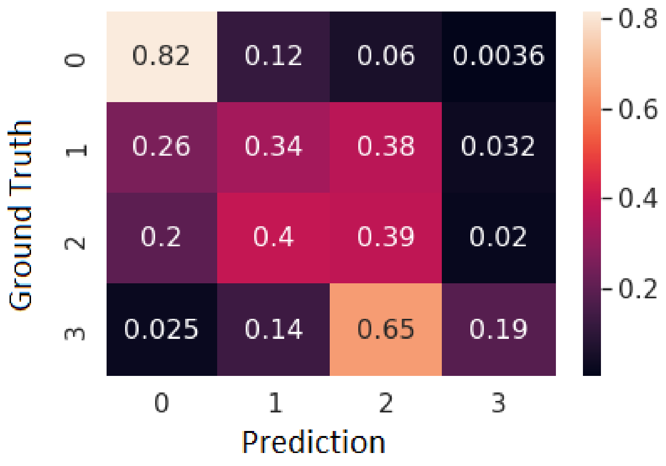

Fig. 22. Confusion matrix of the SARIMA multivariate model

The walk-forward method is a technique used to evaluate time-series models. It involves using a fixed window of historical data to train a model and then using that model to predict future values. The window is then moved forward by a certain number of time steps, and the process is repeated. In this way, each time step can be forecasted as best as possible. Since we faced the same problem as the Multivariate model due to time restrictions, we only evaluated our model on 1 bus stop(Passau, Hbf). The results of the walk-forward method show a macro average of f1-score 0.41, the table 10 presents the results of more statistical metrics.

From the confusion matrix23, we can understand that our model similar to the other models classifies the 0 correctly. Labels 1 and 2 will either be correctly classified or classified as the other. Label 3 gets confused by our model with 2.

	precision	recall	f1-score	support
0	0.88	0.82	0.85	836
1	0.19	0.40	0.26	125
2	0.07	0.40	0.12	101
3	0.97	0.25	0.40	762
macro avg	0.53	0.47	0.41	1824
weighted avg	0.83	0.53	0.58	1824

Table 10. Sarima model with walk forward method for Passau Hbf

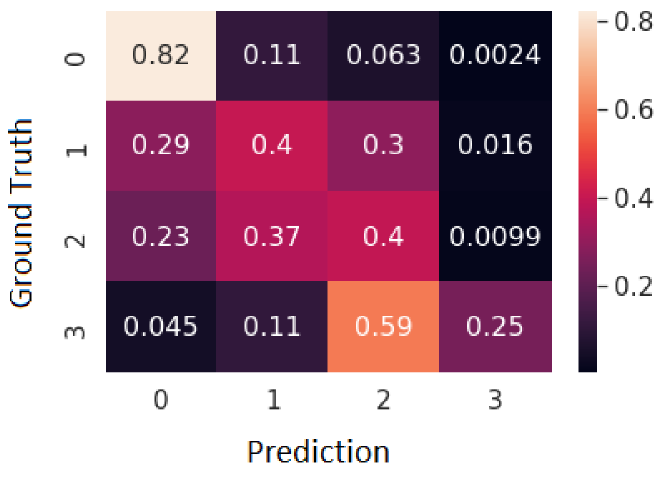

Fig. 23. Confusion matrix of walk forward method

One of the key advantages of SARIMA is that it can model both the trend and seasonality of time-series data. It also allows for the modeling of the residual errors, which can be useful in detecting and correcting any remaining patterns in the data.
The results for Sarima univariate for Passau Hbf are shown in table 11 and in Fig.24. However since SARIMA model takes a lot of time and we ran it for only one bus stop, its results cannot be compared to other models. Running it for all the other bus stops will be in our future work.

Fig. 24. Confusion matrix of SARIMA univariate for Passau Hbf

	precision	recall	f1-score	support
0	0.90	0.79	0.84	836
1	0.12	0.29	0.17	125
2	0.05	0.31	0.09	101
3	0.79	0.19	0.30	762
macro avg	0.46	0.39	0.35	1824
weighted avg	0.75	0.48	0.53	1824

Table 11. Sarima univariate for Passau Hbf

5.3 Results comparison

In this work, the performance and effectiveness of five classifier algorithms are compared. The classifiers DecisionTree, AdaBoost with Decision Tree, Random forest, Naïve Bayes were examined on both on-demand and regular routes dataset and SARIMA on the regular routes data set on the time series datasets collected by DB regio of the tickets sold [1]. We also made hourly forecasts using a prediction window of 12 hours using a persistent classifier as a baseline and predicted and classified a given bus stop into 0, 1, 2, and 3+ categories. Table 12 shows the f1-score of different classes, the macro average of f1-score on the regular routes dataset and considering the multivariate models and evaluating by splitting the data into 3 weeks for training and 1 week for tests and measuring the precision, recall, f1-score and macro and weighted average on precision and recall and f1-score. We can see these measures in detail in tables 1, 3, 4, 5, 9, 10.
Also, the results for SARIMA with the walk forward validation method and SARIMA multivariate model, and SARIMA univariate model for bus stop Passau Hbf are shown in table 13.

Table 15 shows the result for the SARIMA univariate model on the regular routes dataset, considering the stationary and non-stationary hypotheses.
Decision Tree, AdaBoost, Random Forest, and Naïve Bayes models are also applied to the on-demand dataset and we can see the results in table 14.

Multivariate models on regular routes dataset The results in 12 show that the macro average of f1-score for our best model, AdaBoost, is 0.43, and has better performance for classes 0 and 3 than for classes 1 and 2. Random forest, with a 1% lower macro average of f1-score than AdaBoost, is the second best model with the same results for classes 1 and 2 as AdaBoost. Although Decision Tree is the third-best model, its results are close to the two previous models. We can see Naïve Bayes on regular routes is the model that shows the worst performance on the regular routes dataset. Moreover, these models took less than 1 minute to run.
The considerable results of these models are that for all models classes, 0 and 3 are mostly classified correctly while classes 1 and 2 are misclassified. We can see these results in Fig.16, Fig.20, Fig.22 and 8. Based on the results, there is a

Model	f1-sore class 0	f1-sore class1	f1-sore class2	f1-sore class3+	macro average of f1-score
Decision Tree	0.80	0.24	0.16	0.42	0.41
AdaBoost	0.81	0.26	0.17	0.48	0.43
Random forest	0.80	0.26	0.17	0.47	0.42
Naïve Bayes for regular routes	0.88	0.16	0.02	0.35	0.35

Table 12. Multivariate models on regular routes dataset

possible relation between performance in a class and support. As classes 0 and 3 have better f1 scores because there are fewer samples of classes 1 and 2 which are less likely to cover an appropriate amount of the possible feature space or maybe they are just easier to predict. These results bring us some new ideas that we will discuss in the future works section.

SARIMA model for the bus stop Passu Hbf SARIMA model with the walk-forward method and SARIMA multivariate and univariate are the models which were trained for the bus stop Passau Hbf to compare these three methods for the SARIMA model. The main reason for selecting the bus stop Passau Hbf for comparing SARIMA with other models is that this bus stop is dissimilar from other bus stops as it has a high passenger volume. This bus stop is not representative of other bus stops but it is on the far end of the spectrum, which leads to more reliable predictions for classes 1 and 2, and 3+ since it provides more support for those classes in comparison to the other bus stops. The results in table 13 show that SARIMA with walk forward method has performed better than SARIMA multivariate and SARIMA univariate. Although the SARIMA with walk forward method ended up having good results, it must be considered that this model applied for the bus stop Passau Hbf and takes 900 times more time to be trained than Adaboost on all datasets so the result is inconclusive.

Model	f1-sore class 0	f1-sore class1	f1-sore class2	f1-sore class3+	macro average of f1-score	Running Time
SARIMA multivariate	0.86	0.20	0.11	0.31	0.37	15 minute
SARIMA with walk forward	0.85	0.26	0.12	0.40	0.41	95 minute
SARIMA univaiate for Passau Hbf	0.84	0.17	0.09	0.30	0.35	10 minute

Table 13. SARIMA model for the bus stop Passau Hbf

Confusion matrices of the SARIMA multivariate model, SARIMA with walk forward, and SARIMA univariate models on bus stop Passau Hbf show that these models classify class 0 most correctly. Labels 1 and 2 will either be correctly classified or classified as the other. Label 3 gets confused by our model with 2. Fig.22, Fig.23 and Fig.24 show the results in detail.

Multivariate models on on-demand dataset We also examined Decision Tree, AdaBoos, Random Forest, and Naïve Bayes models on the on-demand dataset and we can see the result in table 15. The results show that class 0 is predicted to be the best but there is not a satisfactory performance for other classes. Based on the macro average on the f1-score, we can see the Decision Tree model has the best performance and the Random forest has the worst.

Model	f1-sore class 0	f1-sore class1	f1-sore class2	f1-sore class3+	macro average of f1-score
Decision Tree on on-demand	0.82	0.20	0.11	0.10	0.31
AdaBoost on on-demand	0.75	0.14	0.10	0.09	0.27
Random forest on on-demand	0.69	0.13	0.08	0.10	0.25
Naïve Bayes on on-demand	0.92	0.07	0.03	0.05	0.27

Table 14. Multivariate models on on-demand dataset

Model	f1-sore class 0	f1-sore class1	f1-sore class2	f1-sore class3+	macro average of f1-score	Running Time
Sarima univariate with d = 0	0.90	0.19	0.06	0.01	0.29	75 minute
Sarima univariate with d = 1	0.90	0.19	0.06	0.06	0.30	75 minute

Table 15. SARIMA univariate on regular routes dadaset

Dataset	p-value of ADF test	p-value of KPSS test
All data set (all bus stops)	0.000005	0.100000
bus stops: 5400 Pölzöd	2.552400e-30	0.010000
bus stop: 4000 Passau Hbf	2.451390e-07	0.100000

Table 16. Time series behavior

SARIMA univariate model on regular routes dataset As the overall dataset shows stationary behavior, based on KPSS and ADF tests(table 16), we applied the SARIMA univariate model with d=1 and d=0 for non-stationary and stationary hypotheses respectively. The results are shown in table 15 and we can see that the model with the non-stationary hypothesis leads to better results but the improvement is only by 0.01. The confusion matrices 20 and 21 indicate that 0 is mostly predicted correctly and the other labels are mostly classified as either 0 or 1.

Decision Tree without previous time steps on regular routes dataset The Decision Tree model is also applied without the previous time steps feature on the regular routes dataset to examine the importance of this feature for the forecasting model and we can see in table 17 the model with this feature has better performance. The interesting result is an increase in the f1-score for classes 1 and 2 and a better data generalization in the presence of the previous time steps feature.

Model	f1-sore class 0	f1-sore class1	f1-sore class2	f1-sore class3+	macro average of f1-score
Decision Tree without previous time steps	0.91	0.02	0.00	0.48	0.35
Decision Tree	0.80	0.24	0.16	0.42	0.41

Table 17. Decision Tree with and without previous time steps on regular routes dataset

5.4 Discussion

We started our research with some research questions and concluded with the following results:

Time series behavior The KPSS and ADF tests as well as the seasonal decomposition functions were used to examine the behavior of the data set in terms of stationary properties. The issue we encountered was that the daily pattern analysis of the overall dataset produced stationary results. However, individual analysis of each bus stop revealed non-stationary results for some bus stops according to the KPSS test (the ADF test indicates stationary behavior for all bus stops). Table 16 shows the results for the overall dataset and the randomly selected stationary bus stops Passau Hbf and non-stationary bus stop Pölzöd. We can see there are different results for ADF and KPSS tests for the overall dataset and each bus stop separately. The possible issue might be that we are aggregating over more than one bus stop and different bus stops have different patterns and may have some constructive/destructive interference and each bus stop should be treated as a separate time series. Based on these results, we considered the dataset as a non-stationary time series dataset and used the SaRIMA model. Also, we trained the SARIMA model for a univariate case for d

= 0 (non-stationary case) and d = 1(stationary case) and compared the results as we see in table15. We can see the results are the same, more or less.

Feature Selection and Feature Importance In this research, we added some features to the original data set to examine the effects of these features on our prediction. The weather condition code, temperature, and the one-hour precipitation total in mm (for rain) feature to examine the weather condition impact, *is-holiday* and weekday features to examine holiday effect and distance to ZOB and *route_count* features to examine the landmark effects.

The correlation matrix in Fig.9 of the mentioned features shows neither a strong correlation in either the positive or negative direction between our features nor the target variable. But a small positive correlation exists between *route_count* and our target feature passengers.

Also, we used the permutation feature importance technique(PFI) because it can be applied to a trained model (quite cheap). The results for PFI of the model's Decision Tree, AdaBoost, and Random forest are shown in figures 11, 15, and 17. As we can see in these figures, the previous timesteps feature is the most important feature that follows with the distance to ZOB, *route_count*, and the weather condition code features. The results show that the weekday feature is the least. We can see that this feature even has a negative impact on the performance of the Random forest model. Moreover, in analyzing the previous timesteps, we expected the previous timesteps closer to 1 to have the most importance, while the importance goes up with the power of n by almost all our models and should be investigated further. Ideally, we will look at past predicted values for more than 12 hours and perform a similar permutation feature importance on it till it drops. We could also isolate the past predicted values and see if the performance drops and if it does not, we could remove the other features. Previous passenger volume at a bus stop is very important and should be present to fully describe passenger behavior. The suggestion to the transportation authorities would be to keep a record of the passenger data.

In table 17 we can see a positive influence on training the Decision Tree model with the previous time steps in comparison with the model without the previous time steps. The model trained with the previous time steps has performed better, around 6% based on the macro average f1-score. This model was also able to generalize the data better since it has a better performance for classes 1 and 2. We can see a huge improvement when we include other features in our training model. However, it must be mentioned that it makes the training time 10 times longer.

A reason why the feature importance table is important can be seen the table14. We can see that the Random forest model has the worst result on the on-demand dataset. This can be due to the fact that we do not have a couple of very important features such as *rout_count*, which we know, based on permutation feature importance for each model, are very important features for the models and help the models to improve their performance. The same performance is shown in table 14 for training the other models on the on-demand dataset and we can see that the same as the Random forest model, the prediction was class 0 most of the time. Therefore, the reason behind these bad performances is not because of the model algorithms but because of the dataset.

Future Work In our study we faced some issues that could be investigated to improve our results. Following we can see some problems that we are interested to work on them in our future work.

Improving statistical results: All statistical and machine learning algorithms applied in this research did not show the appropriate prediction for classes 1 and 2, as we can see in table 12. Our results indicate a higher level of f1-score for class labels 0 and 3. This can be attributed to the varying traffic levels at different bus stops. It may be a case of shortcut learning where certain bus routes are always busy, for example, Passau Hbf and some which are empty on a regular basis, and anything in between is hard to learn for the models. Changing class weights, undersampling or artificially creating samples for class 0 (eg. Synthetic Minority Oversampling Technique (SMOTE)) can be further used to deal with unbalanced classes. Some probable suggestions could be combining the classes together and creating two classes: less than 3 and 3+ or different classes, like less than 3 between 3 and 10 and more than 10 also "The improved Random forest model" suggested in [5], which is based on the feature importance measurement could be used to improve our Random forest model.

Aberration study on SARIMA model: SARMA multivariate, SARIMA with Walk forward, and SARIMA univariate models were trained on the bus stop Passau Hbf and the running time of SARIMA with Walk-forward model for only bus stop Passua Hbf took 95 minutes for 500 values and for 1 week, we have $7 \times 24 \times 50 = 8400$ values so we need to reduce the running time. A way to reduce the running time could probably be grouping bus stops with similar time series.
Also, the result of this model with different methods is ineffective for selecting the best model for our classification problem as they were trained on the bus stop Passau Hbf. A way to overcome this problem could be training them on all bus stops. In this case, we probably have to use different machines in parallel to reduce the running time. Another way could be training other models, suggested in this research, for the bus stop Passau and for the bus stop Pölzöd as well as SARIMA for this bus stop.

Maximum steps for previous timesteps feature: We also are interested in doing a study to find the reason for selecting the 12 previous timesteps feature as the most important feature and investigate a possible maximum number for the previous timesteps.

Distance to ZOB: Excluding the past covariates, distance to ZOB was our most important feature and it may be worthwhile to investigate the distance to a particular landmark a bit further and extract more features like that. In the absence of reliable passenger counts, it is a good feature.

Confidence interval: We would add a confidence interval with our results to show the standard deviation. A small standard deviation would mean that the results are robust and reliable.

On-demand dataset: Features like *route_count* and others are important and should also be collected for the on-demand dataset. We observed a drop in f1-score compared to the regular dataset which has these features.

WOHIN DU WILLST: Use this service to do a proof-of-concept using the null hypothesis to decide whether or not we want to include them in our feature selection

Bibliography

[1] Deutsche bahn. https://regional.bahn.de/regionen/bayern. Accessed: 2023-01-04.

[2] Meteostat weather data. https://meteostat.net/en/place/de/passau?s=10893&t=2019-01-01/2019-12-31. Accessed: 2023-01-04.

[3] Rapid api for meteostat. https://rapidapi.com/meteostat/api/meteostat/. Accessed: 2023-01-04.

[4] Weather condition codes. https://dev.meteostat.net/formats.html#weather-condition-codes. Accessed: 2023-01-04.

[5] P. Das A. Gangopadhyay A. R. Chintha A. Paul, D. P. Mukherjee and S. Kundu. Improved random forest for classification. *Transactions on Image Processing*, 27(8), 2018.

[6] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. pages 2623–2631, 07 2019. ISBN 978-1-4503-6201-6. https://doi.org/10.1145/3292500.3330701.

[7] George.E.P. Box and Gwilym M. Jenkins. *Time Series Analysis: Forecasting and Control*. Holden-Day, 1976.

[8] Anila Cyril, Raviraj H Mulangi, and Varghese George. Bus passenger demand modelling using time-series techniquesbig data analytics. *The Open Transportation Journal*, 13(1), 2019.

[9] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[10] T. Hastie, R. Tibshirani, and J.H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer series in statistics. Springer, 2009. ISBN 9780387848846. URL https://books.google.de/books?id=eBSgoAEACAAJ.

[11] K.W. Hipel and A.I. McLeod. *Time Series Modelling of Water Resources and Environmental Systems*. ISSN. Elsevier Science, 1994. ISBN 9780080870366. URL https://books.google.de/books?id=t1zG8OUbgdgC.

[12] Charles L Marohn Jr. *Strong towns: A bottom-up revolution to rebuild American prosperity*. John Wiley & Sons, 2019.

[13] Timo Stadler, Amitrajit Sarkar, and Jan Dünnweber. Bus demand forecasting for rural areas using xgboost and random forest algorithm. In *International Conference on Computer Information Systems and Industrial Management*, pages 442–453. Springer, 2021.

[14] Fatos Xhafa. Lecture notes on data engineering and communications technologies, 2019.