Utrecht University

Department of Information and Computing Science

Master Applied Data Science Thesis

Improving the timetable planning of the NS by classifying train turnover times

First examiner:

Arno Siebens

In cooperation with:

Nederlandse Spoorwegen (NS)

Candidate:

Madio Seck

Student Number:

5921902

3th of July, 2024

Abstract

The Nederlandse Spoorwegen (NS) is one of the key components of public transportation in the Netherlands. It is essential to maintain punctuality and minimize delays in the timetable planning of the train network. This thesis focuses on improving the NS train timetable by trying to improve the planned turnover times of trains at different stations to be more accurate. Machine learning classification models are used to learn complex patterns from large datasets that might otherwise not be fully captured by more traditional methods. This research uses creates a classifier model to support turnover time decision-making, by use of a random forest classifier and a support vector machine classifier model.

Contents

1. Introduction

1.1 Motivation and context

Rail transport in the Netherlands has a ridership of over 438 million per year, with passengers traveling over 17.1 billion kilometers per year ("Nederlandse Spoorwegen", n.d.). It is undoubtedly a cornerstone of the public transportation system in the Netherlands by providing efficient and reliable mobility for these passengers on a daily basis. In this context, it is critical to be aware of the robustness of the train schedules to maintain punctuality and minimize delays. Of all the different operators in this network, Nederlandse Spoorwegen (NS) is the biggest. Thus, this thesis focuses on improving the NS train timetable by trying to predict the turnover times of trains at different stations more precisely.

Turnover time is the time required for a train to switch directions at a station. Factors influencing the turnover times are diverse and include for example the type of the rolling stock, whether a driver change is performed during the turnover, the length of the train and included buffers. Being able to predict this time better enhances the robustness of the train schedule, leading to fewer delays. Additionally, the ability to predict turnover times more precisely can lead to unforeseen events being better handled, such as technical issues or unexpected delays. With a more robust schedule, the NS can respond more dynamically to such disruptions.

The need of NS timetable planners for further insights and tools for developing robust schedules is what motivated this research. Moreover, the methodologies developed in this research have the potential to be applied beyond NS and the Dutch rail networks. Other railway operators and different forms of public transportation could benefit from similar predictive models.

To that end this research uses advanced analytical methods that can uncover patterns that are not easily detectable through more traditional approaches.

1.2 Literature overview

This section gives the reader more context and knowledge on what turnovers exactly are, how they influence timetable planning and different methods that exist or can be experimented with to improve the turnover times to better reflect reality and improve the timetable planning of the train network.

1.2.1 Turnovers defined

As mentioned in an earlier section a turnover time is the period required for a train to switch directions at a station, other names for this occurrence are turnbacks, station reversals, short-turns, runaround or in Dutch it is sometimes called "kopmaken en omlopen". Each of these names results in the same final result, a train changes directions into the opposite direction it went.

In the Netherlands this usually happens by having the machinist deactivate the front train cabin (the locomotive), walking to the other train cabin on the other side of the train and activating it. Afterwards, the train can be pulled in the direction it originally came from. This happens at the final stations where the train track ends, but it can also occur in other stations due to that part of the track ending there (“Kopmaken, omlopen, draaien en driehoeken”, n.d.).

The time this takes can vary depending on a variety of factors such as the station, the type of train, the length of the train, whether cabins get attached or detached etc. Without any delays the time generally planned for turnovers varies between 2 minutes at the fastest, for the newest trains with a change driver "wisselmachinist", to almost 14 minutes, for a very long train that also is attaching or detaching cabins, as is visible in the appendix 6.2.

1.2.2 The timetable problem

The problem of timetabling for railways has had numerous studies done on and solutions proposed for it. The two general forms of timetables are cyclical and non-cyclical. The cyclic timetable has the advantage of being easily memorized by passengers, but since it is operated in the same way for peak and non-peak hours, it may result in higher operating costs according to Cadarso and Ángel, 2012.

The non-cyclical timetable scheduling decisions are based on estimates of the value of running different types of service at specified times according to Brännlund et al., 1998. This results in the timetabler having to decide whether a time or an event is populous enough to warrant public transportation. As one might imagine this involves the timetabler having to keep a very expansive view of the happenings of every location and the flow of people in the area they are planning for. This can be an expensive process, especially so for large cities with lots of events happening at any time and place. However, it can also result in a very minimal deviation of the timetable if the resources are present (Canca et al., 2014).

Due to the nature of trains being public transport and their goal being to be easily accessible and to transport lots of people most countries choose to go for a cyclical timetable to make it as accessible and comprehensible as possible and to minimize the adjustments that need to be done on a day-to-day basis.

1.2.2.1 The Dutch timetable

At the end of 2006, the Netherlands introduced a new train timetable, which was a complete change from its previous timetable which was constructed in 1970. This increased the number of passengers, and trains arriving and departing more on schedule than before and allowed for future growth of the transportation system (Kroon et al., 2009).

To create this new timetable, the timetable problem was modeled by creating a cyclical timetable for a single day and then extending it to a whole week by copy-pasting it. For the daily timetable, a variety of scheduling models and techniques were used. To define the scheduling problem a periodic event scheduling problem (PESP) was used to define the constraints. To solve the smaller instances of these timetable constraints mixed integer programming (MIP) was used which solves the problem as a mathematical formula and finds the optimal solution. To solve all the instances and make sure that all the constraints line up in a single schedule a special module CADANS was developed. CADANS is based on constraint programming techniques, which includes special-purpose constraint programming techniques to solve, what in essence is a giant mathematical formula, in acceptable computing time while still adhering to the PESP constraints. This feasible solution was then optimized for the final timetable (Kroon et al., 2009).

With some adjustments, this is still the timetable that the Netherlands is currently using as a basis for its weekly cyclical operation. These adjustments were made by taking into account data from conductors, passenger counts, smart card (OV-chip card) data, crowding reports, forecasts of passenger growth and, for example, holiday periods. Using this data, the number, size and types of trains in the timetable are adjusted to fit the current needs of the passengers ("Planning van treinen | Reisinformatis | NS", n.d.).

1.2.2.2 Timetable optimization techniques

Most national railway planners use similar methods to create the initial timetable as the NS. Most countries use a variation on MIP, such as mixed integer nonlinear programming (MINLP), to create the initial timetable before optimizing it to better suit the situation (Zhang et al., 2018).

Research on the timetable optimization problem can be differentiated on what the objective function to be optimized is. Some objectives of the research can be maximizing profit, minimizing deviation, minimizing travel time, and maximizing the number of people transported, but it generally comes down to looking at certain situations in regards to its infrastructure capacity and finding the "best" solution for that situation (Yuhua et al., 2022).

Research focused on minimizing the deviation from the timetable while mostly retaining the existing infrastructure and planning involves adding new techniques and strategies to either compensate for the delays which cause deviation by in real-time adapting the timetable to solve the issue or changing the situations to decrease the delays.

The real-time methods require more data and a system which can adapt and change the timetable to suit the current demand to prevent possible deviations from the timetable. The real-time data which could be used are the smart card data of passengers to identify travel patterns and foresee congestion by looking at time and location data and passenger profiles. This dynamic-timetable design shows significant potential and may be a method to improve the timetable (Sun et al., 2014).

Another method is by allowing for stop-skipping, allowing trains to skip some stations when the deviation time is relatively high (Chen et al., 2022), to decrease the overall deviation from the timetable. While this does decrease the deviation time it also increases the waiting time for passengers whose station is skipped thereby decreasing their satisfaction.

Larger structural changes to the railway network such as allowing for unfixed platform time and flexible rail track allocation can improve the capacity of turnback operations thereby reducing delays according to Jiang et al., 2015. These kinds of larger changes which involve the physical world generally can have good results, however there are plenty of alternative methods to reduce deviation time which are cheaper in both cost and time. One of these more cost-effective methods is using machine learning to optimize the timetable.

1.2.3 Machine learning for timetable optimization

In this situation, we're looking at minimizing the deviation between the expected turnover time and the actual turnover time. By being able to more accurately estimate turnover times the amount of delays should decrease. This is due to situations with structural delays taking more time and thereby decreasing the number of times there is a deviation from the timetable. Some restrictions however are that the current timetable can't be adjusted too much, i.e. no changing the rolling stock types, amount of cabinets etc.

Naturally, there can still be improvements to the current timetable as there are still some structural delays which might be fixed. By their own admission, the amount of data analysis methods that the NS employs to improve their turnover times is quite low and that is also where this thesis comes into play.

With the objective being to minimize the deviation, we are trying to predict the correct amount of time that turnovers need to take depending on the situation. So to solve this problem we can either create a model which given data on the situation returns an integer which is the minimal turnover time needed or it makes an estimation of whether the turnover time selected is correct for the situation. In short, it can be solved as a regression problem or a classification problem.

A classification model to support the decision-making in this workflow would be a excellent solution to the problem. This is due to a support classification model being more easily integrated into the existing workflow and so having a higher chance to directly add value to the NS, so it was chosen to implement this.

1.2.3.1 Classification model

A classification model is a type of predictive model used in machine learning and statistics to categorize data into different classes based on its features. The model learns patterns and relationships from training data which, in this case, has the class for each record attached. Meaning that it is a supervised learning method.

Generally, a classification model is relatively transparent, this is due to the classification model's ability to show the correctness and accuracy of a model by use of multiple metrics it has. By calculating the amount of true positive (TP), false positive (FP), false negative (FN) or true negative

(TN) can then be used to determine the accuracy, recall and precision of the model.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{1.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{1.2}$$

$$Precision = \frac{TP}{TP + FP} \tag{1.3}$$

Accuracy is the number of correct predictions made over all predictions. Accuracy is relevant for classification problems with balanced data. However, as data is not balanced for all activities two other metrics are considered. The second and third metrics used are therefore the recall and precision. The recall tells what proportion of true outcomes were predicted as true while the precision tells what proportion of the predictions classified as true were true. The recall and precision together provide a realistic indication of the performance of the model. These last two metrics can be combined to create the F1-score, which is the harmonic mean of recall and precision (van Hassel, 2019).

$$F1score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{1.4}$$

A confusion matrix is used alongside these evaluation metrics. The reason for that is that the models have to predict multi-class classification problems with unbalanced data. These metrics might give a high value if the prediction for the majority class is good even though the accuracy of the minority classes is terrible. By visualizing it in a confusion matrix you can more easily determine whether the model is accurate for all the different classes. This is to prevent a case of the selected solution turning out bias against the minority classes (Hossin & Sulaiman, 2015).

It was chosen not to visualize the decision trees due to the sizable scale of the tree models, single decision tree and random forest, the amount features and data size making it not feasible.

1.3 Research question

The primary objective of this research is to explore the potential of machine learning techniques to predict the realized turnover times of trains at NS. Thus, the research question that guides this study is:

How can machine learning methods help NS to predict train turnover times and thereby increase robustness?

In the context of railway operations, turnover time is a metric that affects the overall efficiency and punctuality of the trains. It refers to the time that is required for a train to switch directions at a station. Accurate predictions of this are required for developing robust train schedules and minimizing delays.

The turnover times can be by a multitude of factors, including the type of rolling stock, driver changes, train length and buffer times in the schedule. Traditional methods of predicting turnover times may not fully capture the complexity of these factors. Therefore, there is a need to employ more advanced predictive models that can handle such complexities.

Machine learning in general offers a promising approach to address this challenge. Machine learning models can learn complex patterns from large datasets that might otherwise be not fully captured by more traditional methods, which makes them well-suited for predicting outcomes that are influenced by multiple interdependent variables. By employing such a model this research aims to accurately predict these realized turnover times based on the provided historical data.

The research objectives are as follows:

1. **Data Collection and Preparation:** Gathering and preprocessing the historical data from NS.
2. **Feature Selection:** Identifying and selecting relevant features that influence turnover times, such as the rolling stock type, driver changes etc.
3. **Model Development:** Developing a machine learning model to predict if a turnover time is suitable for the situation.
4. **Model Evaluation:** Assessing the performance of the model using appropriate evaluation metrics to ensure its accuracy and reliability.
5. **Recommendations:** Providing insights and recommendations for NS timetable planners to improve schedule robustness based on the model's predictions.

2. Data

2.1 Data collection & description

The data used in this research is sourced from the NS databases, and exported as CSV files from the server. The dataset contains detailed historical records of train activities, including arrivals and departures at different stations. The dataset spans a period of one year, from January 1st 2023 to December 31st 2023, providing a comprehensive view of train activities.

Ensuring the reliability of the data is fundamental for the predictive model, as it needs to represent real-world turnover times accurately. The data from NS is considered highly reliable as it is systematically recorded and maintained by the organization on its servers. NS follows strict data collection and validation protocols to ensure that the recorded information accurately reflects realized operations. Despite these measures, preprocessing and validation are necessary to address any inconsistencies that may still arise, ensuring data integrity for accurate modeling.

As the dataset contains only operational and logistical information, no ethical considerations regarding personal information had to be made. As NS wished their data to be kept from the public, the dataset was stored locally and safely.

The dataset consists of 28,233,618 rows and 18 columns. Each row represents a train activity, such as an arrival or a departure from a station. The content and description of each column are presented in Table 2.1. As can be seen in the table, turnover times are not included in the dataset. This needs to be calculated based on the available data.

Column	Description	Type of variable	Unit
TRAFFIC DATE	Date of the activity	Nominal	-
TRAINNUMBER	Number of the train	Nominal	-
TRAINSERIE	Series of the train	Nominal	-
TRAINSERIE_DIRECTION	Direction of the train	Nominal	-
STATION	The station where the activity takes place	Nominal	-
ACTIVITYTYPE	The type of activity	Nominal	-
DISTANCE_M	The distance traveled since the last activity	Ratio	Meters
PLAN_DATETIME	The planned date and time of the activity	Ratio	YYYY-MM-DD HH:mm:ss.sss
REALIZED_DATETIME	The realized date and time of the activity	Ratio	YYYY-MM-DD HH:mm:ss.sss
DELAY	The time between planned and realized time	Ratio	Seconds
TURNOVER_INDICATOR	Indicates whether a turnover took place	Binary	-
PREVIOUS TRAINNUMBER	The number of the previous train, if a turnover took place with a change of train number	Nominal	-
COMBINE	Indicates wether the train combined with another train into a train with more carriages	Binary	-
SPLIT	Indicates wether the train split into multiple trains with fewer carriages	Binary	-
ROLLINGSTOCK TYPE	The type of rolling stock	Nominal	-
NUMBER CARRIAGES	The number of carriages	Ratio	Carriages
DRIVER_CHANGE	Indicates whether a change of driver took place	Binary	-
DEPARTURE_SIGNAL_SHOWS SAFE	The date and time when the train was signalled it could safely depart	Ratio	YYYY-MM-DD HH:mm:ss.sss

Table 2.1: Description of variables

2.2 Preprocessing

Data preprocessing is a critical step in ensuring the data is ready for analysis. The preprocessing pipeline ensures that the data is clean, consistent, and ready for modeling. In this step, the variable of interest, turnover time, is calculated as well.

2.2.1 Data type setting

The first step involves setting the correct data types for each column in the dataset. This ensures that the following operations and calculations are performed correctly and efficiently. Columns with dates and times are set to *datetime*-types, remaining numerical variables are set to integers, and categorical variables are set to strings. Next, the data is sorted by trainnumber, date, and time.

2.2.2 Turnover time calculation

Calculating turnover times is a complex process as there are two situations in which a turnover happens, with each a unique way of being recorded in the dataset:

1. **End of route turnover:** This happens when a turnover is performed at the end of a route and the train changes trainnumber as it starts its new route. Here the *TURNOVER_INDICATOR* is set to 1, and the trainnumber it had before performing the turnover is registered in the PREVIOUS_TRAINNUMBER column. In this case, the turnover time is the interval between the last arrival of the previous train number and the first departure of the new train number.
2. **Mid-route turnover:** This happens when a turnover is performed in the middle of a route. Here, the *TURNOVER_INDICATOR* is also set to 1, but there is no *PREVIOUS_TRAINNUMBER* since the route doesn't change. The turnover time is calculated based on the last arrival at the same station of the same train.

In both cases, the planned turnover time and the realized turnover time are calculated based on the *PLAN_DATETIME* and *REALIZED_DATETIME* columns.

2.2.3 Data cleaning

The next step is cleaning the data, this consists of the removal of data that is not relevant and outliers. Only records of turnovers are kept, all other instances are filtered out. This still leaves some records that are not relevant however, such as a few instances of German trains that are not representative for this analysis and activity types that are not of interest.

As depicted in figure 2.1, turnover times roughly follow an inverse-gamma distribution. In the initial phase of the project, it was determined that turnovers longer than 2500 seconds (roughly 40 minutes) were neither feasible nor of interest. Therefore, the long tail of the depicted distribution is essentially cut off, resulting in a data loss of around 2%.

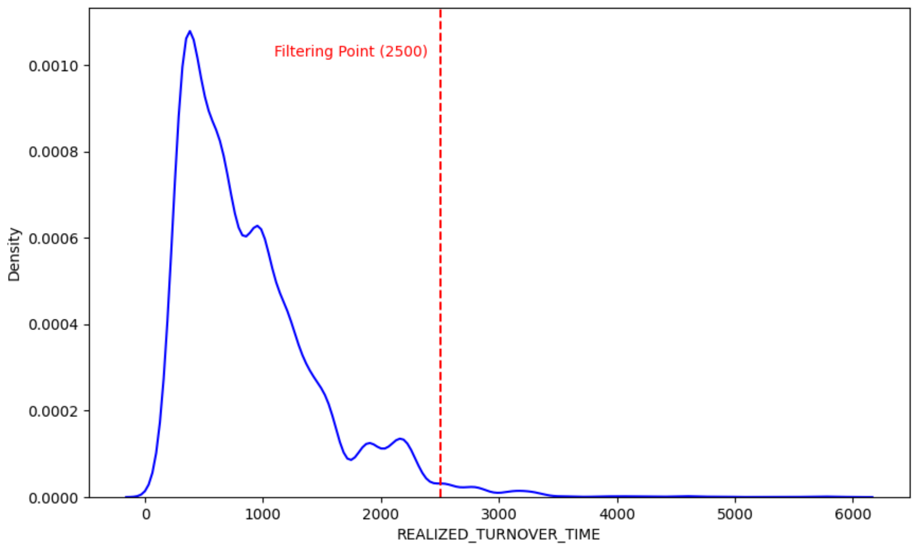


Figure 2.1: Distribution of realized turnover time in seconds

Further filtering was applied to remove turnovers which suffered a severe delay. As the planners at NS are unable to predict delays beforehand, the proposed machine learning models cannot utilize this data either. To account for this fact, trains which departed more than three minutes late or early were removed. This filtering results in an additional reduction of the original data by 10%.

Figure 2.2 illustrates the significant impact of the aforementioned filtering steps on the dataset. This filtering results in shorter turnover times with minimal delays and significantly enhances the correlation between planned and realized turnover times.

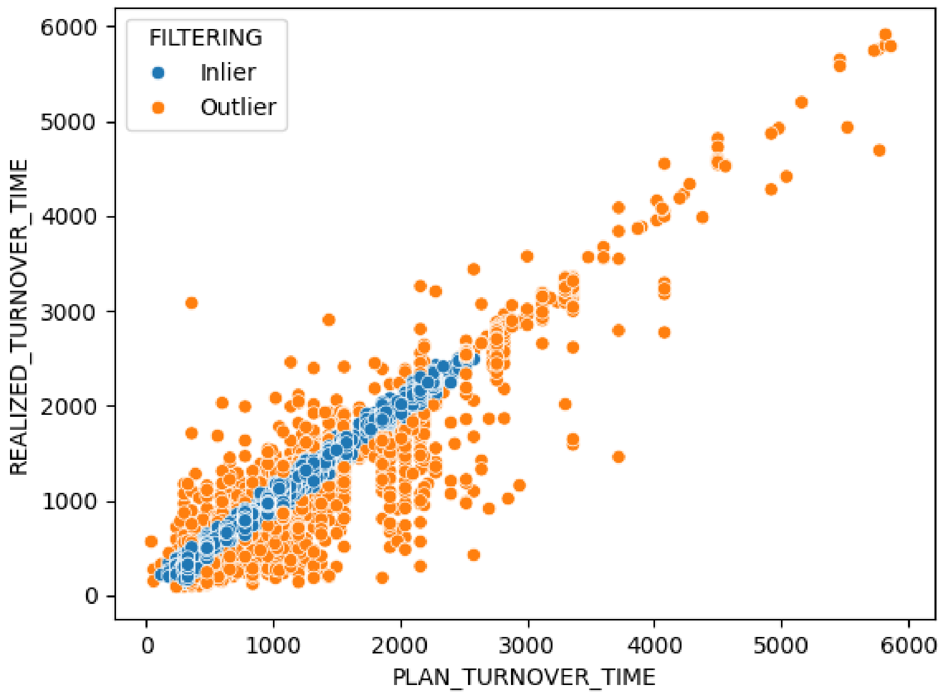


Figure 2.2: Scatterplot showing the relationship between planned and realized turnover times, with outliers and inliers separated

2.2.4 Feature engineering

Lastly, data transformation in the form of feature engineering was performed. The cumulative distance traveled is calculated based on distances between previous stations, the day of the week and hour of the day are extracted from the planned datetime and made cyclical and the difference in realized and planned turnover time is categorized.

2.2.5 Encoding of nominal variables

The nominal variables of the dataset need to be encoded for use by the models. This is because the classification models can't directly process non-numerical data such as strings. To this end the *LabellEncoder* function of the *Sklearn* library is used on all nominal features.

2.2.6 Dataset after preprocessing

As only records of turnovers are kept, the amount of rows in the dataset is greatly reduced from the original 28,233,618 to 247,970. While the amount of columns has increased from 18 to 21 due to the feature engineering. As some of these columns may not be relevant for the model, the amount of columns is further narrowed down during feature selection, discussed in section 2.3.2. Table 2.2 shows the columns of the preprocessed dataset and their data types.

Column	Description	Data Type
TRAFFIC_ DATE	Date of the activity	datetime
TRAINNUMBER	Number of the train	int
TRAINSERIE	Series of the train	int
TRAINSERIE_DIRECTION	Direction of the train	string
STATION	The station where the activity takes place	string
ACTIVITYTYPE	The type of activity	string
DISTANCE_M	The distance traveled since the last activity	int
N DATETIME	The planned date and time of the activity	datetime
REALIZED_DATETIME	The realized date and time of the activity	datetime
DELAY	The time between planned and realized time	int
TURNOVER INDICATOR	Indicates wether a turnover took place	int
PREVIOUS TRAINNUMBER	The number of the previous train, if a turnover took place with a change of trainnumber	int
COMBINE	Indicates wether the train combined with another train into a train with more carriages	int
SPLIT	Indicates wether the train split into multiple trains with fewer carriages	int
ROLLINGSTOCK TYPE	The type of rolling stock	string
NUMBER CARRIAGES	The number of carriages	int
DRIVER CHANGE	Indicates wether a change of driver took place	int
DEPARTURE_SIGNA SHOWS SAFE	The date and time when the train was signalled it could safely depart	datetime
UM_DISTANCE_M	The cumulative distance traveled before performing the turnover	int
DAY OF WEEK	The day of the week (Monday=0,Sunday=6 =	int
HOUR	The hour of the datetime	int
DIFF_TURNOVER_TIME_CAT	Classification of the difference in realised and planned turnover time	string

Table 2.2: Preprocessed dataset features with their name, description and data types

2.3 Exploratory data analysis

Data exploration has been an integral part of the aforementioned steps. Visualization guided the creation of the prepossessing pipeline and subsequently helped identify key features. This section begins by detailing the target variable of this study and then examines how some input variables relate to it. Due to space constraints, only the most notable findings will be highlighted here. All the exploratory data analysis figures are visible in appendix 6.3.

2.3.1 Target variable

The value to be predicted in this study is DIFF_TURNOVER_TIME_CAT. This is the classified difference in turnover time planned and realized. It stands for indicating whether a chosen turnover time is too long, too short or just right. A turnover time is deemed *perfect* when the absolute difference between the planned and realized time is 1 minute or less. If the positive difference (planned time is more than realized time) is more than 1 minute then it's *early*. If the negative difference (planned time is less than realized time) is more than 1 minute then it's *late*.

The distribution, as visible in figure 2.3, shows that there is an imbalance in the frequency of occurrence of the different categories. You can see this more clearly in figure 6.2.

This is especially the case in the earlier and later segments of the data. This imbalance mostly has to do with the timetable of the NS being quite good, so situations, where trains leave too early or too late, are in the minority. Due to the size of the dataset, there are still enough records of each class to be useful, however depending on the model the class imbalance could still result in the models having more trouble classifying the minority classes.

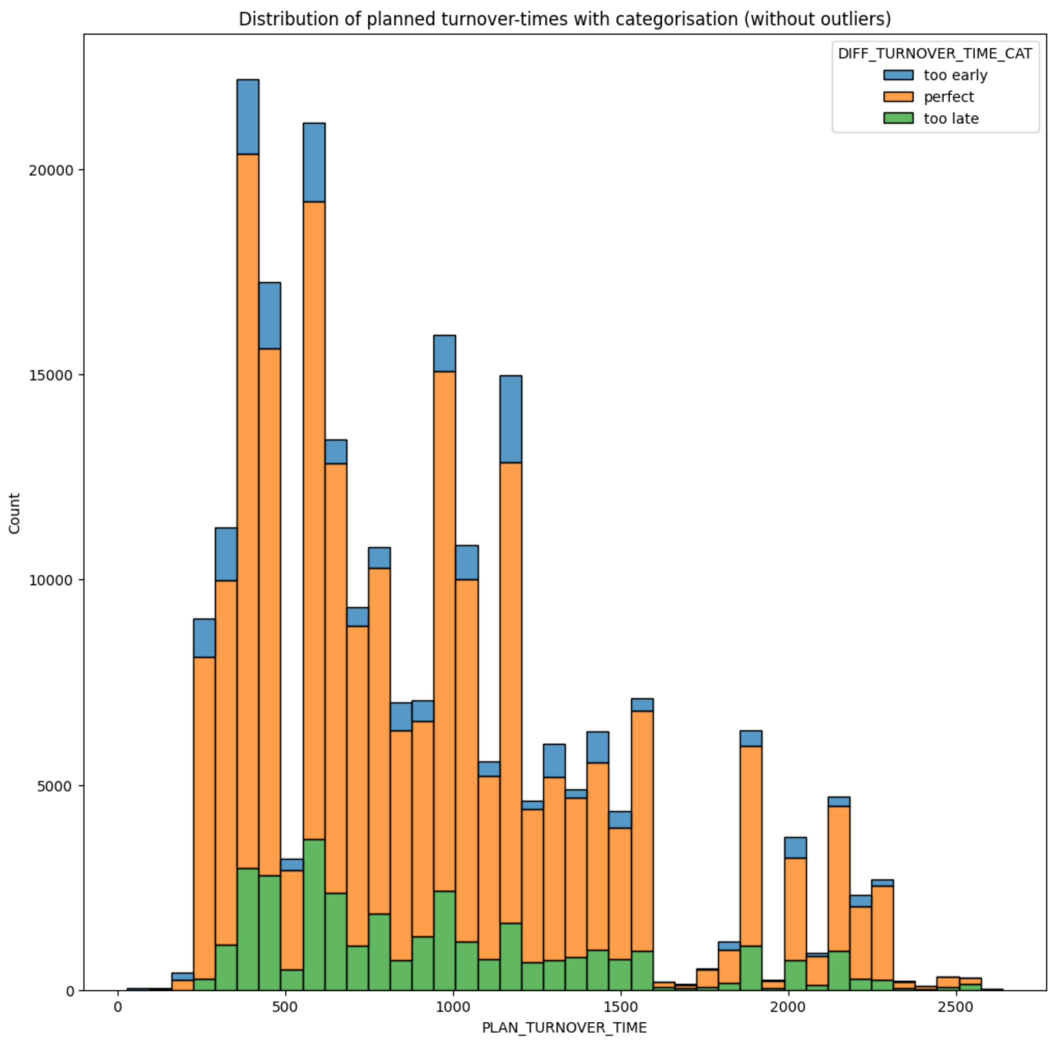


Figure 2.3: Scatterplot showing the relationship between planned and realized turnover times, with outliers and inliers separated

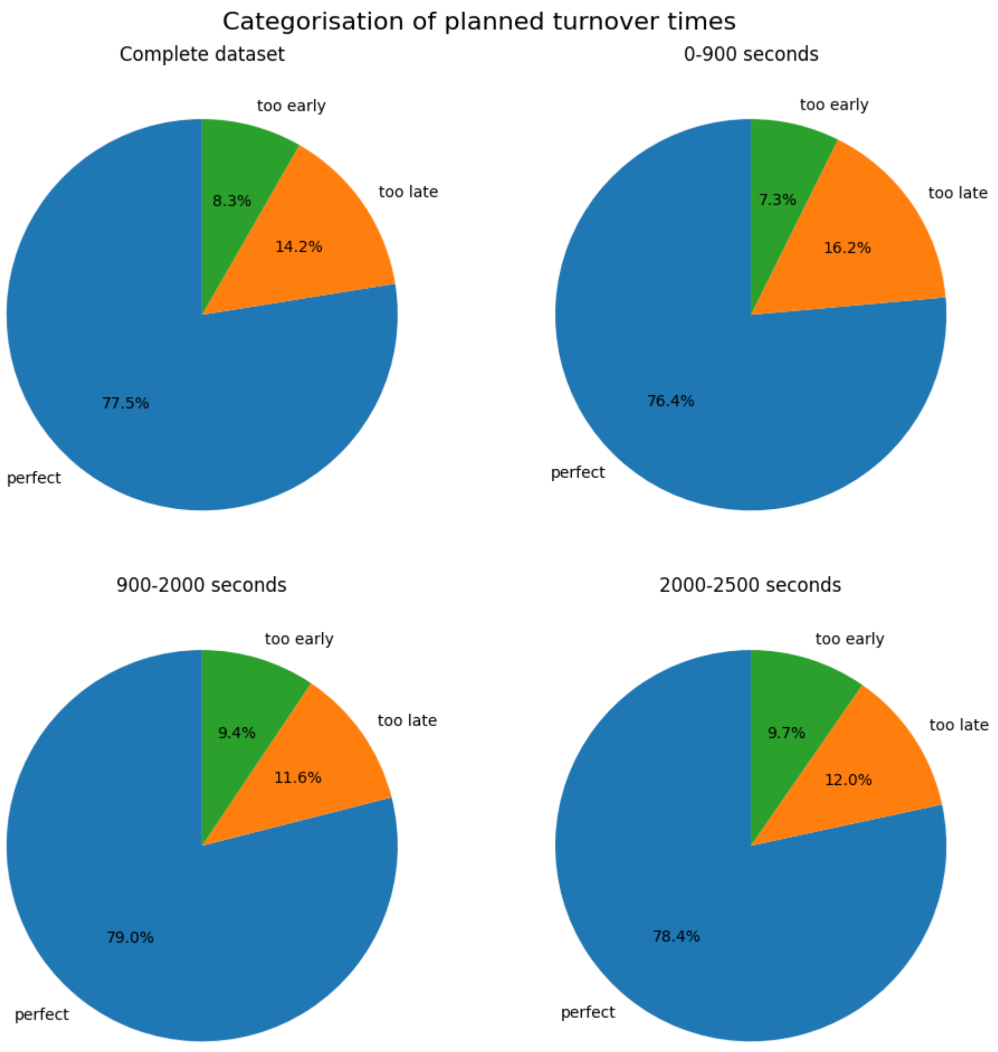


Figure 2.4: Multiple pie charts showing the distribution of the categorized turnover-times for the complete and segmented datasets

2.3.2 Input Variables

The input variables visible in table 2.1 are known by the NS to affect turnover time in various degrees. As an example, figure 2.5 demonstrates how turnover times vary across different types of rolling stock. Factors like rolling stock types, driver changes and the number of carriages are particularly relevant for short turnovers but become less significant as turnover times increase.

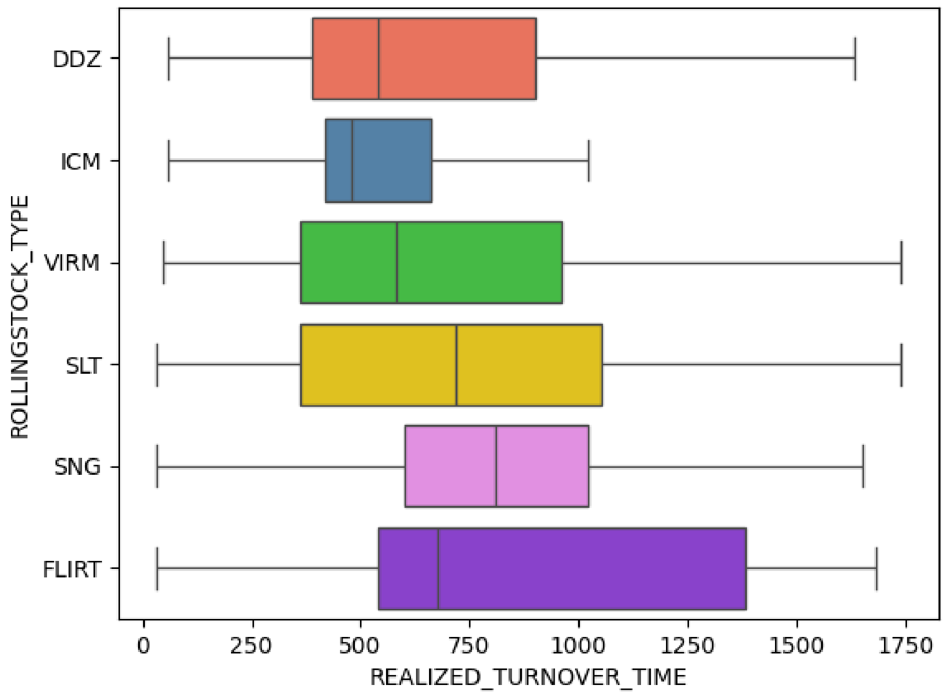
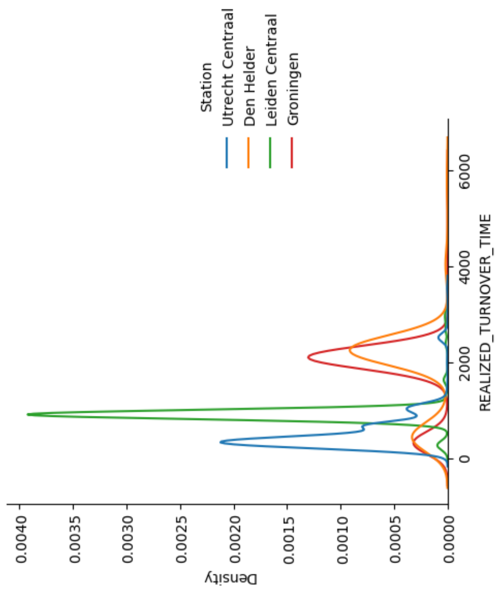


Figure 2.5: Realized turnover times for different types of rolling stock

Combining or splitting the train cabins has a significant effect as they increase turnover time by an average of three minutes. Still, such operations occurred in less than 5% of the data.

Perhaps one of the most important input variables is the station at which a turnover occurs. Only a third of Dutch stations facilitate turnovers, with the ten largest accounting for more than half of them.

While most turnover stations follow the long-tailed distribution of turnover time, seen earlier in figure 2.1, some significantly deviate from this pattern with quite irregular distributions. These discrepancies are clearly visible in figure 6.3. According to NS, such differences are caused by the layout (e.g. track configuration, nearby dock size) and location of the station. Although valuable data may be embedded within these discrepancies, such station-specific data is also difficult to generalize which is why it is not included within this short project.



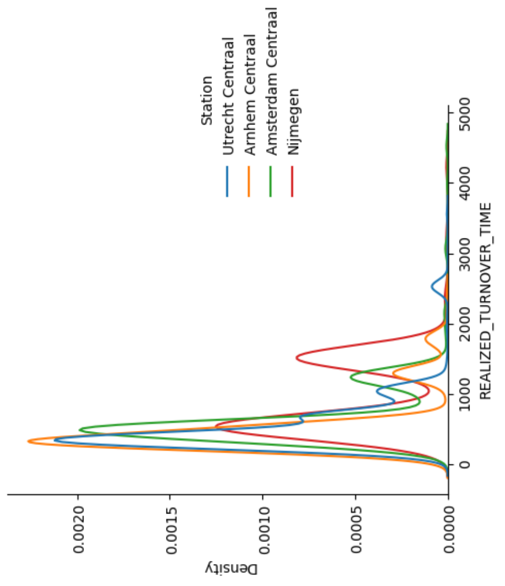


Figure 2.6: Distribution of turnover times at various stations categorized into common distribution patterns (left) and irregular distributions (right), with Utrecht Centraal for reference.

A less obvious variable affecting turnover time is the hour of the day, shown in figure 2.7. Few turnovers occur at night, while most occur during the evening rush hour. Interestingly, turnovers are shorter on average during these busy hours, suggesting that turnovers outside of rush hours take longer than the minimal required time. The varying frequency of driver changes and combining or splitting operations throughout the day may explain some of these changes in turnover times at each hour.

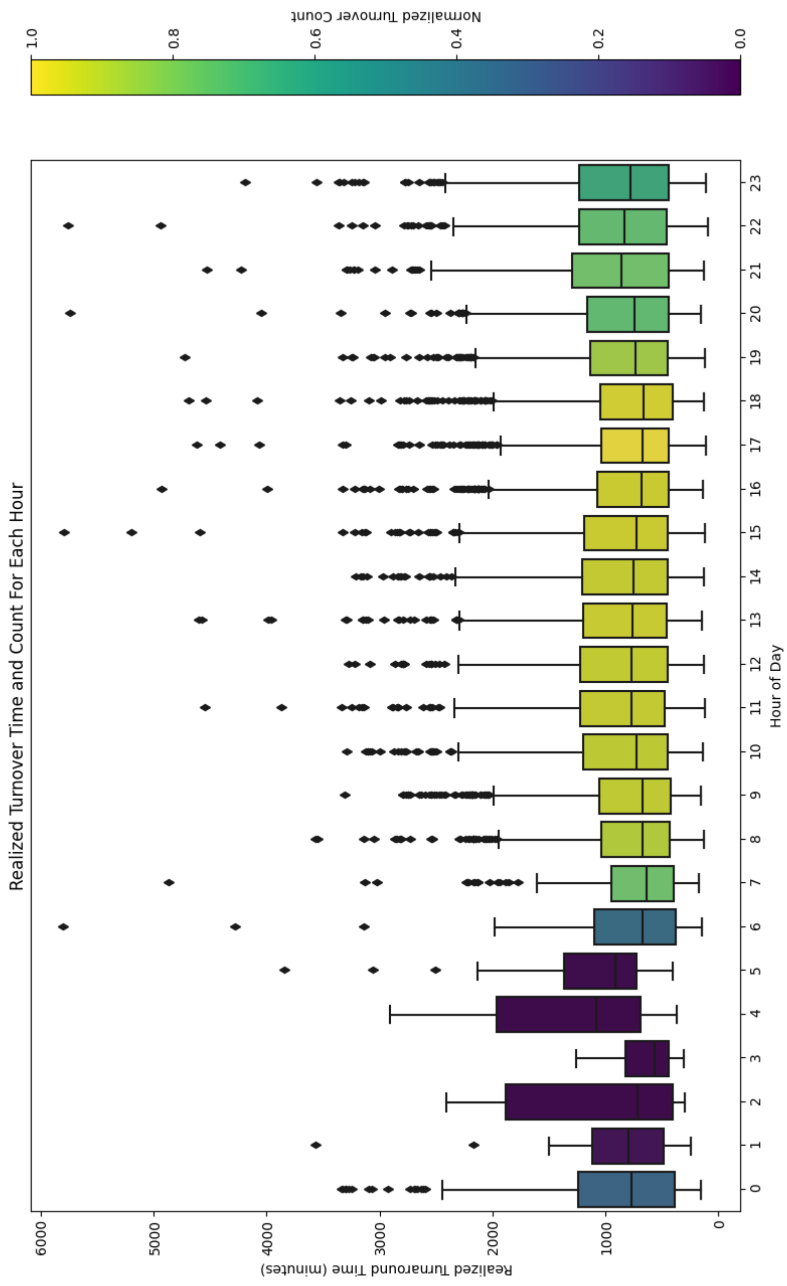


Figure 2.7: Realized turnover time and normalized turnover count by hour of the day

The importance of these variables was initially estimated by looking at the results from the exploratory data analysis and was later confirmed by looking at the feature importance used in the final random forest model. As you can see in table 6.8, the before mentioned features did have quite a bit of influence on the model. The most notable of which the time of day and week, represented as *HOUR_cos*, *HOUR_sin*, *DAY_OF_WEEK_cos*, *DAY_-OF_WEEK_sin*, the number of carriages, train serie and estimated turnover time.

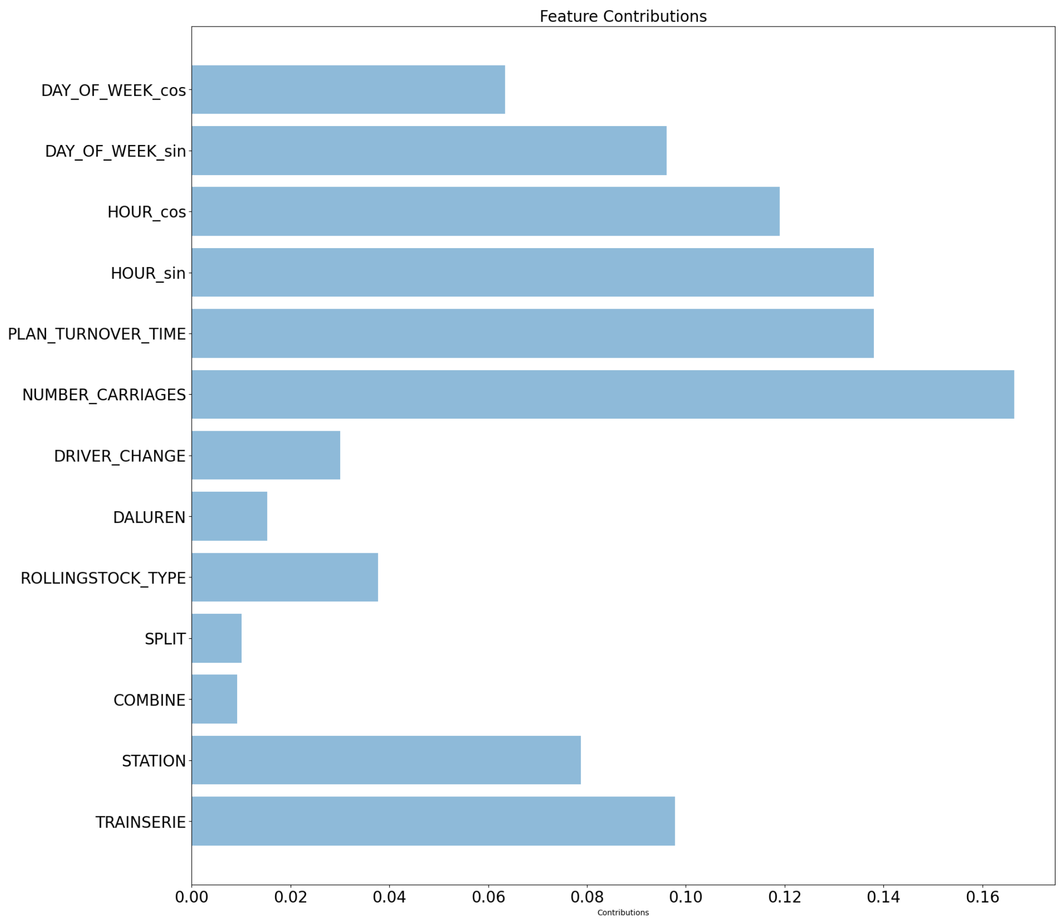


Figure 2.8: Feature importance, final-selection features

3. Method

3.1 Machine learning method used

Research around turnover times generally is not focused on using classification models, but rather on regression models so there aren't any pre-existing models and researches to build upon. However there is plenty of research on different machine learning models and their performance, so that was used as the basis for deciding which models to use.

The machine learning methods used were support vector machine classifier (SVC), random forest classifier (RF) and as a baseline model single decision tree classifier. These different models have different approaches to classifying the data and are all proven and effective methods.

All the models were trained, tested and evaluated on a train-test dataset. The selected dataset of a model was split using the *train_test_split* function of *Scikit-learn* with a 80-20 train-test on the same random seed. Meaning that the model was trained on 80% of the dataset and tested and evaluated on the remaining 20%.

3.1.1 Single decision tree classifier

A decision tree is a supervised learning algorithm based upon a flowchart-like structure in which each internal node represents a "test" on a feature, each branch represents the outcome of the test, and each leaf node represents a class label (contributors, n.d.).

This model was chosen as it is already widely used as a baseline for classifier models, especially so for decision tree models. It is the most simple version possible of a classifier model that is not a dummy model that always picks the same result.

3.1.2 Random forest classifier (RF)

An RF model is an ensemble of many decision trees using bootstrapping and random feature selection. It works by creating multiple separate decision trees, bootstrapping the data for them and later combing them into a single tree. This ensemble of decision trees, the RF, generally perform better than single decision trees due to the correlation of the tree being significant. The risk of getting overfitting is reduced and the error converges into some generalized value.

RF attains high classification accuracy and can handle outliers and noise in the data. RF is used in this work because it is less susceptible to overfitting and has shown good classification results in other research such as Iftikhar et al., 2018.

3.1.3 Support vector machine classifier (SVC)

An SVC works by representing data points into a plane and then separating those data points by use of an optimal hyperplane (the decision boundary). This decision boundary is constructed by looking at the nearest distance between the decision boundary and points to determine where the classification border should be. In essence, an SVC model uses a convex optimization problem to allow you to find the global optimum which separates the data points into classes (Aroef et al., 2020).

In addition, SVC can not only solve linear classification problems, but can also be applied to nonlinear classification cases by choosing from among different kernel functions (Wang et al., 2017). By choosing different kernels the function to estimate the decision boundary changes and depending on the dataset, different kernels lead to a better fit. The kernels work by computing the similarity between two points and to the linearized decision boundary, without having to compute in high dimensional space. The different kernel functions (i.e., linear, polynomial, radial basis, and sigmoid) map the data differently (Iftikhar et al., 2018).

The final model was run on the linear kernel of the SVC, as tests on the complete dataset showed that this kernel was the most suitable for the dataset, as visible in the figures in section 6.8.1.

The advantage of SVC is that minimal parameter adjustment is required, it is robust to outliers, and it handles class imbalance better than most models (Zhang et al., 2015). The disadvantages are that it is expensive to run due to having to include the requirements of a Gaussian function for each instance of the training set. This results in the training time increasing and performance to decrease (Iftikhar et al., 2018).

3.2 Feature selection

The NS initially came up with a dream-tool goal, which included a number of features namely; STATION, ROLLINGSTOCK_TYPE, DRIVER_CHANGE, PLANNED_ARRIVAL_TIME, NUMBER_CARRIAGES. The goal was to at minimum use this list of features and possible expand upon it.

Several additions were made to this list of features to increase the performance of the model. These additions were made based on several feature importance analysis which show the feature importance for several combinations of features, visible in figures 6.8, 6.7 and 6.6.

With the goal of the classification model being to support the current workflow, so to advise on whether a estimated turnover time is correct, it was chosen to add the planned turnover time (PLAN_TURNOVER_TIME). Furthermore to improve performance it was chosen to add features that could be engineered from the original set of features. So derived features, such as *HOUR_SIN* were added to the model as well.

3.3 Improving the models

Once the classification methods and features to be used were selected initial models could be trained and evaluated. The performance of these models could be further improved by segmenting the data into multiple segments and by tuning the hyper-parameters to better fit the problem.

3.3.1 Segmenting the data

By segmenting the data the amount of models that need to perform increases to the amount of data segments, but it also allows the different models to be more specialised and reliable on that part of the data. By segmenting the data the variance is reduced due to outliers on longer turnover times not impacting the estimation of shorter turnovers. Furthermore in some segments the size of the minority class increased which helps against the class imbalance.

This results in the different models having increased performance on their segment of data compared to on the entire dataset. Depending on the size of the data and how well the different categories are distributed this can also result in overfitting due to the data being skewed. So for that reason models are trained on the complete dataset and on the data segments and afterwards compared using the evaluation metrics to determine which is most suitable for this problem on a model without causing overfitting.

The data was segmented by looking at the distribution of the preprocessed data classified, visible in figure 6.1. From this we can tell that there are 3 peaks in which we will segment our data. The segments of turnover times are from 0 to 900 seconds, 900 to 2000 seconds and 2000 to 2500 seconds. These segmented dataset slightly differ in distributions of the classes, as is visible in figure 6.2.

3.3.2 Hyper-parameter tuning

Hyperparameter tuning is the practice of changing the settings with which a model is trained with the aim of improving performance. There are some cases where models with the hyperparameters at their default value outperform the tuned model (Weerts et al., 2020). To that end, the different models were trained on both their default hyperparameters and tuned hyperparameters to find the best performing one.

The models we used for classification, RF and SVC, each have some hyperparameters with high tuning risk, the performance loss that is incurred when a hyperparameter is not tuned, but set to a default value. Which suggests that it is important to tune these models. According to the research of Weerts et al., 2020, for a SVC model the most important hyperparameter is *C*, the regularization parameter which tells the model how much you want to avoid misclassifying the data. And for a RF model it is *max_features*, which is the maximum number of features to consider when looking for the best split ("RandomForestClassifier", n.d.). So these features will be more focused upon in the tuning of hyperparameters.

The hyperparameter tuning was done by running a gridsearchCV on both models with a variety of values for the parameters. GridsearchCV is an exhaustive search over the the given parameter grid to determine which combination of parameters gives the best result and afterwards, it returns those parameters ("GridsearchCV documentation", n.d.). Because gridsearchCv essentially is just trying out every combination of hyperparameters and training and evaluating the model, it can become quite expensive and time-consuming depending on the model and the size of the dataset.

To that end the models were run on a selected grid of parameters, visible in table 3.1. Random forest was a relatively cheap model to run so the gridsearch was performed on the complete or segmented dataset. However SVC was a much more expensive model to run especially so on the entire dataset. So to improve the performance of the gridsearch, the hyperparameter tuning of the SVC model was done on a dataset containing only a week's worth of data. The hyperparameters from the best-performing model were used for the final model on the complete dataset.

Model	Parameter	Meaning	Range	Best params
Random forest				
	max_depth	The maximum depth of the tree.	[50, 80, 100]	80
	max_features	The number of features to consider when looking for the best split.	[2, 3, 4]	4
	min_samples_leaf	The minimum number of samples required to be at a leaf node.	[3, 4, 5]	3
	min_samples_split	The minimum number of samples required to split an internal node.	[8, 10, 12]	10
	n_estimators	The number of trees in the forest.	[100, 300, 500, 750, 1000]	100
SVC				
	C	Regularization parameter	[0.1, 1, 10, 100, 1000]	1000
	gamma	Kernel coefficient for ‘rbf’, ‘poly’ and ‘sigmoid’ kernels	[1, 0.1, 0.01, 0.001, 0.0001]	1
	kernel	Specifies the kernel type to be used in the algorithm.	[’rbg’, ’linear’, ’poly’, ’sigmoid’]	linear

Table 3.1: Table showing which hyperparameters of the model were tested in the gridsearch

4. Results

4.1 Overview of the results

Overall the models were able to classify the *perfect* turnover times to a high degree, however the single decision tree and the best RF model had difficulty with the minority classes of *early* and *late*. As due to a possible lack of data, class imbalance occurred where the models performed well on the majority class but less well on the minority class leading to somewhat biased models. The best SVC model however had no trouble classifying all the different classes.

By segmenting the data of the single decision tree and the RF model they gave better results in comparison to using the whole dataset. Furthermore by tuning the RF model on the complete dataset and applying those hyperparameters to the segmented dataset gave the best results for the RF model.

While for the SVC model using the complete dataset and tuning the hyperparameters on the complete dataset gave the best results.

4.2 Single decision tree classifier

The single decision tree model was run on both the complete dataset as well as the segmented dataset. The model performed better with the segmented dataset and this was especially the case for the minority classes of *early* and *late*. As is visible in the metric result overview table 6.1 and the confusion matrix figures 6.9, 6.10, 6.11 and 6.12.

The difference between the metrics of accuracy, precision, recall and f1-score doesn't seem that significant between the complete dataset and the segmented dataset models, but that is due to the imbalance of data. When looking at the confusion matrices there is a clear difference in the accuracy of the false negative predictions there. This coincides with the difference in size of the minority classes in the dataset for the different segments.

For example, for the single decision tree model, the value of *late* in the confusion matrix for the complete dataset is 0.27 and the class is 14.2% of the used dataset. For the third segmented dataset with turnover time data between 0 and 900s, the same value is 0.45 and the class is 16.2% of the used dataset. Similarly for the other segmented data results, when the size of the class increases compared to the complete dataset, the ability to classify that class also increases. This is despite the absolute amount of data being less for the segmented data models, but its relatively being higher increases its chance.

The best model, which is the segmented dataset, has as true scores in the confusion matrix for the *perfect* class being around 0.80 for each data segment, the *early* class being 0.21, 0.33 and 0.28 respectively for the different segments and *late* class being 0.45, 0.35 and 0.28 respectively for the different segments as is visible in the confusion matrices in figures 6.10, 6.11 and 6.12. When compared to the complete dataset which has a true score for the *perfect* class being 0.79, the *early* class being 0.26 and *late* class being 0.27, the segmented datasets give equal to better results as is visible in table 6.4.

4.3 Random forest classifier (RF)

The random forest classifier model was run on both the complete dataset as well as the segmented dataset and both these models were also tuned using gridsearchCV to determine the optimal hyperparameters. The best model was using the segmented dataset, but the hyperparameters were tuned on the complete dataset.

Tuning the segmented dataset on its own segment caused overfitting and class imbalance and not tuning the model resulted in results comparable to the baseline model for the minority classes. The majority class of *perfect* stayed around the same score as the baseline model no matter the dataset of tuning. However, as is visible in table 6.8, tuning on the complete dataset and segmenting the dataset gave the best model.

This best model resulted in a true score for the *perfect* class being around 0.80 for each data segment, the *early* class being 0.48, 0.61 and 0.8 respectively for the different segments and *late* class being 0.67, 0.63 and 0.55 respectively for the different segments as is visible in the confusion matrices in figures 6.17, 6.20 and 6.23. This is an improvement compared to the results of the single decision tree baseline model, especially so for the minority classes as is visible in table 6.2.

4.4 Support vector machine classifier (SVC)

The support vector machine classifier model was run on both the complete dataset as well as the segmented dataset and both these models were also tuned using gridsearchCV to determine the optimal hyperparameters.

Running the model on the segmented dataset generally caused class imbalance due to the model overfitting to the majority class. The results for the *perfect* class were comparable to the baseline and random forest model, however, the minority classes were completely underrepresented with true positive values of 0 occurring multiple times in the confusion matrices as visible in table 6.20 or in figures 6.32, 6.35 or 6.38.

The best model was tuned and run on the complete dataset for the best results. This resulted in a true score for the *perfect* class being around1, the *early* class being 0.99 and *late* class being 1 as is visible in the confusion matrices in 6.29. Due to these values being so high, they were checked to see if overfitting was happening. This was done using the confusion matrix to see if the false positives and negatives were low. Furthermore, a cross-validation was run, which resulted in similar values of 0.95+ for the true positives. Thus could be concluded that the model was not overfitted. In general, the model is a significant improvement compared to the baseline model for every class, especially so for the minority classes.

5. Conclusion

In this thesis, we looked at using classification machine learning methods to classify whether the estimated time for a turnover in a specific situation was sufficient. The machine learning methods used were random forest classification, support vector machine classification and single decision tree as a baseline.

The random forest and single decision tree models benefited from splitting the data, this resulted in better results for the minority classes. The support vector machine model gave the best results on the complete data. Both the random forest and support vector machine model were tuned and this improved the results for the minority classes especially.

The support vector machine model was by far the best model for this situation, being able to accurately predict each of the classes with a 99% accuracy without overfitting. So to answer the research question, I would recommend the use of a support vector machine model to support the turnover time decision-making workflow of the NS. This model could be used when creating the timetable to verify that the estimated turnover times are correct for the situation. Thus leading to a more accurate and robust timetable due to fewer unexpected delays occurring.

5.0.1 Discussion

Due to the imbalance of the occurrences of the classes, the single decision tree and RF model had class imbalance problems and trouble predicting the minority classes. This is a limitation of the data which might be solved by proportioning the classes more equally. Unfortunately, the significance of the class imbalance was noticed too late in the research to experiment with different data selection methods such as downsampling or upweighting the minority classes (“Imbalanced data”, n.d.) (Brown & Mues, 2012).

Additional features could also improve the data. Data about the number of passengers, the weather, events in the area of the station, time of year etc. might add more explainability and robustness to the models. This combined with more experimenting with feature composition and multicollinearity might improve the results as well.

As was mentioned before the support vector machine model had by far the best results, but it was also the most expensive to run. This was especially the case when it was combined with gridsearchCV to determine the best hyperparameters. Were this to be attempted again, I would recommend running the model on better hardware, implementing the gridsearch concurrently or using an alternative hyperparameter tuning method such as random search or bayseian search to reduce the time-cost.

There are variations to the SVM models such as NOT-ABBREV (LMDRT-SVM) that in some cases have shown superiority in both training speed and effectiveness (Wang et al., 2017). However, due to LMDRT-SVM having significantly less documentation and implementation guidelines, it was chosen to use SVC. The scale of this thesis didn't allow time to try this more unfamiliar model.

Bibliography

Nederlandse spoorwegen. (n.d.). Retrieved June 24, 2024, from https://en.wikipedia.org/wiki/Nederlandse_Spoorwegen

Kopmaken, omlopen, draaien en driehoeken. (n.d.). Retrieved June 21, 2024, from https://www.nicospilt.com/index_kopmaken.htm

Cadarso, L., & Ángel, M. (2012). Integration of timetable planning and rolling stock in rapid transit networks. *Annals of operations research 199: 113-135*. https://doi.org/https://doi.org/10.1007/s10479-011-0978-0

Brännlund, U., Lindberg, P., Nõu, A., & Nilsson, J. (1998). Railway timetabling using lagrangian relaxation. *Transportation Science 32(4):358-369*. https://doi.org/RailwayTimetablingUsingLagrangianRelaxation.

Canca, D., Barrena, E., Algaba, E., & Zarzo, A. (2014). Design and analysis of demand-adapted railway timetables. *Journal of Advanced Transportation, 48(2), 119–137*. https://doi.org/https://doi.org/10.1002/atr.1261

Kroon, L., Huisman, D., Abbink, E., Fioole, P., Fischetti, M., Maróti, G., Schrijver, A., Steenbeek, A., & Ybema, R. (2009). The new dutch timetable: The or revolution. *Interfaces, 39(1), 6–17*. https://doi.org/https://doi.org/10.1287/inte.1080.0409

Planning van treinen | reisinformatis | ns. (n.d.). Retrieved June 22, 2024, from https://www.ns.nl/reisinformatie/service-verbeteren/planning-van-treinen.html

Zhang, M., Wang, Y., Su, S., Tang, T., & Ning, B. (2018). A short turning strategy for train scheduling optimization in an urban rail transit line: The case of beijing subway line 4. *Journal of Advanced Transportation, 2018, 1–19*. https://doi.org/https://doi.org/10.1155/2018/5367295

Yuhua, Y., Marcella, S., Dario, P., & Shaoquan, N. (2022). Train timetabling with passenger data and heterogeneous rolling stocks circulation on urban rail transit line. *Soft Computing, 27(18), 12959–12977*. https://doi.org/https://doi.org/10.1007/s00500-022-07057-0

Sun, L., Jin, J., Lee, D., Axhausen, K., & Erath, A. (2014). Demand-driven timetable design for metro services. *Transportation Research. Part C, Emerging Technologies, 46, 284–299*. https://doi.org/https://doi.org/10.1016/j.trc.2014.06.003

Chen, Z., Li, S., D'Ariano, A., & Yang, L. (2022). Real-time optimization for train regulation and stop-skipping adjustment strategy of urban rail transit lines. *Omega, 110, 102631*. https://doi.org/https://doi.org/10.1016/j.omega.2022.102631

Jiang, Z., Tan, Y., Wang, F., & Bu, L. (2015). Turnback capacity assessment and delay management at a rail transit terminal with two-tail tracks. *Mathematical Problems in Engineering, 2015, 1–12*. https://doi.org/https://doi.org/10.1155/2015/369767

van Hassel, O. (2019). Predicting the turnaround time of an aircraft: A process structure aware approach. https://doi.org/https://research.tue.nl/en/studentTheses/predicting-the-turnaround-time-of-an-aircraft

Hossin, M., & Sulaiman, M. (2015). A review on evaluation metrics for data classification evaluations. *International journal of data mining knowledge management process, 5(2), 1*. https://doi.org/https://d1wqtxts1xzle7.cloudfront.net/37219940/5215ijdkp01-libre.pdf?1428316763=&response-content-disposition=inline%3B+filename%3DA_REVIEW_ON_EVALUATION_METRICS_FOR_DATA.pdf&Expires=1719542648&Signature=Lp~Tp-5t5x0t0SnB1CetSCC2injJJrPAjv0hpeE-b-PLNeuraxwd9StBqDIyVyjW5fHsCiSYGa2ngy9ywE5kCREIjpVEElPgeEnfRlF5SeLx3ML9IHS9-cV1p4DhQQ3iPH957mQy5Vt5ivfcQ0h~ZYOOyGPebi0uO7EY5aeYrKjkTGiWESXkqDOcGph0h8BeBAnQu~12XLx895s8OBP7LzWOOGTf5EvCzJV0c7aytSomcBxVViWy2iLYgJOItOuQx8wKr8~6leu4smkGoCJfG-p3rS9QAEwnmJH35KF1v0BO8S90LOLQPB417-rbh-eG9jnmE5istYGTFMoYVMSYHg__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

contributors, W. (n.d.). Decision tree [Accessed: 2024-06-24]. https://en.wikipedia.org/wiki/Decision_tree

Iftikhar, A., Mohammad, B., Muhammad, J., & Aneel, R. (2018). Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection. *IEEE Access*. https://doi.org/10.1109/ACCESS.2018.2841987

Aroef, C., Rivan, Y., & Rustam, Z. (2020). Comparing random forest and support vector machines for breast cancer classification. *Aroef | TELKOMNIKA (Telecommunication Computing Electronics and Control)*. https://doi.org/http://doi.org/10.12928/telkomnika.v18i2.14785

Wang, H., Gu, J., & Wang, S. (2017). An effective intrusion detection framework based on svm with feature augmentation. *Knowledge-based Systems, 136, 130–139*. https://doi.org/https://doi.org/10.1016/j.knosys.2017.09.014

Zhang, S., Sadaoui, S., & Mouhoub, M. (2015). An empirical analysis of imbalanced data classification. *Computer and Information Science, 8(1), 151*.

Weerts, H., Mueller, A., & Vanschoren, J. (2020). Importance of tuning hyperparameters of machine learning algorithms. *ArXiv*. https://doi.org/https://doi.org/10.48550/arXiv.2007.07588

RandomForestClassifier [Accessed: 2024-06-25]. (n.d.). https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Gridsearchcv documentation [Accessed: 2024-06-24]. (n.d.). https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

Imbalanced data [Accessed: 2024-06-28]. (n.d.). https://developers.google.com/machine-learning/data-prep/construct/sampling-splitting/imbalanced-data

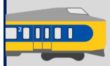
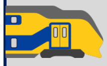
Brown, I., & Mues, C. (2012). An experimental comparison of classification algorithms for imbalanced credit scoring data sets. *Expert systems with applications*, *39*(3), 3446–3453. https://doi.org/10.1016/j.eswa.2011.09.033

6. Appendix A

6.1 Link to github repository

Hyperlink to github repository with the code used to create the models

6.2 Calculating Norm Times pdf

Materieel	Lengte	Bakken	in 27m bakken	HC-Norm	Configuratie	Keertijd zonder Wisselmachinist	Keertijd met Wisselmachinist	Combineren		Splitsen	
								bodem-waarde	norm-waarde	bodem-waarde	norm-waarde
Intercitymaterieel											
Materieelgebonden stop: 0,9 minuut - Maximaal 5 in treinschakeling											
ICM 	80,6 m	3		1	Dienstgroep: OH	5 min	3 min	3 min	3,5 min	2 min	2,5 min
	107,1 m	4			Dienstgroep: OC						
	161,2 m	6				6 min					
	187,7 m	7									
	214,2 m	8		2		7 min					
	241,8 m	9									
	268,3 m	10				8 min					
	294,8 m	11									
	321,3 m	12									
	322,4 m	12									
	348,9 m	13		3		9 min					
	375,4 m	14									
	403,0 m	15				10 min					
	401,9 m	15									
Verlengd Interregiomaterieel											
Materieelgebonden stop: 0,9 minuut - Maximaal 3 in treinschakeling											
VIRM 	108,6 m	4		1	Dienstgroep: AD	5 min	3 min	3 min	3,5 min	2 min	2,5 min
	162,1 m	6			Dienstgroep: OA	6 min					
	217,2 m	8		2		7 min					
	270,7 m	10									
	324,2 m	12				8 min					
	325,8 m	12		3							
	379,3 m	14				9 min					
Dubbeldekker Zonering											
Materieelgebonden stop: 0,9 minuut - Maximaal 2 in treinschakeling											
DDZ	101,1 m	4		1	Dienstgroep: LBM	5 min	3 min	4 min	5 min	2 min	2,5 min
	153,9 m	6			Dienstgroep: LAM	6 min					
	202,2 m	8		2							
	255,0 m	10				7 min					
	307,8 m	12		3		8 min					
TRAXX + Intercityruituigen											
Materieelgebonden stop: 1,0 minuut											
TRAXX +ICR	196,2 m	6	7,3	2	Dienstgroep: ARA	9 min	5 min	-	-	-	-
	222,6 m	7	8,2		Dienstgroep: IDB	10 min					
	249,0 m	8	9,2		Dienstgroep: IDA	-					
	275,4 m	9	10,2		Dienstgroep: GEA	11 min					
Intercity Nieuwe Generatie											
Materieelgebonden stop: 0,9 minuut - Maximaal 3 in treinschakeling											
ICNG	110,0 m	5	4,1	1	Dienstgroep: WD	6 min	4 min	3 min	4 min	2,5 min	3,5 min
	165,5 m	8	6,1		Dienstgroep: WA	7 min					
	220,0 m	10	8,1	2		8 min					
	275,5 m	13	10,2			9 min					
	330,0 m	15	12,2								
	331,0 m	16	12,3								

Materieel	Lengte	Bakken	in 27m bakken	HC-Norm	Configuratie	Keertijd zonder Wisselmachinist	Keertijd met Wisselmachinist	Combineren		Splitsen	
								bodem-waarde	norm-waarde	bodem-waarde	norm-waarde
Sprinter Lighttrain											
Materieelgebonden stop: 0,7 minuut - Maximaal 3 in treinschakeling											
SLT	69,4 m	4	2,6	1	Dienstgroep: LE	5 min	4 min	3 min	4 min	4 min	5 min
	100,5 m	6	3,7		Dienstgroep: LC						
	138,8 m	8	5,1			6 min					
	169,9 m	10	6,3								
	201,0 m	12	7,4	2		7 min					
	208,2 m	12	7,7								
	239,3 m	14	8,9								
	270,3 m	16	10,0			8 min					
Sprinter Nieuwe Generatie											
Materieelgebonden stop: 0,7 minuut - Maximaal 3 in treinschakeling											
SNG	59,5 m	3	2,2	1	Dienstgroep: LG	4 min	2 min	3 min	3,5 min	2 min	3 min
	75,7 m	4	2,8		Dienstgroep: LF						
	119,0 m	6	4,4								
	135,2 m	7	5,0			5 min					
	151,4 m	8	5,6								
	178,5 m	9	6,6								
	194,7 m	10	7,2	2		6 min					
	210,6 m	11	7,8								
	227,1 m	12	8,4								
Flinker Leichter Innovativer Regionaltriebzug											
Materieelgebonden stop: 0,7 minuut - Maximaal 3 in treinschakeling											
FLIRT	63,2 m	3	2,3	1	Dienstgroep: MW	3 min	2 min	3 min	3 min	2 min	3 min
	80,7 m	4	3,0		Dienstgroep: MC	4 min					
	126,4 m	6	4,7								
	143,9 m	7	5,3			5 min					
	161,4 m	8	6,0								
	189,6 m	9	7,0	2							
	207,1 m	10	7,7			6 min					
	224,6 m	11	8,3								
	242,1 m	12	9,0								
Flinker Leichter Innovativer Regionaltriebzug - Treindienst Alphen-Gouda											
Materieelgebonden stop: 0,5 minuut - Maximaal 3 in treinschakeling											
FLIRT TAG	45,7 m	2	1,7	-	Dienstgroep: TAG	3 min	2 min	3 min	3 min	2 min	2 min
	91,4 m	5	3,4			4 min					
	137,1 m	6	5,1								

6.3 Data Exploration Figures

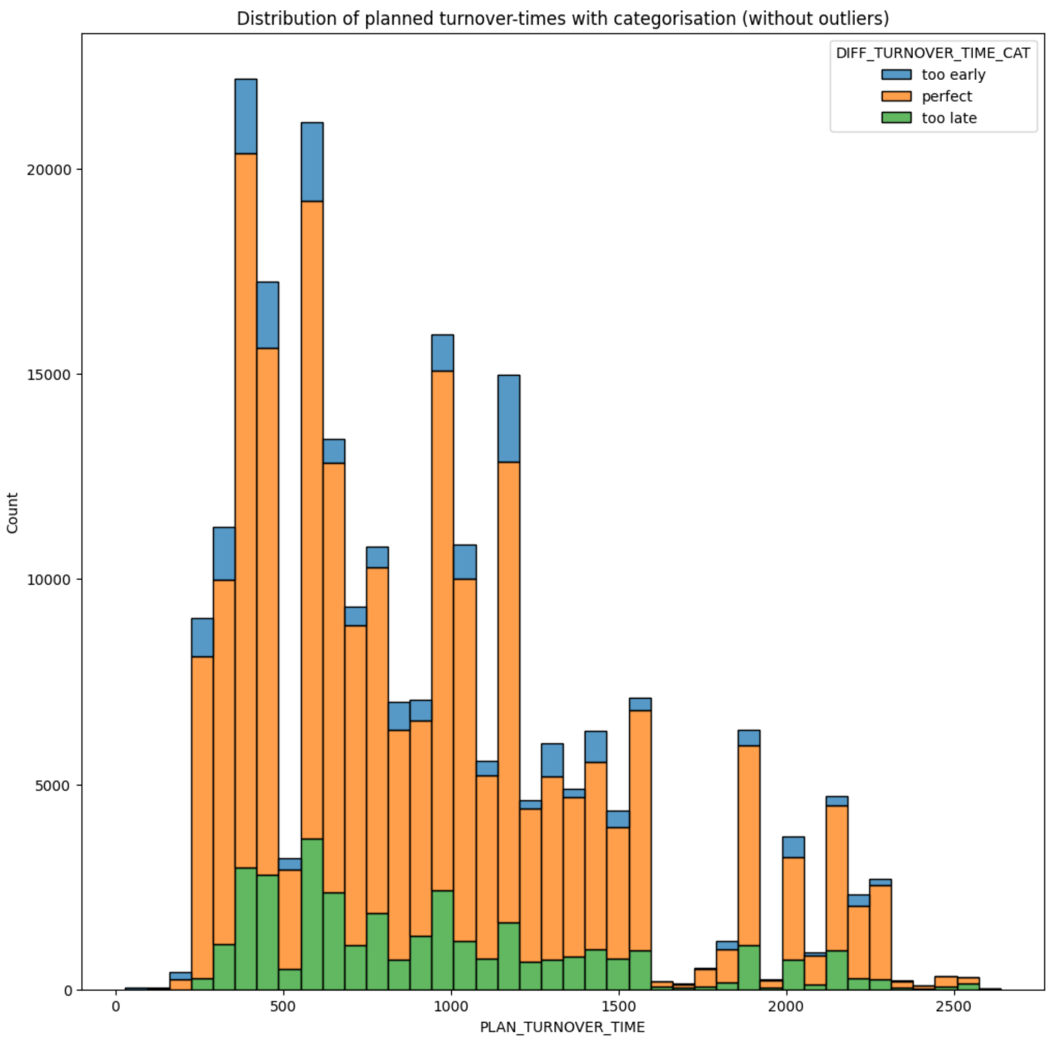


Figure 6.1: Distribution of data with turnover time categorisation

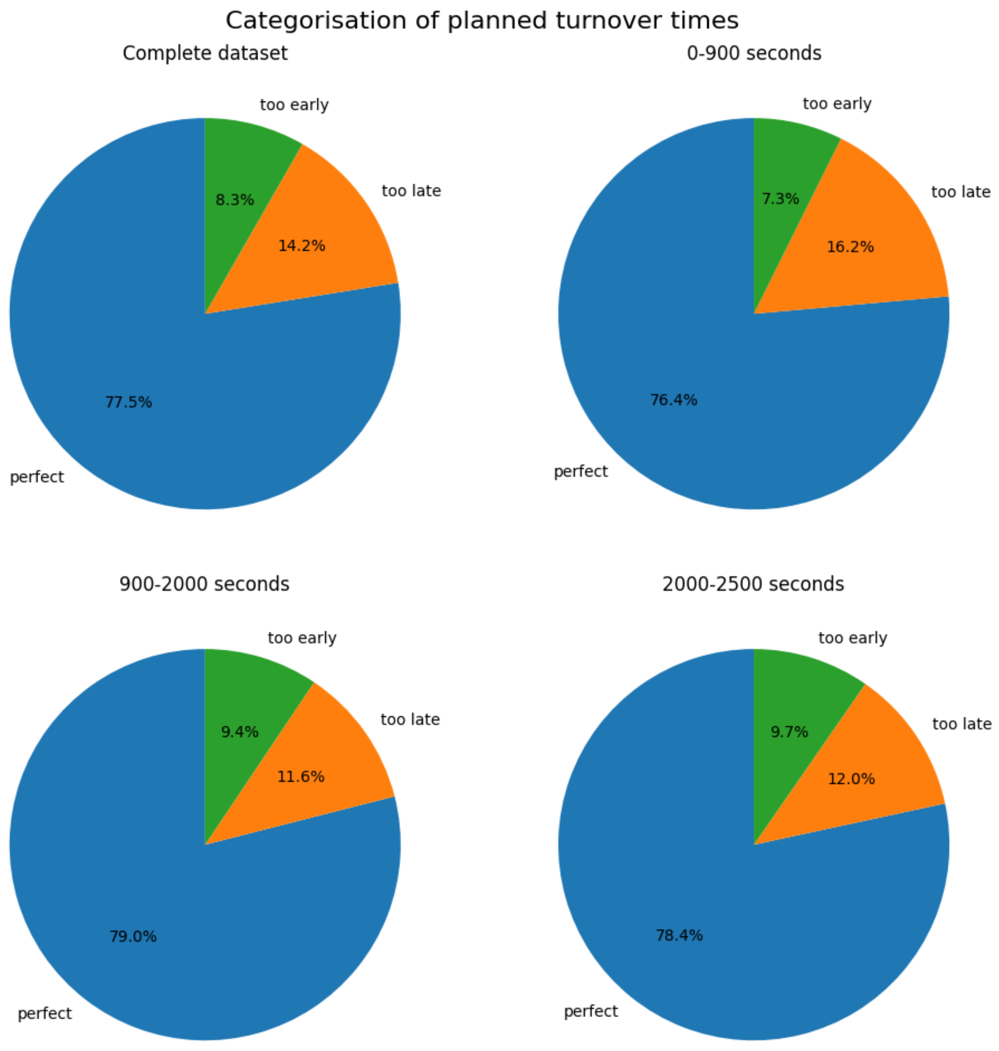
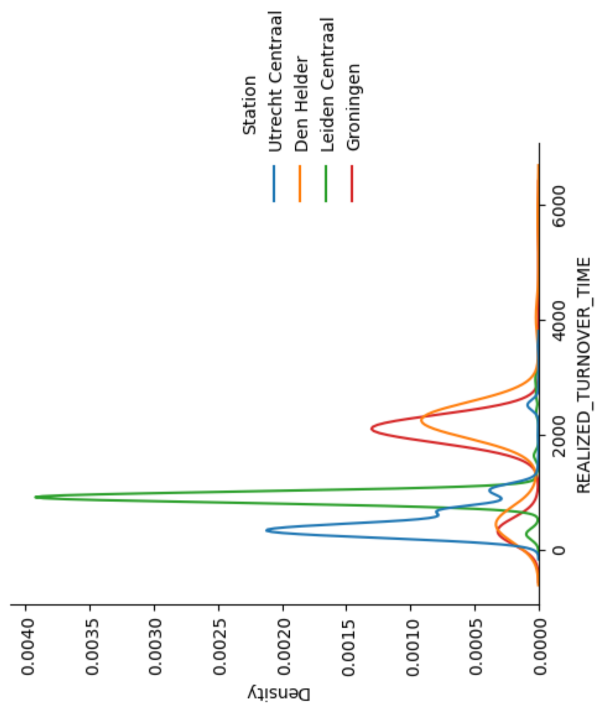


Figure 6.2: Multiple pie charts showing the distribution of the categorised turnover-times for the complete and splitted datasets



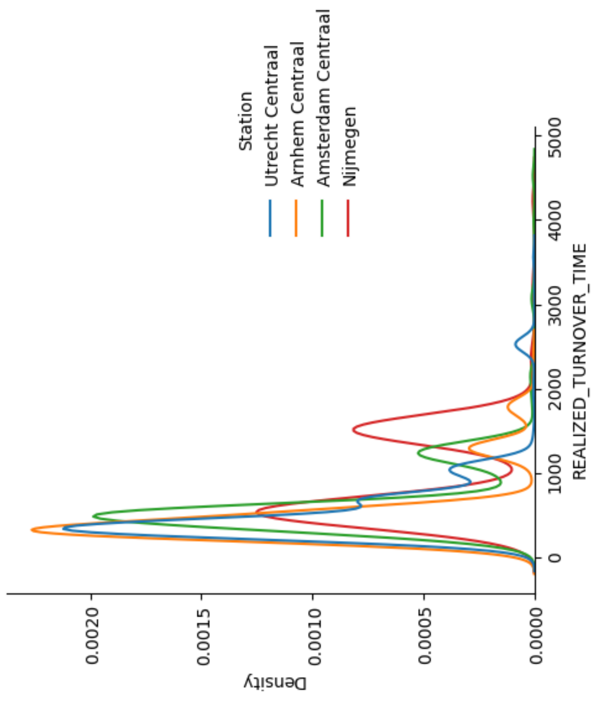


Figure 6.3: Distribution of turnover times at various stations categorized into common distribution patterns (left) and irregular distributions (right), with Utrecht Centraal for reference.

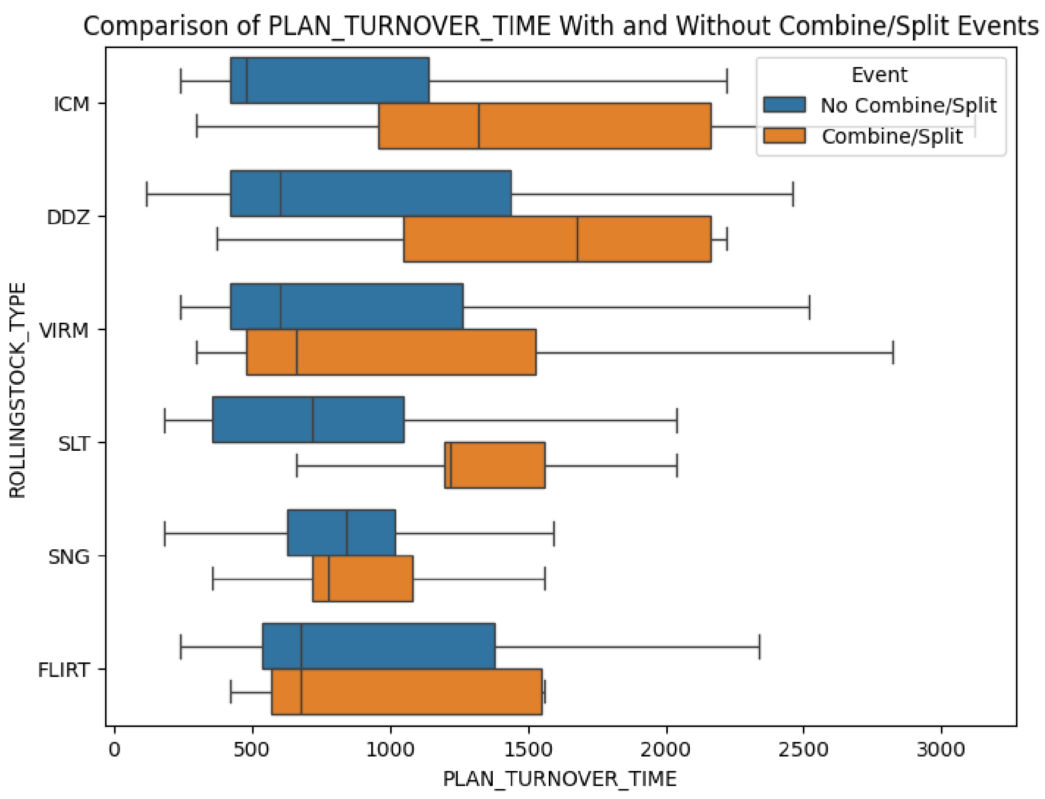


Figure 6.4: Boxplot showing the planned turnover times with and without combine/ split events

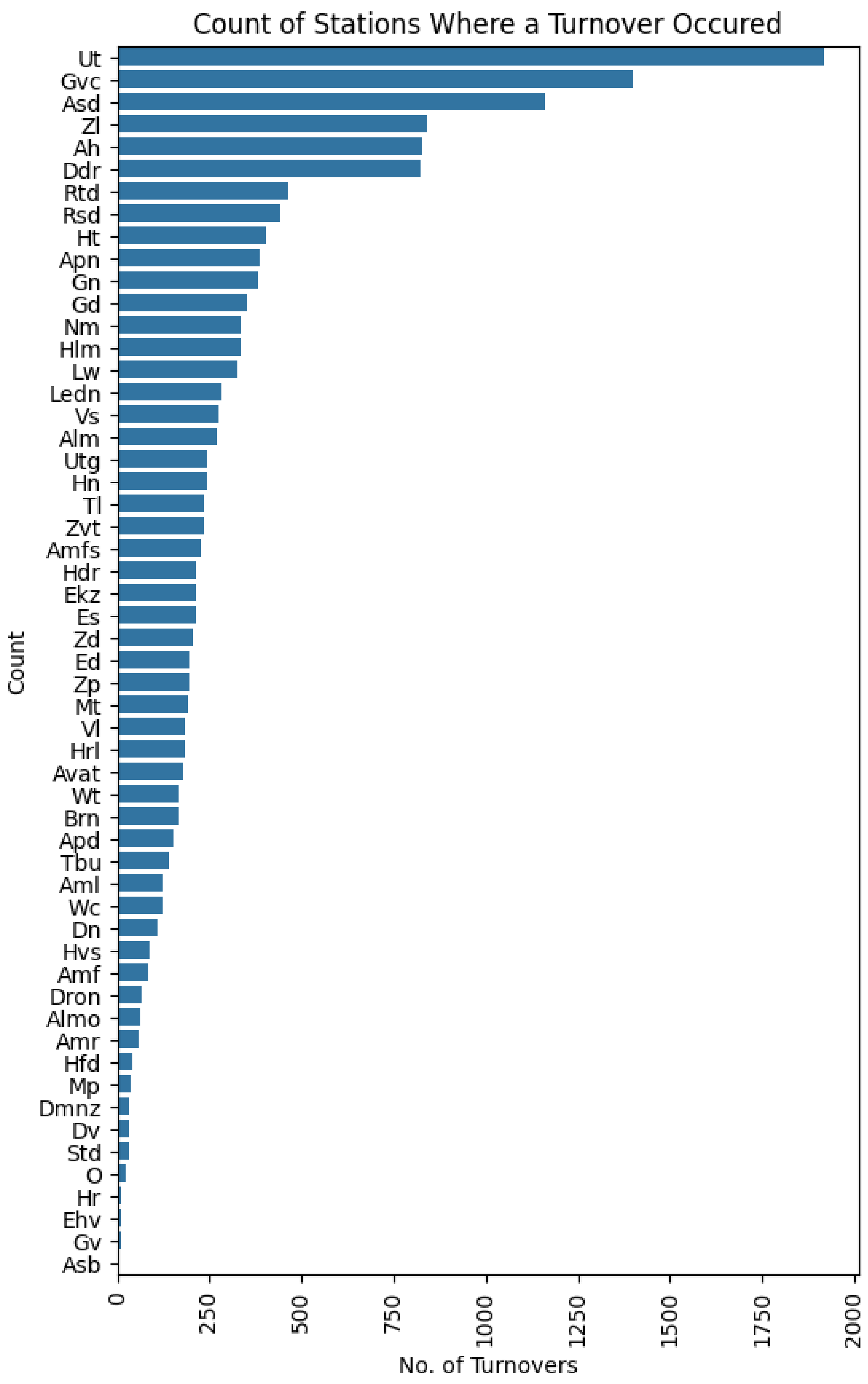


Figure 6.5: Distribution of the turnover per station

6.4 Feature importance Figures

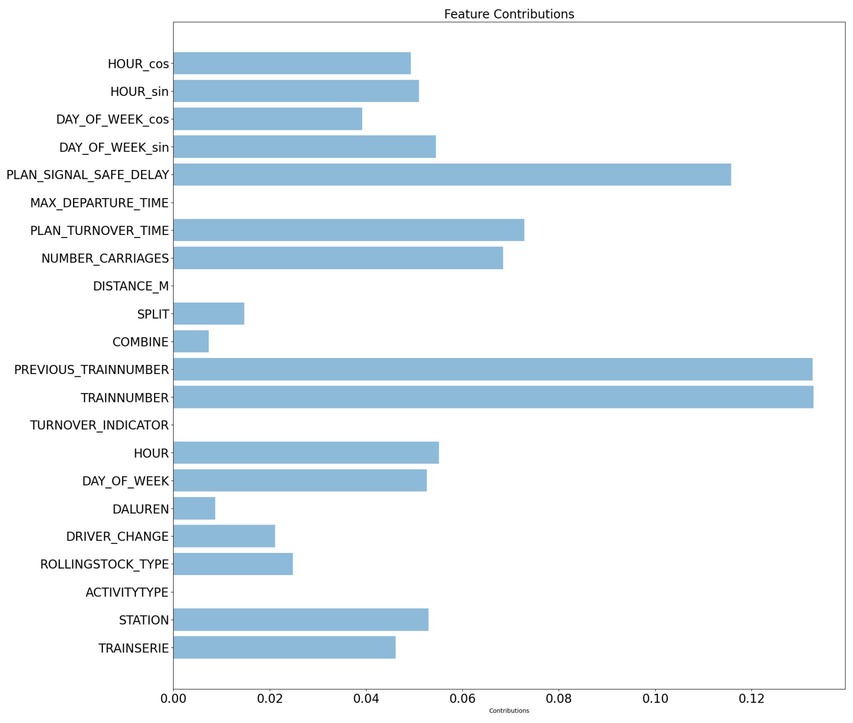


Figure 6.6: Feature importance, all possible features

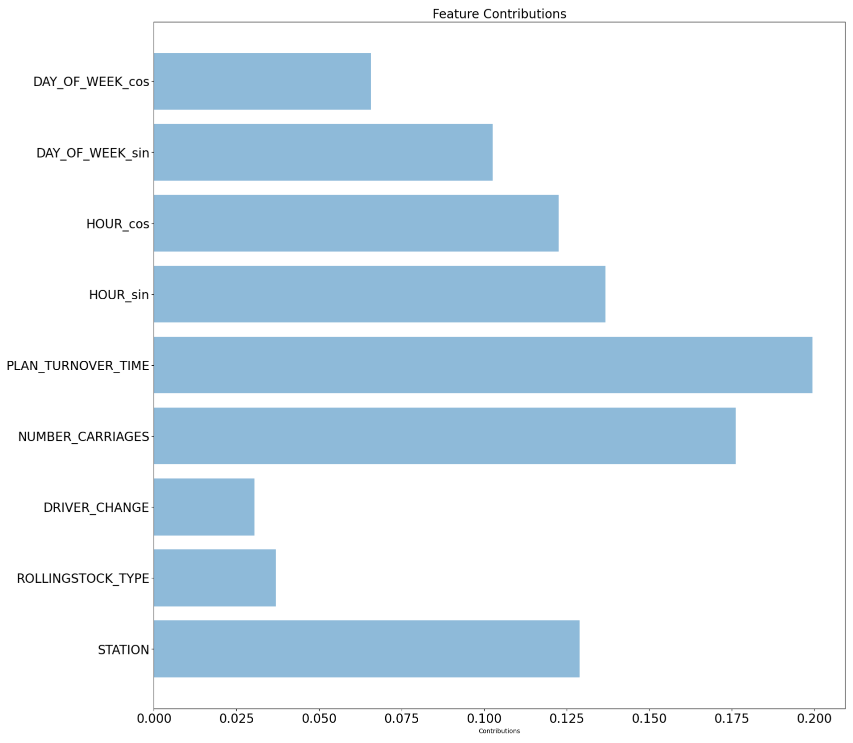


Figure 6.7: Feature importance, dream-tool features

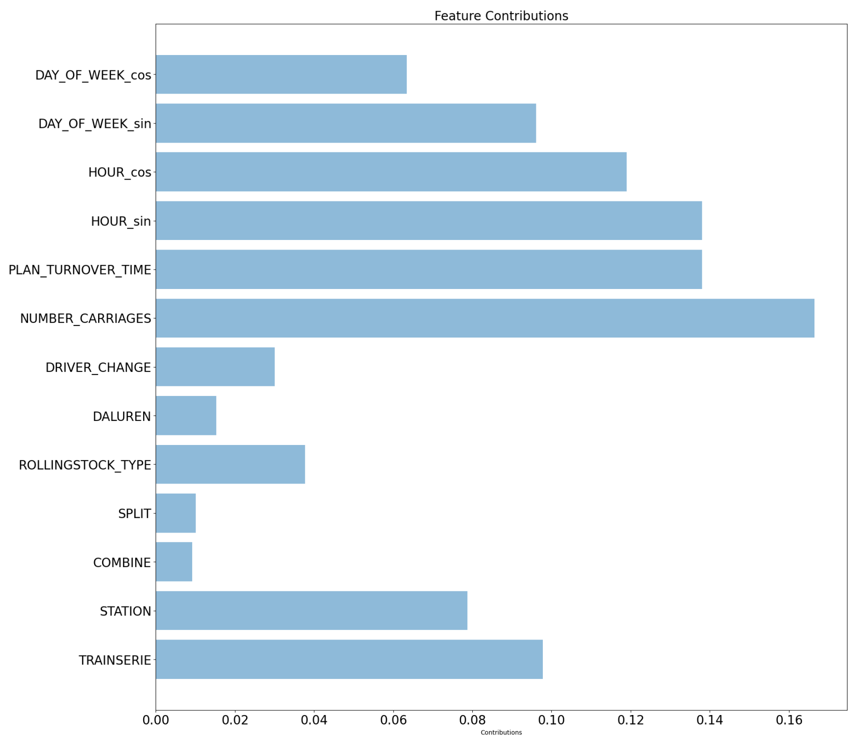


Figure 6.8: Feature importance, final-selection features

6.5 Best performing models metrics tables

6.5.1 Models classification report comparison

Model	**Dataset**	**Accuracy**	**Precision**	**Recall**	**F1-score**
Single Tree					
	Complete	0.73	0.68	0.73	0.69
	0-900s	0.73	0.69	0.74	0.71
	900-2000s	0.76	0.72	0.76	0.74
	2000-2500s	0.73	0.68	0.73	0.70
Random Forest					
	0-900s	0.78	0.75	0.78	0.72
	900-2000s	0.78	0.72	0.78	0.72
	2000-2500s	0.78	0.72	0.78	0.72
SVC					
	Complete	0.99	0.99	0.99	0.99

Table 6.1: Table showing the metrics of the best versions of each model

6.5.2 Models confusion matrix comparison

Model	Dataset	(TP) early	(TP) perfect	(TP) late
Single Tree				
	Complete	0.26	0.79	0.27
	0-900s	0.21	0.79	0.45
	900-2000s	0.33	0.82	0.35
	2000-2500s	0.28	0.80	0.28
Random Forest				
	0-900s	0.46	0.79	0.72
	900-2000s	0.59	0.81	0.68
	2000-2500s	0.67	0.80	0.45
SVC				
	Complete	1.00	0.99	01.00

Table 6.2: Table showing the TP of the different classes for each model. All these results

6.6 Single Decision Tree Classifier

6.6.1 Results model complete dataset

	precision	recall	f1-score	support
perfect	0.79	0.90	0.84	36752
too early	0.26	0.14	0.19	3927
too late	0.27	0.14	0.19	6677
accuracy			0.73	47356
macro avg	0.44	0.40	0.41	47356
weighted avg	0.68	0.73	0.70	47356

Table 6.3: Classification report

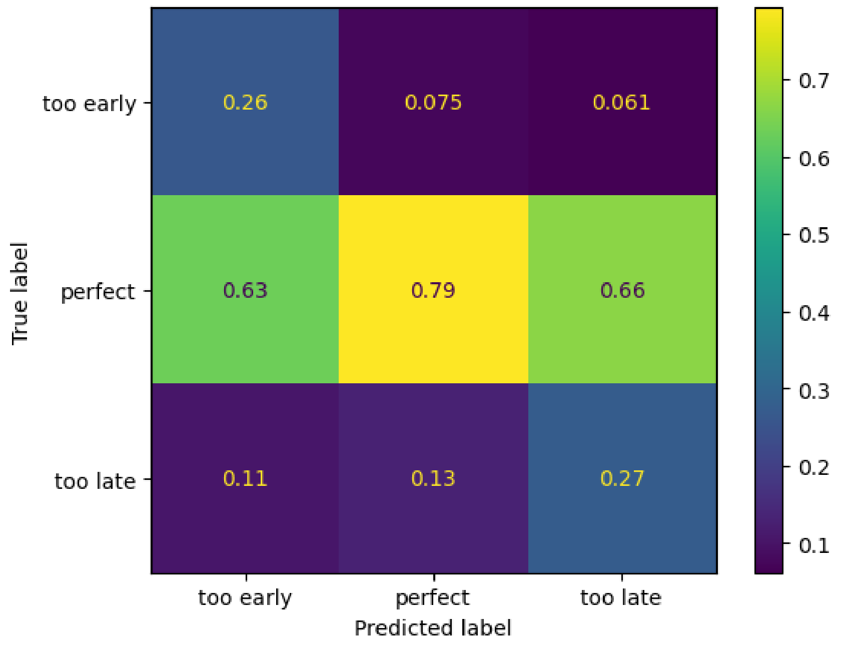


Figure 6.9: Confusion matrix

6.6.2 Confusion matrix comparison of dataset

Model	Dataset	(TP) early	(TP) perfect	(TP) late
Single Tree				
	Complete	0.26	0.79	0.27
	0-900s	0.21	0.79	0.45
	900-2000s	0.33	0.82	0.35
	2000-2500s	0.28	0.80	0.28

Table 6.4: Table showing the TP of the different classes for each dataset

6.6.3 Results model 0-900s dataset

	precision	recall	f1-score	support
perfect	0.79	0.91	0.85	19807
too early	0.21	0.11	0.15	1864
too late	0.45	0.24	0.32	4343
accuracy			0.74	26014
macro avg	0.48	0.42	0.44	26014
weighted avg	0.69	0.74	0.71	26014

Table 6.5: Classification report

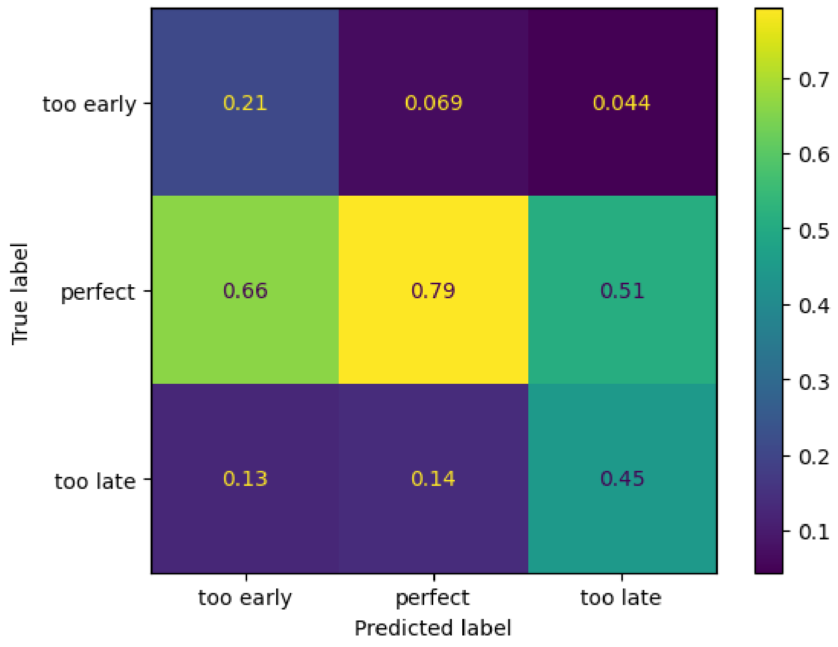


Figure 6.10: Confusion matrix

6.6.4 Results model 900-2000s dataset

	precision	recall	f1-score	support
perfect	0.79	0.91	0.85	19807
too early	0.21	0.11	0.15	1864
too late	0.45	0.24	0.32	4343
accuracy			0.74	26014
macro avg	0.48	0.42	0.44	26014
weighted avg	0.69	0.74	0.71	26014

Table 6.6: Classification report

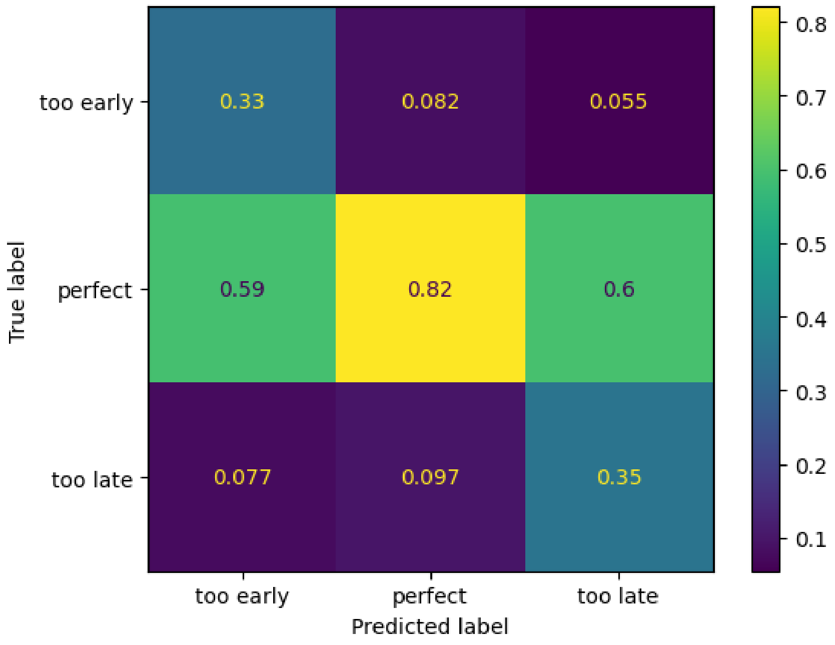


Figure 6.11: Confusion matrix

6.6.5 Results model 2000-2500s dataset

	precision	recall	f1-score	support
perfect	0.82	0.91	0.86	14665
too early	0.33	0.20	0.25	1752
too late	0.35	0.20	0.26	2078
accuracy			0.76	18495
macro avg	0.50	0.44	0.46	18495
weighted avg	0.72	0.76	0.74	18495

Table 6.7: Classification report

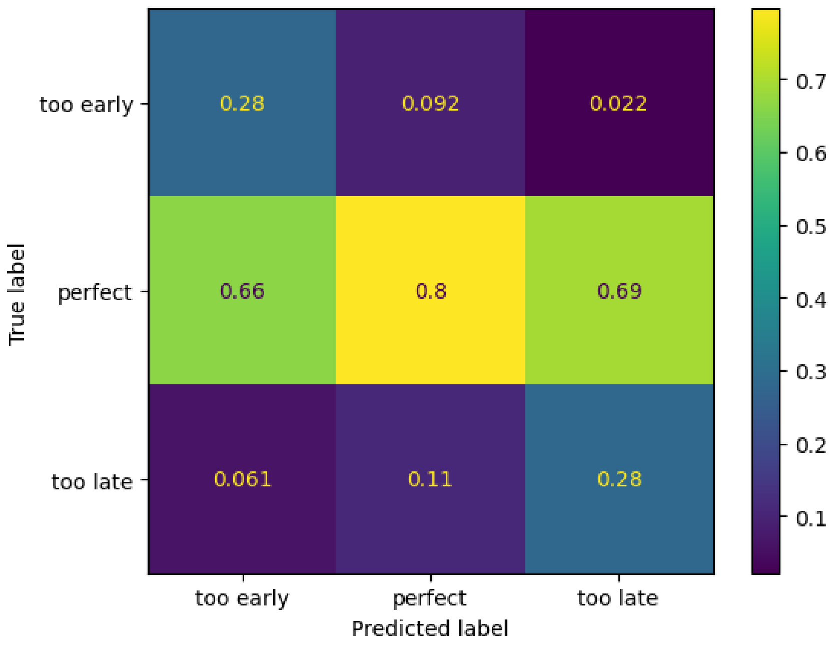


Figure 6.12: Confusion matrix

6.7 Random Forest Classifier

6.7.1 Confusion matrix comparison of datasets

Tuning	Dataset	(TP) early	(TP) perfect	(TP) late
On complete dataset				
	Complete	0.50	0.79	0.47
	0-900s	0.46	0.79	0.72
	900-2000s	0.59	0.81	0.68
	2000-2500s	0.67	0.80	0.45
On segmented dataset				
	0-900s	0	0.79	0.92
	900-2000s	0.53	0.82	0.55
	2000-2500s	0	0.76	0
Untuned				
	Complete	0.29	0.79	0.30
	0-900s	0.25	0.80	0.45
	900-2000s	0.41	0.82	0.35
	2000-2500s	0.38	0.81	0.28

Table 6.8: Table showing the TP of the different classes for each dataset

6.7.2 Results model complete dataset

6.7.2.1 Untuned

	precision	recall	f1-score	support
perfect	0.79	0.92	0.85	36624
too early	0.29	0.12	0.17	4025
too late	0.30	0.13	0.18	6707
accuracy			0.74	47356
macro avg	0.46	0.39	0.40	47356
weighted avg	0.68	0.74	0.70	47356

Table 6.9: Classification report untuned

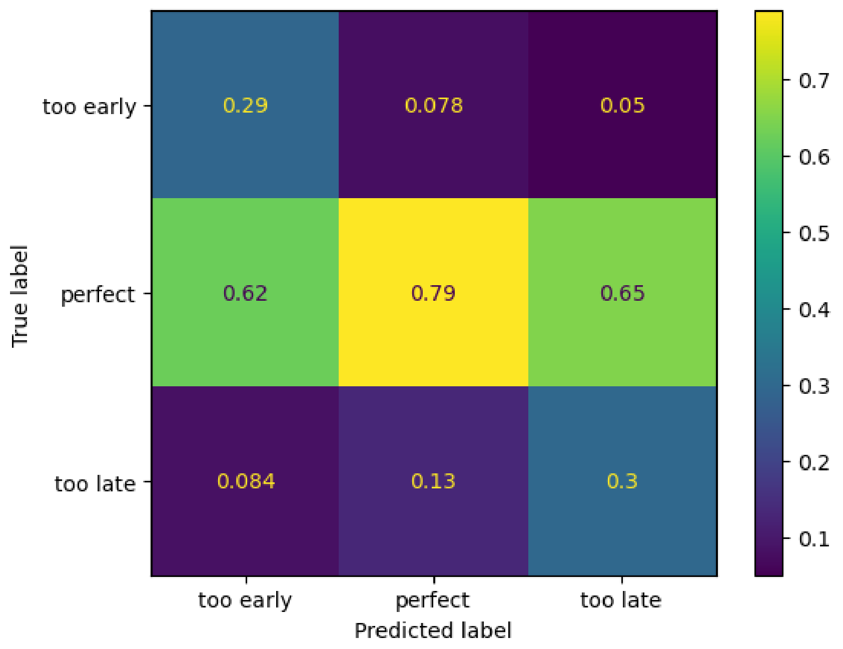


Figure 6.13: Confusion matrix untuned

6.7.2.2 Tuned

	precision	recall	f1-score	support
perfect	0.79	0.98	0.87	36793
too early	0.51	0.07	0.13	3930
too late	0.46	0.07	0.12	6633
accuracy			0.78	47356
macro avg	0.59	0.37	0.37	47356
weighted avg	0.72	0.78	0.71	47356

Table 6.10: Classification report tuned

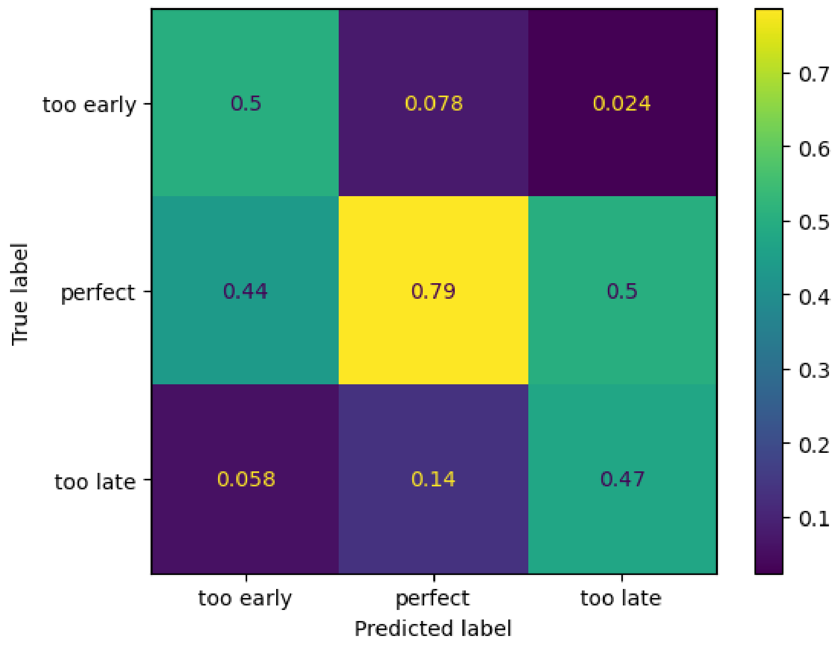


Figure 6.14: Confusion matrix tuned

6.7.3 Results model 0-900s dataset

6.7.3.1 Untuned

	precision	recall	f1-score	support
perfect	0.79	0.92	0.85	19859
too early	0.27	0.09	0.14	1946
too late	0.48	0.25	0.32	4209
accuracy			0.75	26014
macro avg	0.51	0.42	0.44	26014
weighted avg	0.70	0.75	0.71	26014

Table 6.11: Classification report untuned

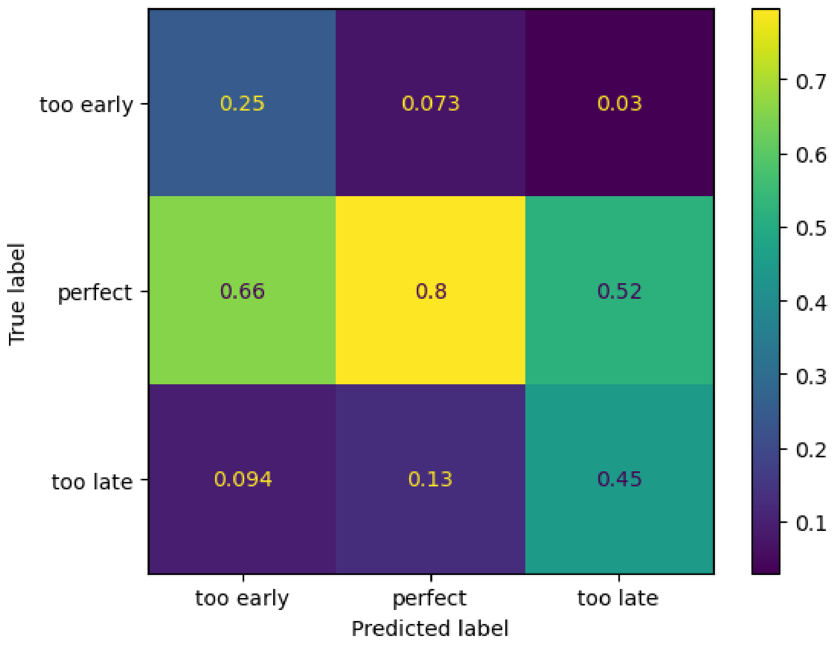


Figure 6.15: Confusion matrix untuned

6.7.3.2 Tuned on complete dataset

	precision	recall	f1-score	support
perfect	0.79	0.98	0.87	19823
too early	0.48	0.04	0.08	1929
too late	0.67	0.19	0.29	4262
accuracy			0.78	26014
macro avg	0.64	0.40	0.42	26014
weighted avg	0.74	0.78	0.72	26014

Table 6.12: Classification report tuned

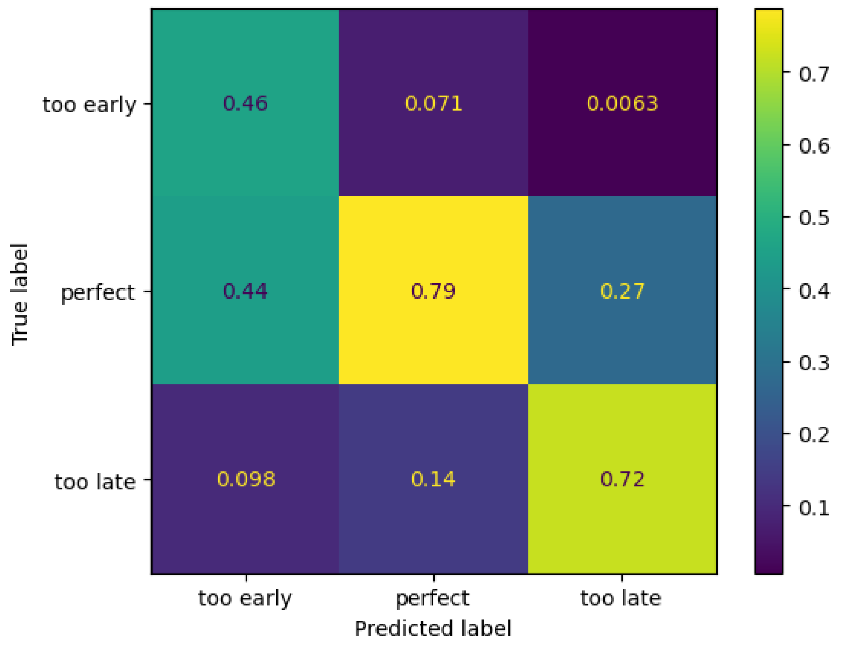


Figure 6.16: Confusion matrix tuned

6.7.3.3 Tuned on segment

	precision	recall	f1-score	support
perfect	0.79	1.00	0.88	19823
too early	0.00	0.00	0.00	1929
too late	0.92	0.13	0.23	4262
accuracy			0.79	26014
macro avg	0.57	0.38	0.37	26014
weighted avg	0.75	0.79	0.72	26014

Table 6.13: Classification report tuned

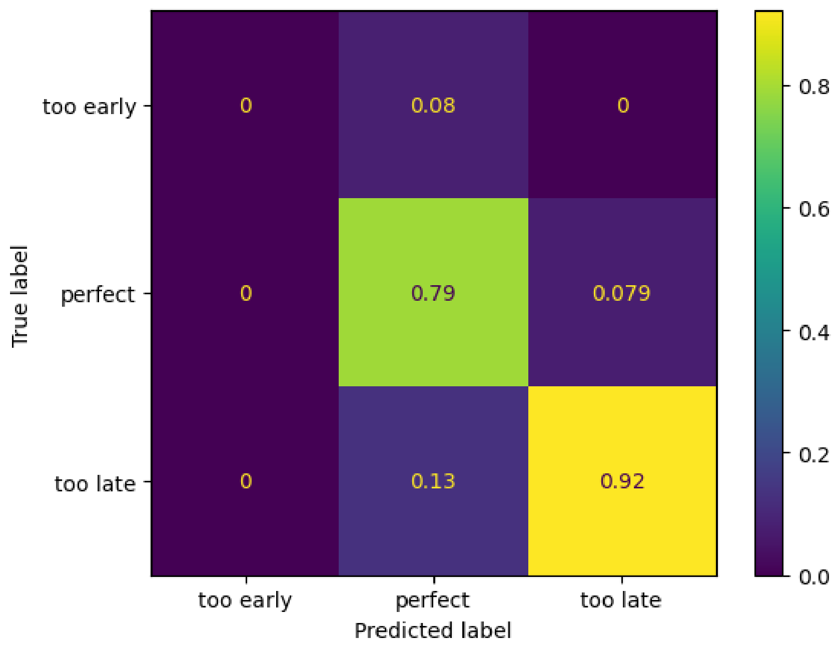


Figure 6.17: Confusion matrix tuned

6.7.4 Results model 900-2000s dataset

6.7.4.1 Untuned

	precision	recall	f1-score	support
perfect	0.82	0.93	0.87	14568
too early	0.46	0.25	0.32	1741
too late	0.41	0.19	0.26	2186
accuracy	0.78	18495		
macro avg	0.57	0.46	0.48	18495
weighted avg	0.74	0.78	0.75	18495

Table 6.14: Classification report untuned

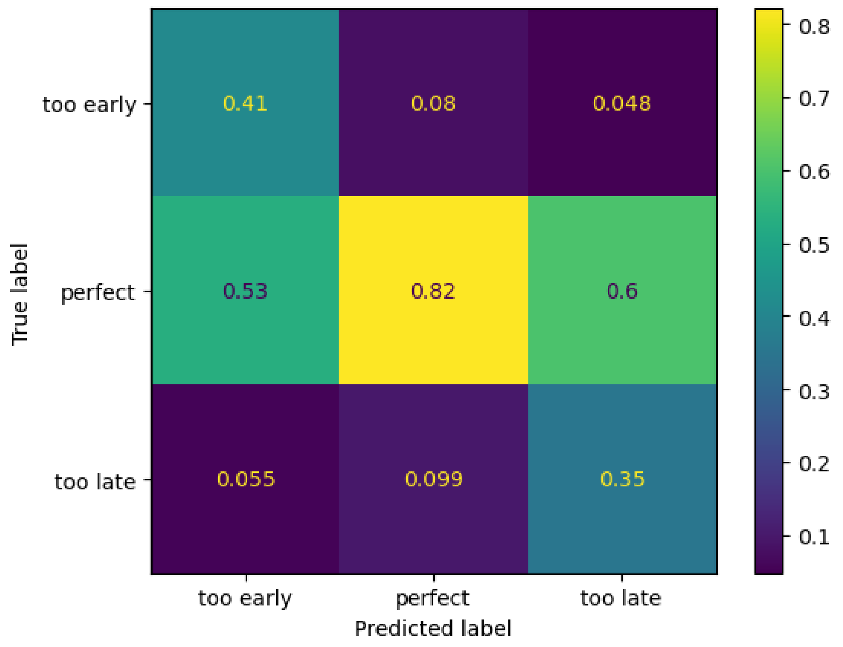


Figure 6.18: Confusion matrix untuned

6.7.4.2 Tuned on complete dataset

	precision	recall	f1-score	support
perfect	0.79	0.98	0.87	19823
too early	0.48	0.04	0.08	1929
too late	0.67	0.19	0.29	4262
accuracy			0.78	26014
macro avg	0.64	0.40	0.42	26014
weighted avg	0.74	0.78	0.72	26014

Table 6.15: Classification report tuned

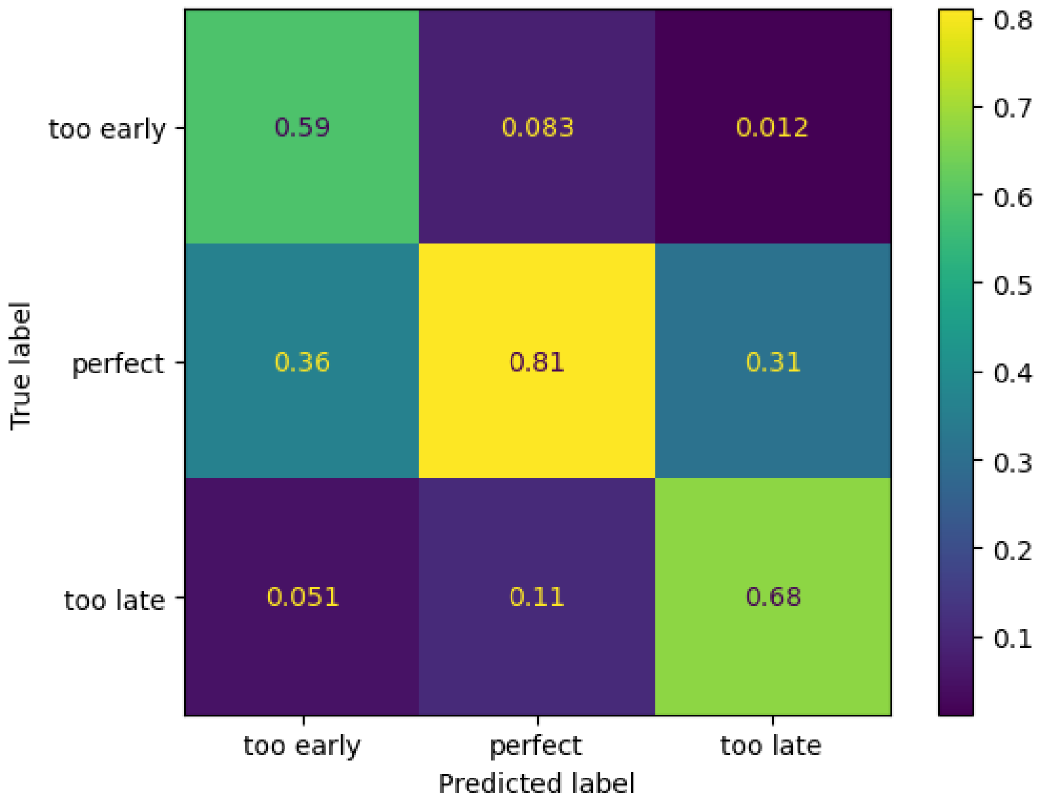


Figure 6.19: Confusion matrix tuned

6.7.4.3 Tuned on segment

	precision	recall	f1-score	support
perfect	0.82	0.96	0.88	19823
too early	0.53	0.29	0.37	1929
too late	0.55	0.10	0.17	4262
accuracy			0.80	26014
macro avg	0.63	0.45	0.47	26014
weighted avg	0.76	0.80	0.75	26014

Table 6.16: Classification report tuned

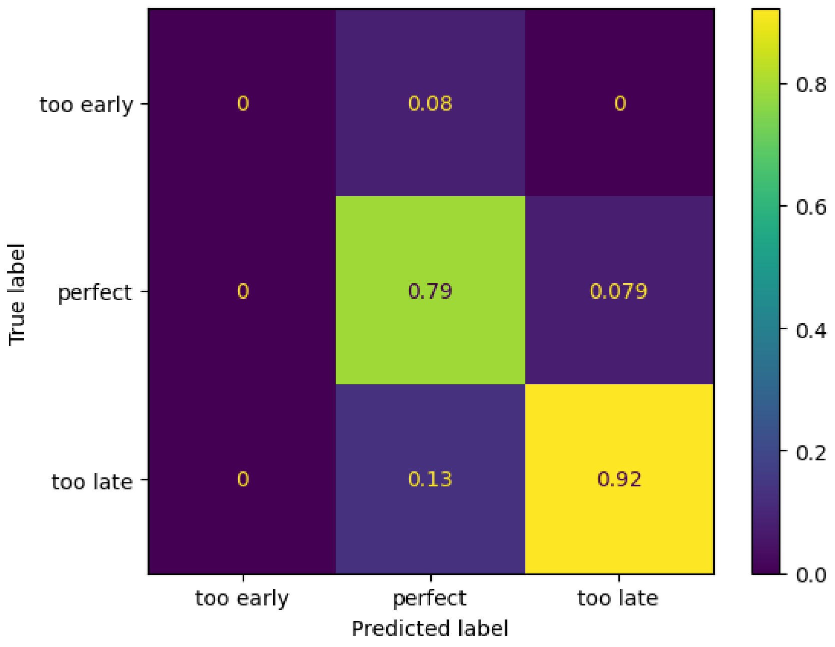


Figure 6.20: Confusion matrix tuned

6.7.5 Results model 2000-2500s dataset

6.7.5.1 Untuned

	precision	recall	f1-score	support
perfect	0.80	0.93	0.86	2230
too early	0.39	0.17	0.24	267
too late	0.36	0.16	0.22	351
accuracy	0.76	2848		
macro avg	0.52	0.42	0.44	2848
weighted avg	0.71	0.76	0.72	2848

Table 6.17: Classification report untuned

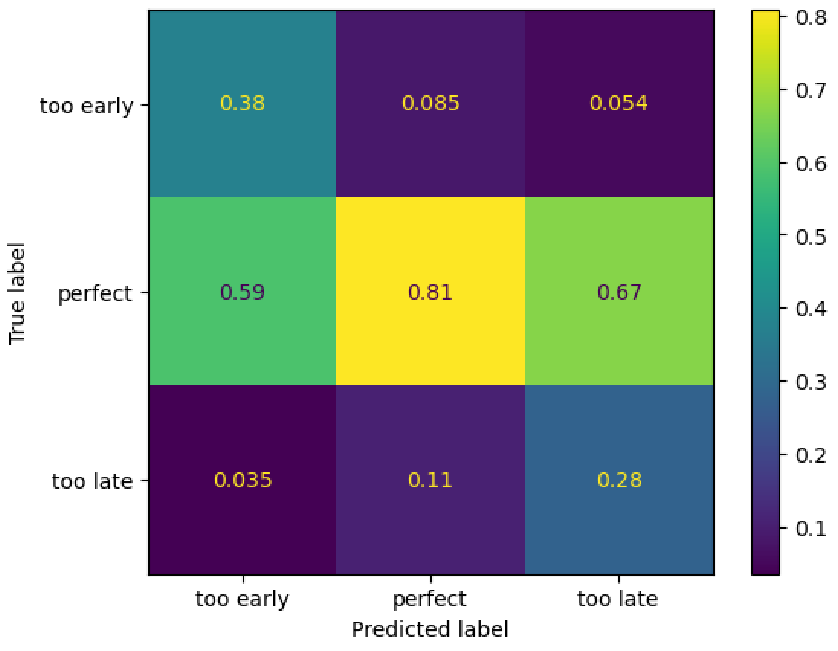


Figure 6.21: Confusion matrix untuned

6.7.5.2 Tuned

	precision	recall	f1-score	support
perfect	0.80	0.98	0.88	2236
too early	0.80	0.13	0.23	274
too late	0.55	0.11	0.18	338
accuracy			0.80	2848
macro avg	0.72	0.41	0.43	2848
weighted avg	0.77	0.80	0.74	2848

Table 6.18: Classification report tuned

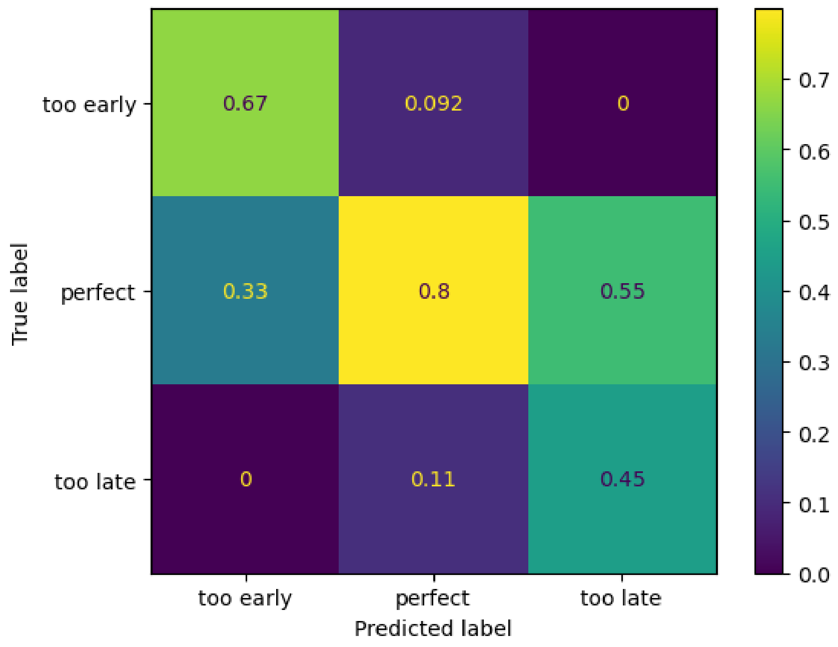


Figure 6.22: Confusion matrix tuned

6.7.5.3 Tuned on segment

	precision	recall	f1-score	support
perfect	0.76	1.00	0.86	19823
too early	0.00	0.00	0.00	1929
too late	0.00	0.00	0.00	4262
accuracy			0.76	26014
macro avg	0.25	0.33	0.29	26014
weighted avg	0.58	0.76	0.65	26014

Table 6.19: Classification report tuned

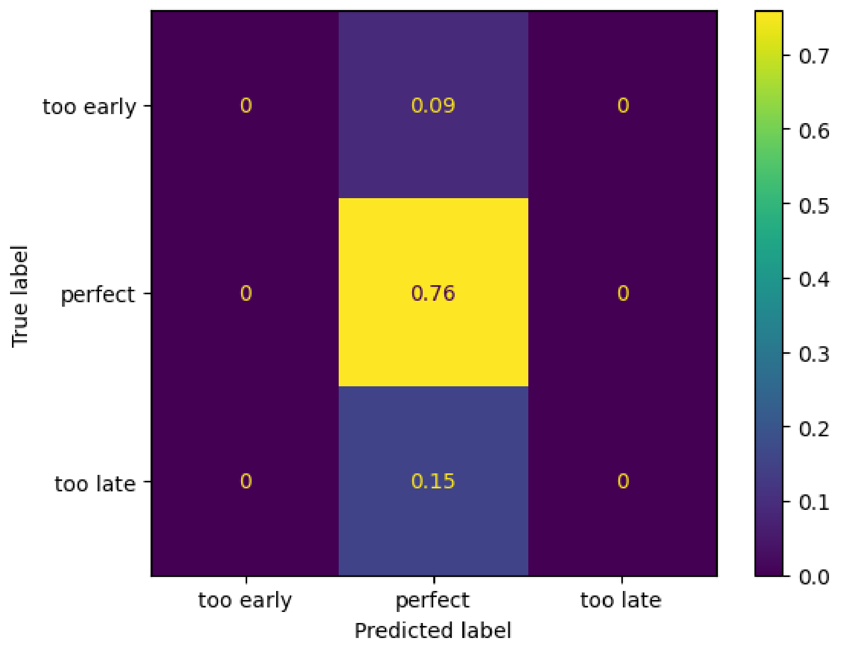


Figure 6.23: Confusion matrix tuned

6.8 Support Vector Machine Classifier

6.8.1 Kernel comparison on 1 week of the dataset

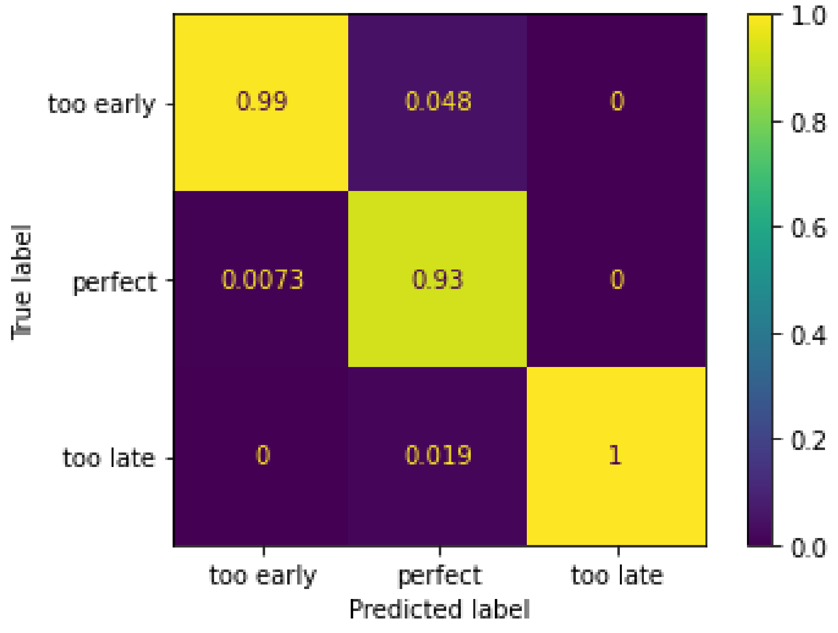


Figure 6.24: Confusion matrix linear kernel

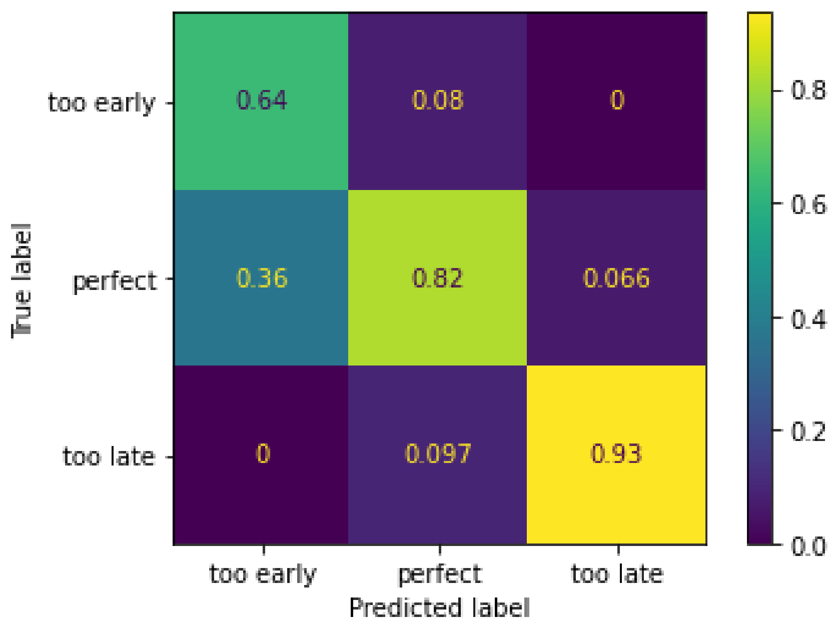


Figure 6.25: Confusion matrix polynomial kernel

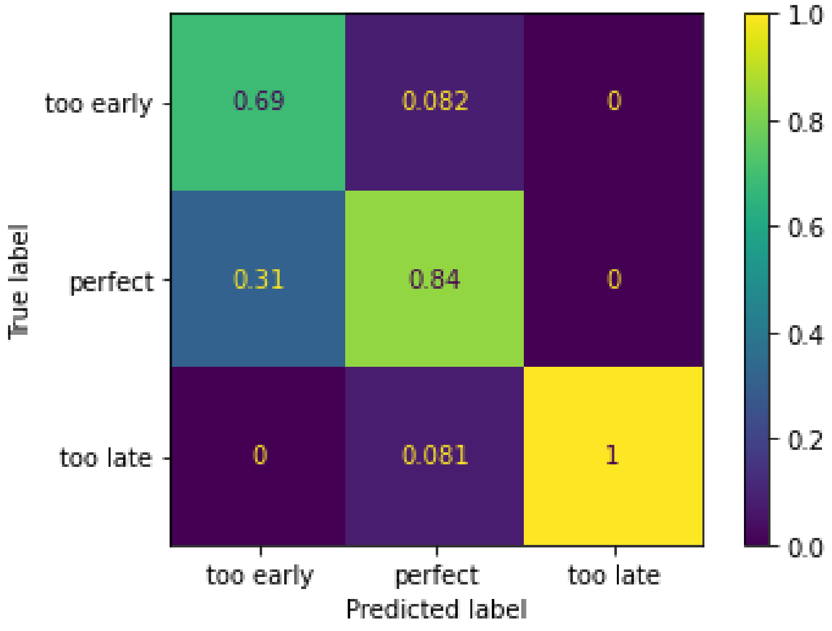


Figure 6.26: Confusion matrix radial kernel

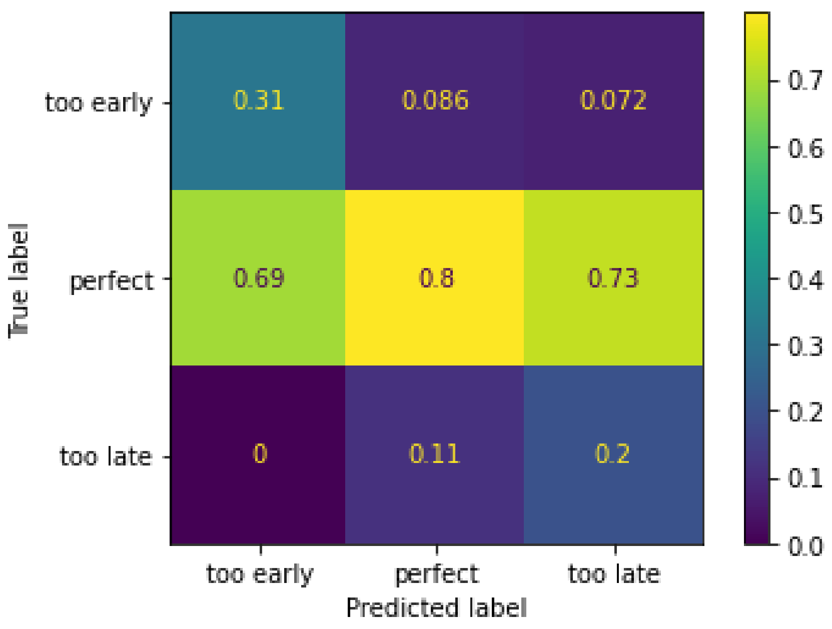


Figure 6.27: Confusion matrix sigmoid kernel

6.8.2 Confusion matrix comparison of datasets

Tuning	Dataset	(TP) early	(TP) perfect	(TP) late
On complete week dataset				
	Complete	1	0.99	1
	0-900s	0.67	0.77	0.35
	900-2000s	0	0.81	0
	2000-2500s	0.67	0.78	0.21
On segmented week dataset				
	0-900s	0	0.79	0
	900-2000s	0	0.78	0
	2000-2500s	1	0.81	0.62
Untuned				
	Complete	0.29	0.79	0.30
	0-900s	0.25	0.76	0.39
	900-2000s	0.40	0.79	0.12
	2000-2500s	1	0.79	0.48

Table 6.20: Table showing the TP of the different classes for each dataset

6.8.3 Results model complete dataset

6.8.3.1 Untuned

	precision	recall	f1-score	support
perfect	0.78	0.95	0.85	36720
too early	0.15	0.02	0.04	3902
too late	0.10	0.03	0.04	6734
accuracy			0.74	47356
macro avg	0.34	0.33	0.31	47356
weighted avg	0.63	0.74	0.67	47356

Table 6.21: Classification report untuned

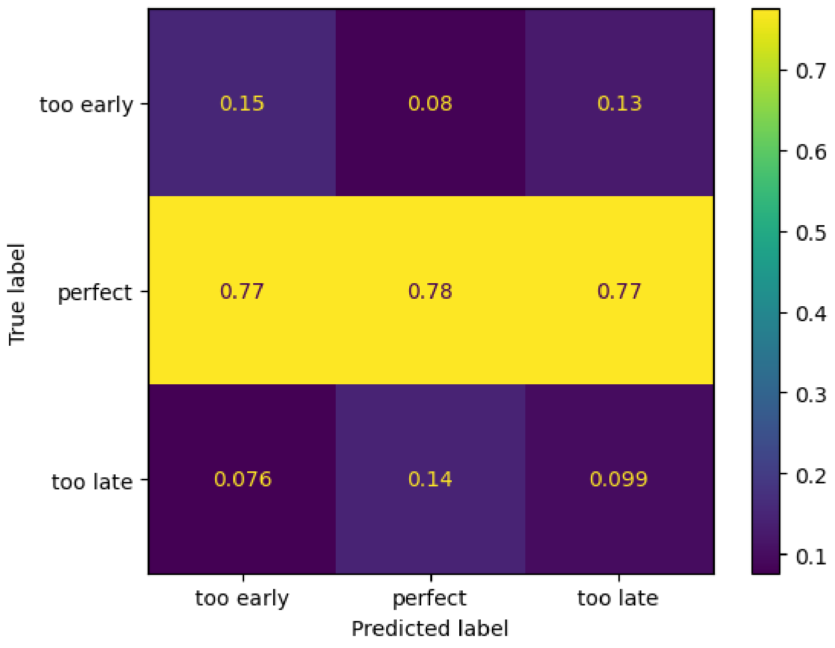


Figure 6.28: Confusion matrix untuned

6.8.3.2 Tuned

	precision	recall	f1-score	support
perfect	0.99	1.00	0.99	36766
too early	1.00	0.93	0.96	3930
too late	1.00	0.98	0.99	6660
accuracy	0.99	47356		
macro avg	0.99	0.97	0.98	47356
weighted avg	0.99	0.99	0.99	47356

Table 6.22: Classification report tuned

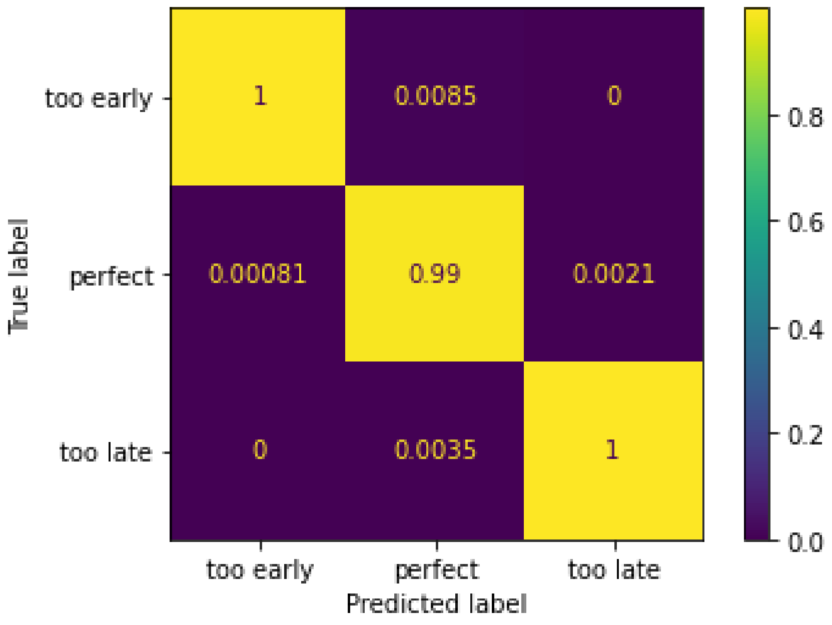


Figure 6.29: Confusion matrix tuned

6.8.4 Results model 0-900s dataset

6.8.4.1 Untuned

	precision	recall	f1-score	support
perfect	0.76	0.99	0.86	19818
too early	0.25	0.01	0.02	1878
too late	0.39	0.01	0.01	4318
accuracy			0.76	26014
macro avg	0.47	0.34	0.30	26014
weighted avg	0.66	0.76	0.66	26014

Table 6.23: Classification report untuned

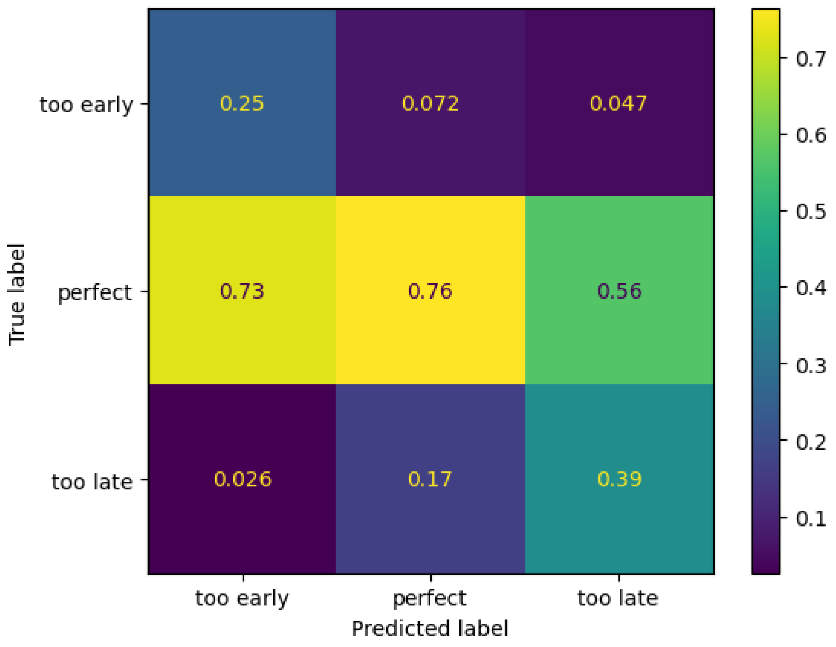


Figure 6.30: Confusion matrix untuned

6.8.4.2 Tuned on complete dataset

	precision	recall	f1-score	support
perfect	0.79	1.00	0.88	1444
too early	0.00	0.00	0.00	126
too late	0.00	0.00	0.00	257
accuracy	0.79	1827		
macro avg	0.26	0.33	0.29	1827
weighted avg	0.62	0.79	0.70	1827

Table 6.24: Classification report tuned

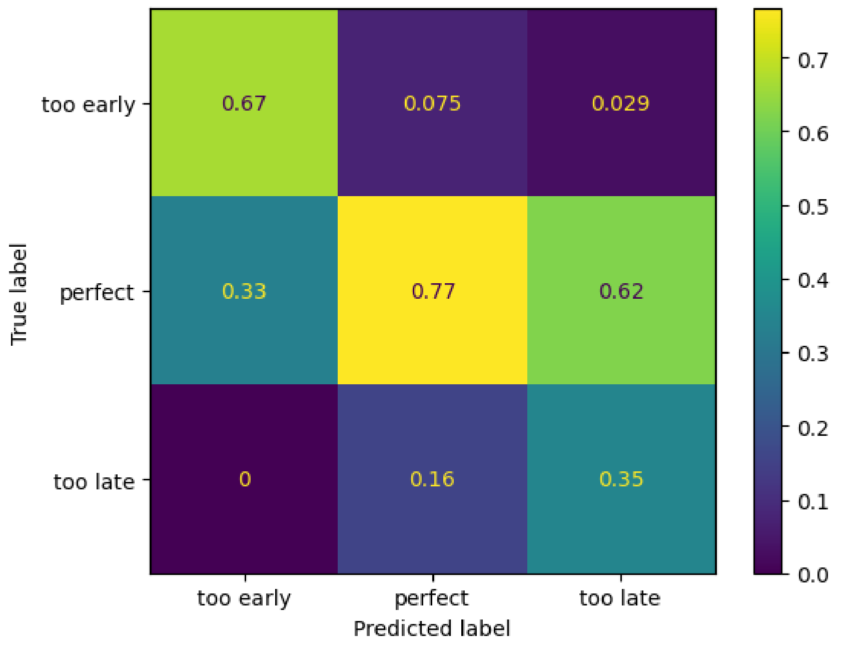


Figure 6.31: Confusion matrix tuned

6.8.4.3 Tuned on segment

	precision	recall	f1-score	support
perfect	0.79	1.00	0.88	1444
too early	0.00	0.00	0.00	123
too late	0.00	0.00	0.00	260
accuracy			0.79	1827
macro avg	0.26	0.33	0.29	1827
weighted avg	0.62	0.79	0.70	1827

Table 6.25: Classification report tuned

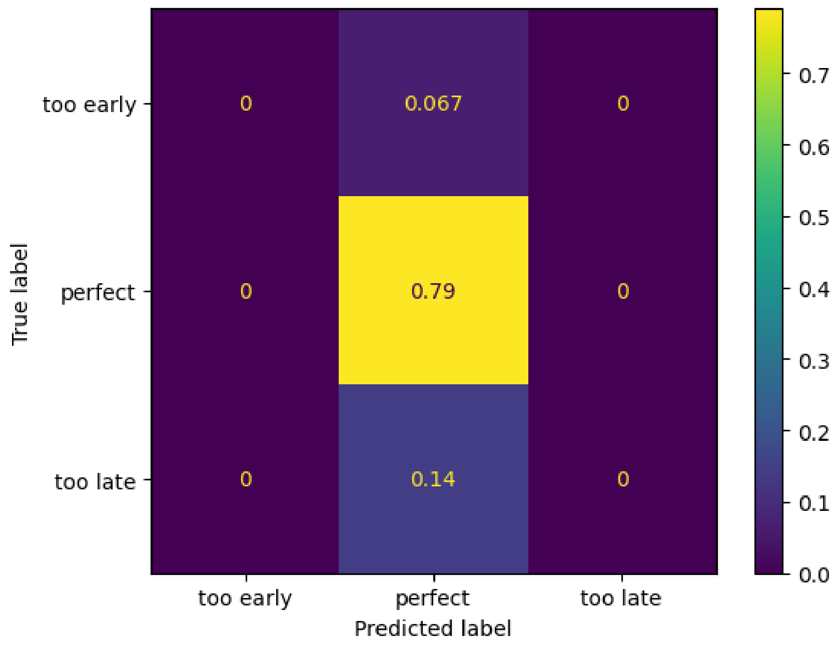


Figure 6.32: Confusion matrix tuned

6.8.5 Results model 900-2000s dataset

6.8.5.1 Untuned

	precision	recall	f1-score	support
perfect	0.79	0.99	0.88	14621
too early	0.40	0.00	0.00	1701
too late	0.12	0.01	0.02	2173
accuracy			0.78	18495
macro avg	0.44	0.33	0.30	18495
weighted avg	0.68	0.78	0.70	18495

Table 6.26: Classification report untuned

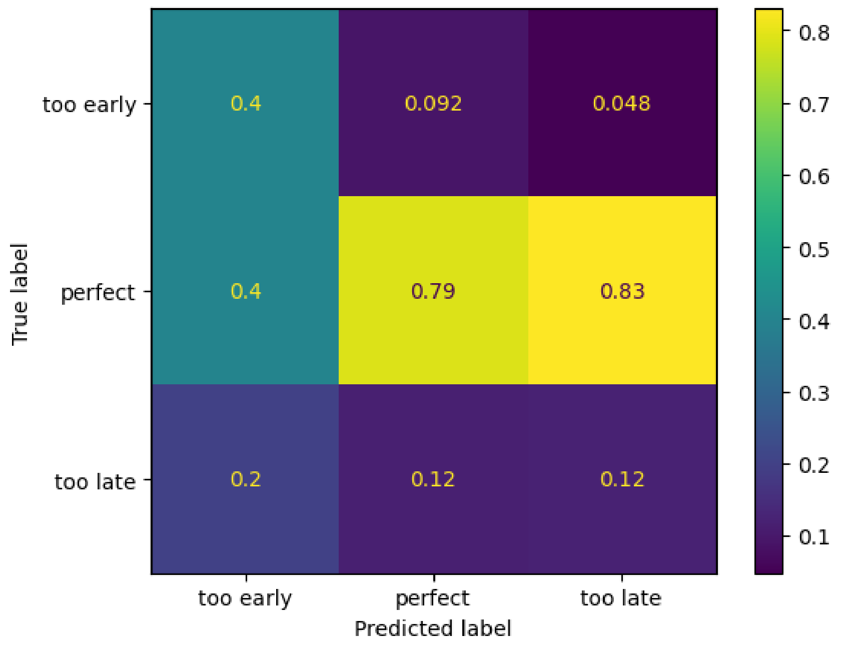


Figure 6.33: Confusion matrix untuned

6.8.5.2 Tuned on complete dataset

	precision	recall	f1-score	support
perfect	0.80	1.00	0.89	782
too early	0.00	0.00	0.00	73
too late	0.00	0.00	0.00	127
accuracy	0.80	982		
macro avg	0.27	0.33	0.30	982
weighted avg	0.63	0.80	0.71	982

Table 6.27: Classification report tuned

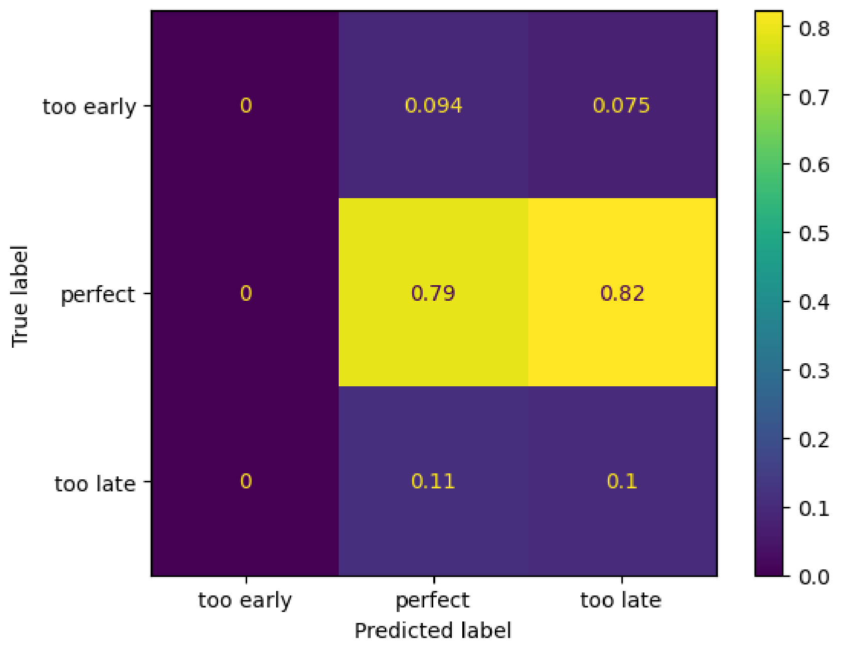


Figure 6.34: Confusion matrix tuned

6.8.5.3 Tuned on segment

	precision	recall	f1-score	support
perfect	0.78	1.00	0.88	767
too early	0.00	0.00	0.00	95
too late	0.00	0.00	0.00	120
accuracy			0.78	982
macro avg	0.26	0.33	0.29	982
weighted avg	0.61	0.78	0.69	982

Table 6.28: Classification report tuned

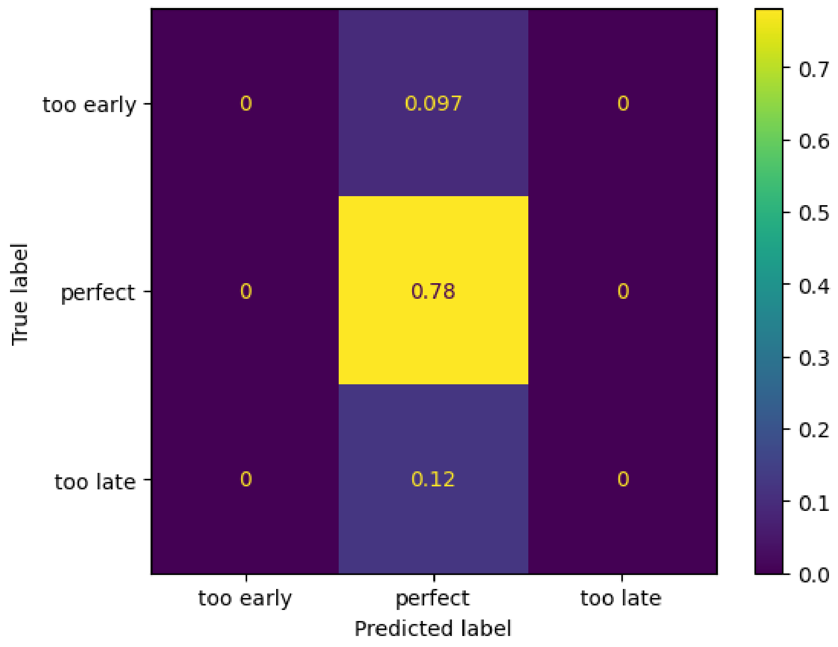


Figure 6.35: Confusion matrix tuned

6.8.6 Results model 2000-2500s dataset

6.8.6.1 Untuned

	precision	recall	f1-score	support
perfect	0.79	0.99	0.88	2252
too early	1.00	0.01	0.02	263
too late	0.48	0.04	0.07	333
accuracy			0.79	2848
macro avg	0.76	0.35	0.32	2848
weighted avg	0.78	0.79	0.71	2848

Table 6.29: Classification report untuned

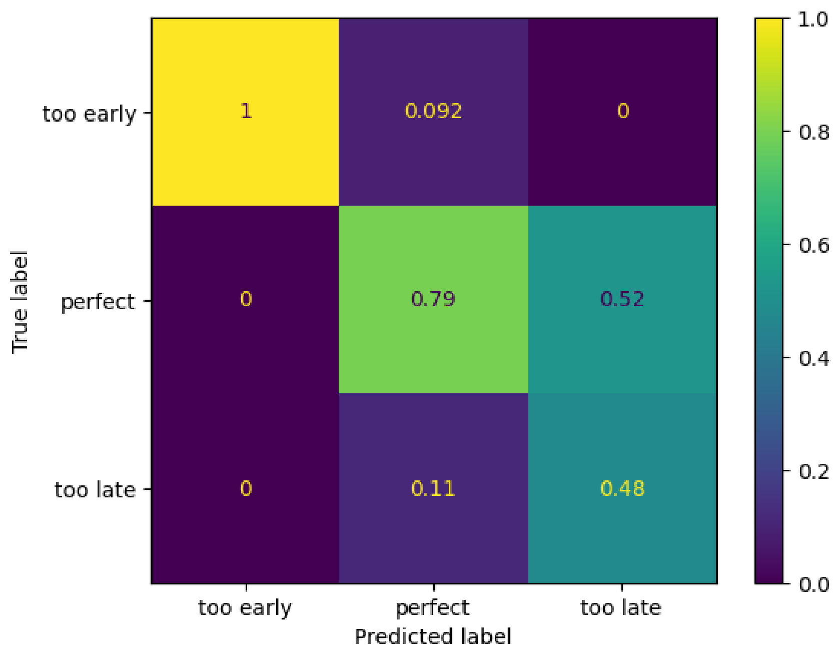


Figure 6.36: Confusion matrix untuned

6.8.6.2 Tuned on complete dataset

	precision	recall	f1-score	support
perfect	0.79	1.00	0.88	114
too early	0.00	0.00	0.00	9
too late	0.00	0.00	0.00	22
accuracy	0.79	145		
macro avg	0.26	0.33	0.29	145
weighted avg	0.62	0.79	0.69	145

Table 6.30: Classification report tuned

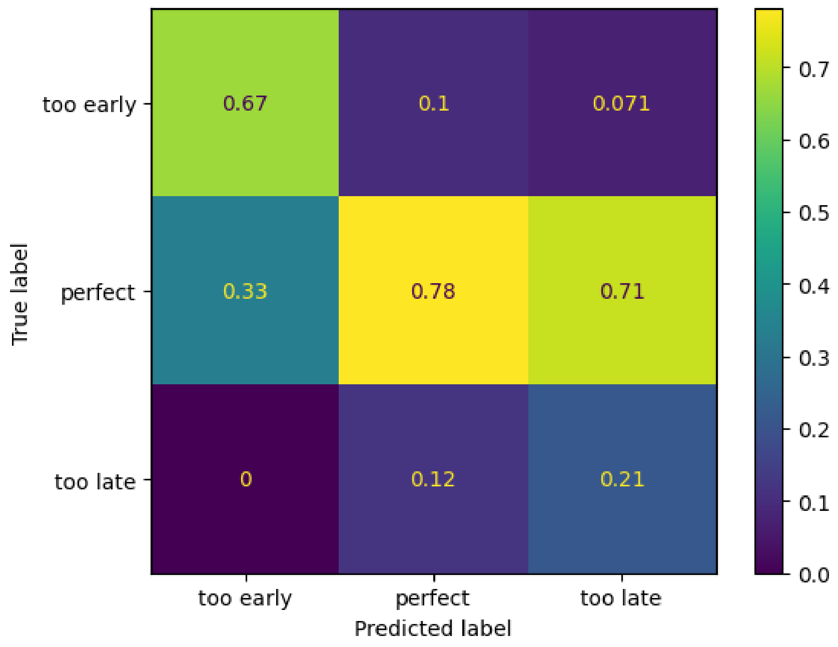


Figure 6.37: Confusion matrix tuned

6.8.6.3 Tuned on segment

	precision	recall	f1-score	support
perfect	0.81	1.00	0.89	2289
too early	1.00	0.02	0.03	252
too late	0.62	0.02	0.03	307
accuracy			0.81	2848
macro avg	0.81	0.34	0.32	2848
weighted avg	0.80	0.81	0.72	2848

Table 6.31: Classification report tuned

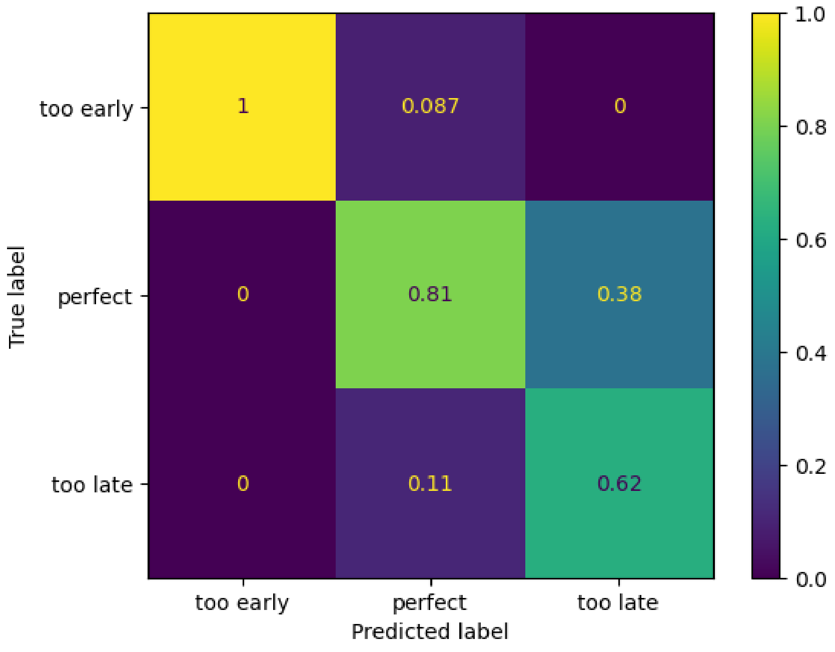


Figure 6.38: Confusion matrix tuned