5 Procedure of Machine Learning model evaluation

As mentioned before, the Machine Learning models were provided by the scikit-learn Python library (cf. Pedregosa et al. 2011). Apart from scikit-learn, other libraries such as numpy (cf. Harris et al. 2020), matplotlib (cf. Hunter 2007), pandas (cf. McKinney 2010) and seaborn (cf. Waskom 2021) were implemented in a Jupyter Notebook environment to conduct the work in this thesis. After importing the libraries, pandas was used to convert the data of the csv files into a data frame to facilitate the declaration of independent and dependent variables. Also, the variable `blueWin` has been selected as the dependent variable. Regarding model training and accuracy computation, a 5-fold cross-validation was implemented, because it is one of the simplest and most popular methods to estimate prediction error (cf. Hastie, Tibshirani, and Friedman 2001, 241). In cross-validation, the dataset is split into K (in this case 5) roughly equal-sized parts. Here, one component will be used as the testing set while the union of the other sets serve as the training set (cf. Hastie, Tibshirani, and Friedman 2001, 242). After computing the accuracy of the model, another component serves as the testing set. Ultimately, the amount of testing sets will be equal to K.

It will be necessary to compute the mean of all accuracies to calculate the accuracy of a model. Let $\kappa : \{1, ..., N\} \mapsto \{1, ...K\}$ be a function that maps an observation to its partition, where N is the size of the dataset (cf. Hastie, Tibshirani, and Friedman 2001, 242).

$$\frac{1}{K}\sum_{k=1}^{K}\left(\frac{1}{\sum_{i=1}^{N} 1_{\{k\}}(\kappa(i))}\cdot\sum_{i=1}^{N} 1_{\{\hat{y}_i\}}(y_i)\cdot 1_{\{k\}}(\kappa(i))\right)$$

The indication function $1_{\{k\}}$ checks if a data point is an element of the testing set while $1_{\{\hat{y}_i\}}$ checks whether an observation is equal to the dependent variable. Additionally, the product of the reciprocal of K and the first sum represents the average of the accuracies.

5.1 Fitting the dataset and training the model

Before applying cross-validation onto the dataset, a model has to be defined through a pipeline to handle the data. The `cross_val_score()` function from the scikit-library takes multiple parameters as input (cf. Pedregosa et al. 2011), including the pipeline, dependent and independent variables, number of splits and the scoring parameter that defines an evaluation metric such as accuracy.

After declaring the input parameters, training the model follows. The table below illustrates the amount of time required for training a model when providing a specific dataset and the size of each dataset. Additionally, a graph has been plotted to visualize the training time duration. The benchmark

was conducted using a AMD Ryzen 7 3700X 8-Core Processor.

ML Model and dataset size	1. Dataset	2. Dataset	3. Dataset
Logistic Regression	0min 9s 176ms	0min 1s 795ms	0min 2s 2ms
Decision Tree	0min 18s 781ms	0min 8s 147ms	0min 8s 502ms
Random Forest	4min 44s 179ms	2min 2s 869ms	2min 10s 67ms
K-Nearest-Neighbor	0min 19s 435ms	0min 7s 159ms	0min 7s 209ms
Naive Bayes	0min 1s 297ms	0min 0s 566ms	0min 0s 592ms
Support Vector Machine	45min 24s 618ms	8min 38s 829ms	8min 59s 51ms
Neural Network	19min 10s 762ms	12min 48s 592ms	13min 14s 243ms
Ada Boost	1min 11s 737ms	0min 32s 860ms	0min 30s 497ms
Gradient Boosting	5min 0s 597ms	2min 10s 238ms	2min 11s 466ms
Amount of games	60 435 games	25 082 games	25 082 games

Table 2: ML Model with accuracy at certain time interval

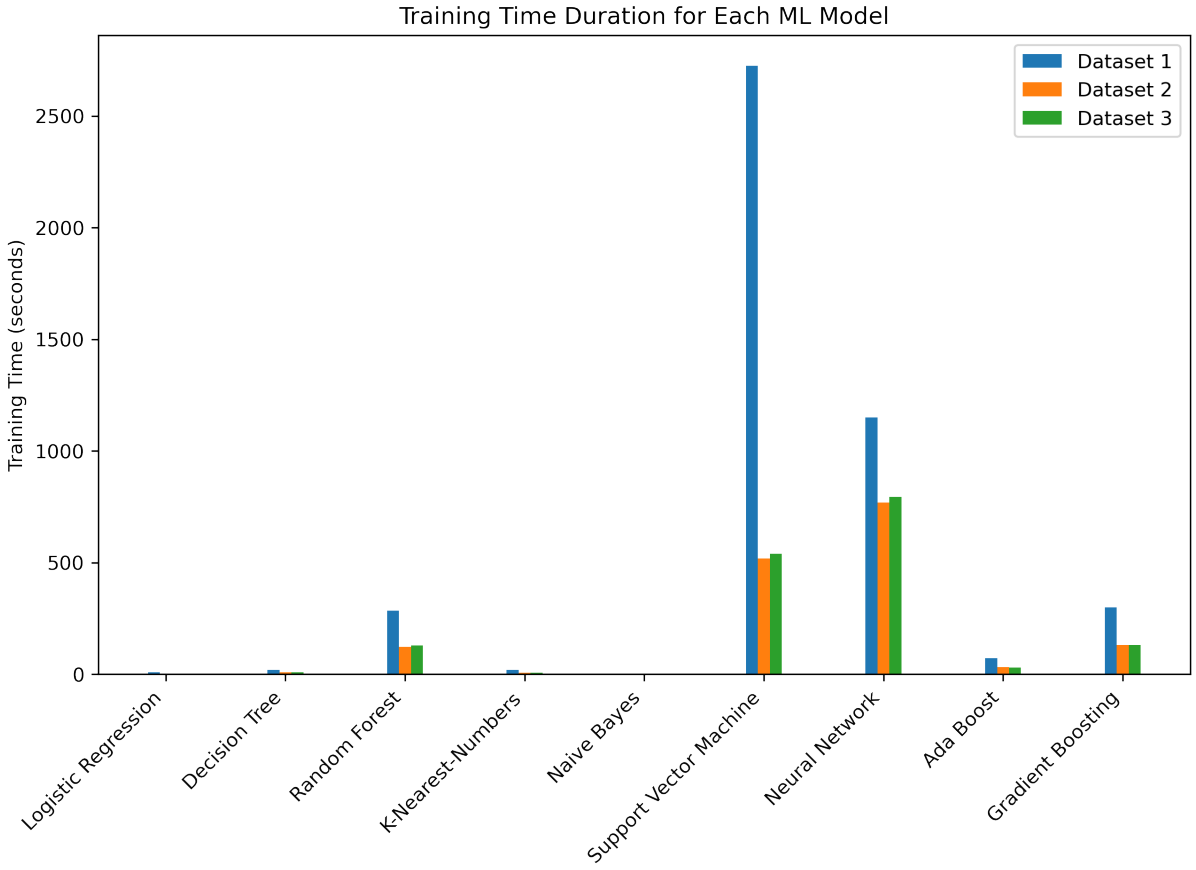


Figure 10: Training time plot

5.2 First time interval dataset accuracies

After providing each model with the first dataset, the following table, containing the accuracies with their corresponding interval and algorithm, has been made:

ML Model	20%	40%	60%	80%	100%
Logistic Regression	64.27%	71.84%	78.69%	87.27%	98.52%
Decision Tree	57.34%	64.50%	71.90%	81.95%	97.54%
Random Forest	63.23%	71.32%	78.50%	87.13%	98.55%
K-Nearest-Neighbor	59.62%	66.63%	73.14%	81.40%	95.09%
Naive Bayes	54.15%	61.13%	70.87%	81.44%	95.33%
Support Vector Machine	64.40%	71.97%	78.78%	87.16%	98.32%
Neural Network	63.30%	70.35%	76.71%	85.32%	98.57%
Ada Boost	63.09%	69.58%	76.37%	85.04%	97.49%
Gradient Boosting	64.00%	71.35%	78.27%	86.73%	98.34%

Table 3: ML Model with accuracy at certain time interval

As can be seen in the table above, the accuracy of every model at the last interval stood out and exceeded the 95% accuracy mark. Additionally, accuracy of each training model increases by time which indicates that the models were functioning, since adding more information about the state of the game lead to a more reliable prediction. After mapping each time mark to its corresponding accuracy, derivation of the visualized graph follows:

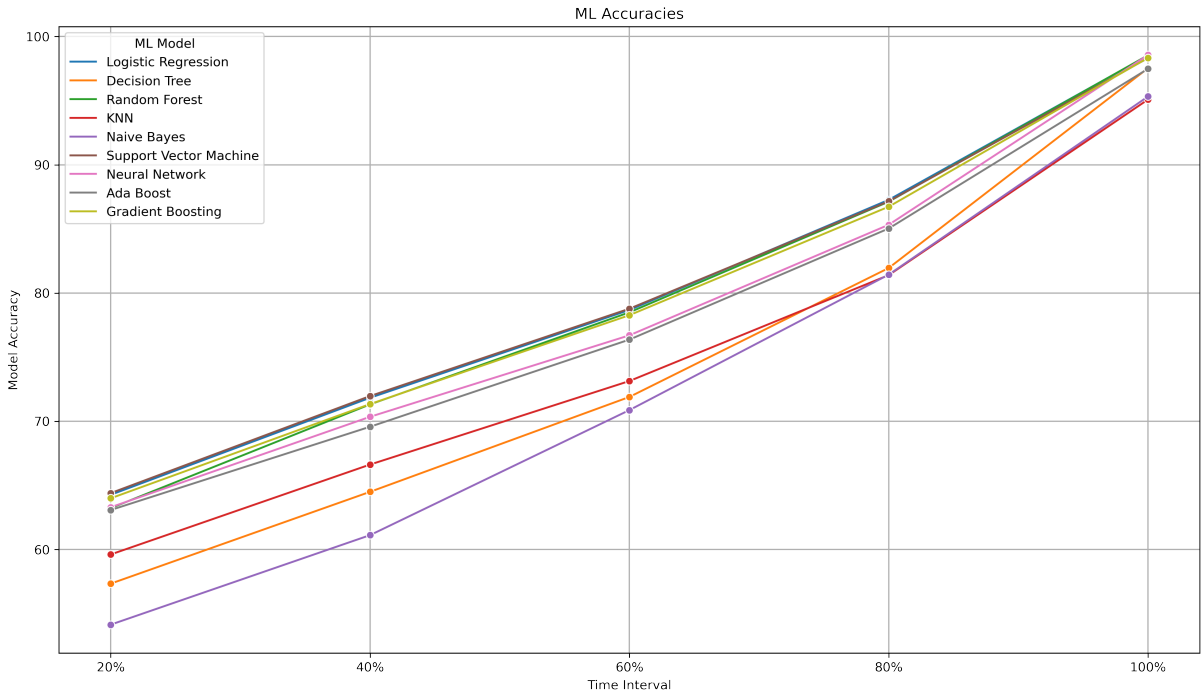
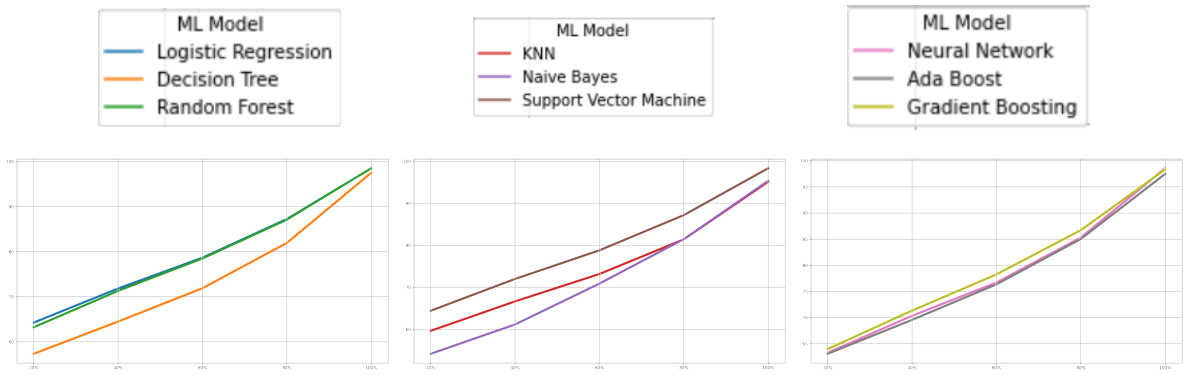


Figure 11: First dataset accuracies



Throughout the whole training process, Logistic Regression and SVM maintained their high accuracy. Gradient Descent also performed well and scored only slightly lower overall. Random Forest, Ada Boost and Neural Network delivered a moderate performance, when only 20% of the game data was provided. However, Neural Network reached an accuracy of 98.57% when trained on 100% of the data and performed best, together with Logistic Regression, Random Forest, SVM and Gradient Boosting models which achieved accuracies of 98.52%, 98.55%, 98.32% and 98.34% respectively. It is essential to acknowledge that the Neural Network only contains two hidden layers, consisting of 64 and 32 nodes respectively, and can therefore be extended to achieve better outcomes. Models which worked worse and started with an accuracy lower than 60 % were KNN, Decision Tree and Naive Bayes. Nevertheless, Decision Tree ended up having similar outcomes as Ada Boost, reaching a score of 97.54%. Although Naive Bayes started of with the worst results, it eventually took over KNN and scored slightly better in terms of accuracy as both models attained 95.33% and 95.09% each.

5.3 Timestamp dataset accuracies

The second table introduces the outcome of the second dataset which contains the accuracies measured at the different timestamps represented by minutes elapsed.

ML Model	min 10	min 14	min 20	min 27	end
Logistic Regression	59.75%	63.02%	69.09%	81.70%	98.53%
Decision Tree	52.50%	54.33%	59.19%	73.02%	96.03%
Random Forest	58.11%	62.04%	68.46%	81.25%	97.78%
K-Nearest-Neighbor	54.25%	56.86%	62.27%	77.41%	94.21%
Naive Bayes	55.95%	59.48%	67.04%	80.34%	94.44%
Support Vector Machine	58.79%	62.77%	68.91%	81.44%	98.19%
Neural Network	54.23%	56.83%	61.99%	75.20%	98.16%
Ada Boost	59.20%	62.24%	69.10%	81.32%	97.69%
Gradient Boosting	59.41%	62.73%	69.16%	81.54%	97.79%

Table 4: ML Model with accuracy at certain minute

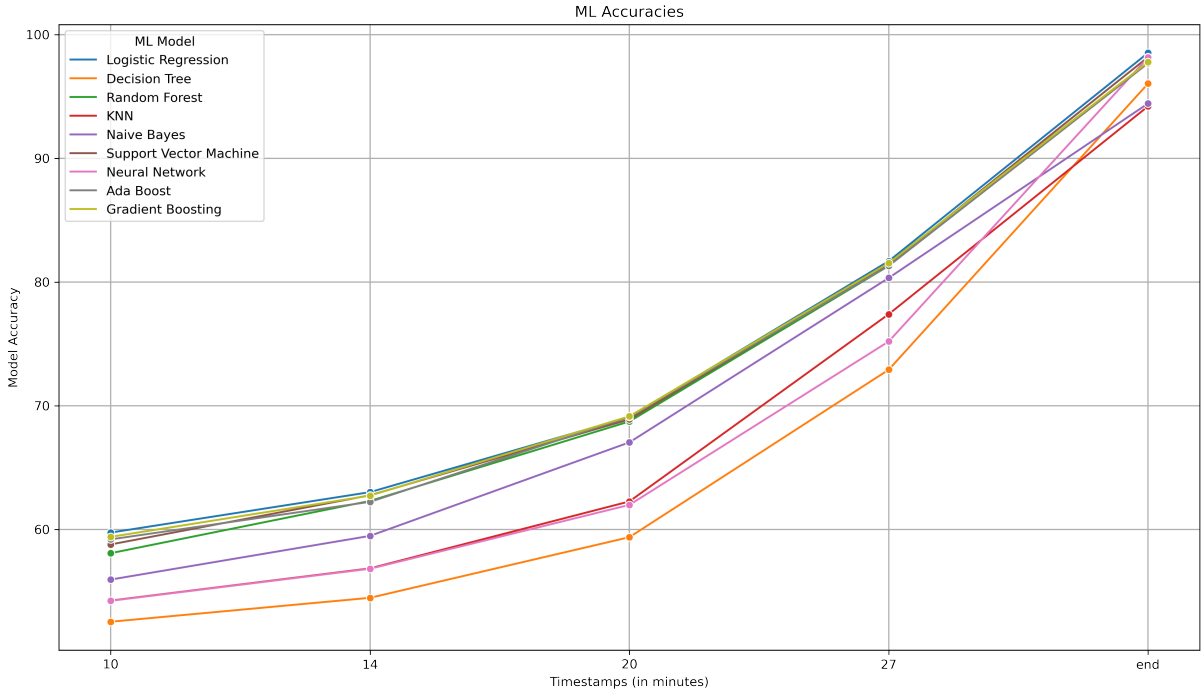
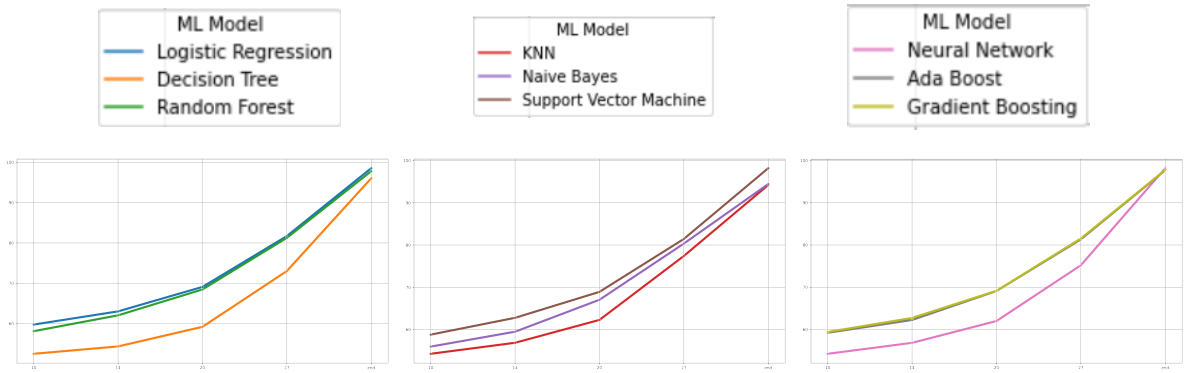


Figure 12: Second dataset accuracies



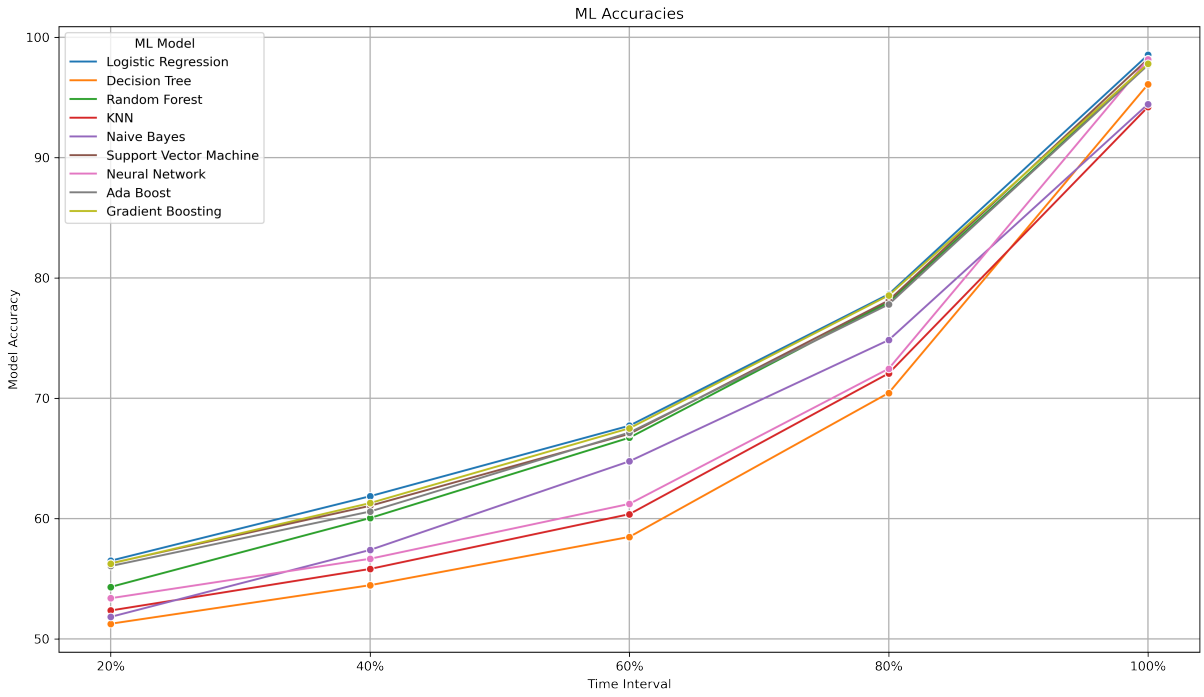
In the second dataset, Logistic Regression still performs best and achieves relatively high scores in comparison to other algorithms. However, all models produce an accuracy lower than 60% at the 10-minute timestamp. In accordance with previous findings, Machine Learning models such as Logistic Regression, Random Forest, SVM and Gradient Boosting outperform other models together with Ada Boost. Also, Naive Bayes' performance rests above Decision Tree, KNN and the Neural Network, but slows down after the 27-minute mark and ultimately converges to an accuracy level comparable to that of KNN. Furthermore, Decision Tree ends up at a higher success rate than KNN and Naive Bayes, after performing worse in the previous minute marks. On top of that, the Neural Network outputs a relatively lower score in the first four time stamps and end up spiking at 98.16%.

5.4 Second time interval dataset accuracies

From the last table, which is based on the dataset containing time intervals of the identical games included in the second dataset, the following results can be deduced:

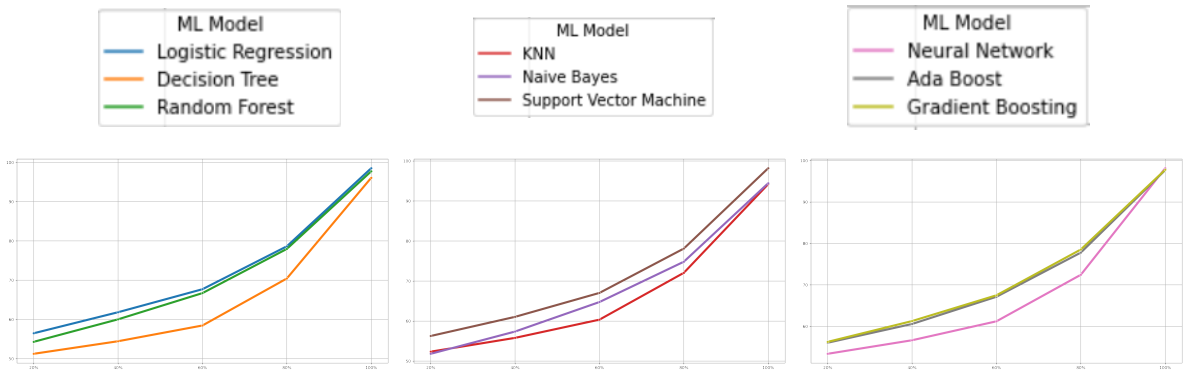
ML Model	20%	40%	60%	80%	100%
Logistic Regression	56.49%	61.85%	67.71%	78.64%	98.53%
Decision Tree	51.26%	54.46%	58.47%	70.44%	96.09%
Random Forest	54.31%	60.04%	66.72%	77.97%	97.74%
K-Nearest-Neighbor	52.36%	55.81%	60.36%	72.06%	94.21%
Naive Bayes	51.83%	57.39%	64.76%	74.83%	94.44%
Support Vector Machine	56.28%	61.06%	67.04%	78.12%	98.19%
Neural Network	53.38%	56.65%	61.22%	72.45%	98.16%
Ada Boost	56.05%	60.58%	67.14%	77.80%	97.69%
Gradient Boosting	56.26%	61.28%	67.50%	78.55%	97.79%

Table 5: ML Model with accuracy at certain time interval



1

Figure 13: Third dataset accuracies



Here, the high accuracy of Logistic Regression, SVM, Ada Boost and Gradient Boosting is still apparent. However, the values which were returned deviate from each other at the 40% time interval, but still come together at the last time interval. While Random Forest achieves scores lower than previously listed models at the 20% time interval, it ends up scoring an accuracy of 97.74%. Also, the models Decision Tree, KNN, Naive Bayes and Neural Network all perform worse than Random Forest. As KNN and Naive Bayes attain a score lower than 95%, both Neural Network and Decision Tree end up achieving an accuracy of 98.16% and 96.09% respectively. One characteristic of the plot which remains apparent is the performance of Naive Bayes at the 40%, 60% and 80% time interval as it exceeds the values of Neural Network, KNN and Decision Tree. Moreover, the Decision Tree model delivers relatively low accuracies, but ultimately overtakes Naive Bayes and KNN.

5.5 General observations

Overall, Logistic Regression, SVM and Gradient Boosting deliver the highest accuracies throughout all datasets while Decision Tree, KNN and Naive Bayes consistently scored below average.

As Neural Network shows a recurring pattern of underperforming at early intervals and significantly improving at later intervals, it does not maintain this behavior in the first dataset. Moreover, the Random Forest classifier always returns accuracies superior to those from the Decision Tree. This result can be deduced from the risk of overfitting (cf. Ibm 2024a) and therefore implies that a Random Forest should almost always be chosen over a Decision Tree, unless memory and computation time are of relevance. Furthermore, out of all models which adhere to a tree structure, Decision Tree performed worst while Gradient Boosting returned the best results in the second and third dataset. Regarding Ada Boost and Random Forest, both models yielded similar results with slight deviations, except in the first dataset, where the performance notably varies. As for the first dataset, Random Forest eventually outperforms Gradient Boosting, while Ada Boosts outcomes fall between Decision Tree and Gradient Boosting.

Accuracies from the 20%/10-minute time point or 40%/14-minute time point also deliver mentionable findings, as the results of Logistic Regression, SVM and Gradient Boosting are always better than the other models (one exception occurs in the second dataset where Ada Boost scores higher than SVM). This indicates that these models would be a better fit for win predictors, when little information about a match is given. The models in the second and third dataset never score better than 60% in the first time point. However, when increasing the number of matches from 25,082 to 64,556, the majority of models exceed 60%.

The observations made from the graphs suggest that it would be more suitable to set the time points to timestamps instead of time intervals, since the second dataset scored higher than the third one although the size of both data collections were identical. Furthermore, a higher score in the first dataset which comprises a larger amount of data indicates that the sample size matters and could elevate the model's performance in terms of accuracy.

6 The Pearson Correlation Coefficient-Based Algorithm (PCCBA)

Akin to training and evaluating Machine Learning models, a custom heuristic that does not rely on Machine Learning was developed, utilizing the Pearson correlation ρ_{xy}. This correlation has been discovered when producing heat maps with the seaborn library. These heat maps contain the correlations between the independent and dependent variables and set the correlation method to 'pearson' as default. Thus, this correlation was selected. The algorithm makes use of the following formula to calculate the probability of the blue team winning:

$$P(X=1) = \sum_{c=1}^{\gamma} \left(\frac{\left(|\rho_{B_c Y}| + |\rho_{R_c Y}| \cdot \frac{1}{2}\right)}{\sum_{d=1}^{\gamma} \left(|\rho_{B_d Y}| + |\rho_{R_d Y}| \cdot \frac{1}{2}\right)} \cdot \frac{b_{tc}+1}{b_{tc}+r_{tc}+2} \right)$$

Here, the index t represents a data point of the testing set, while indices c and d represent the predictor variables. Also, γ is set equivalent to the half of all independent variables, as two variables which correlate with either the blue team or red team can be assigned to the same category. The left term of the sums inner product normalizes the Pearson correlation between an independent variable and the output variable with respect to the sum of all correlations. While variable B represents the independent variables related to the blue team, variable R indicates an association with the red team. The calculation of the Pearson correlation between variables X and Y occurs through dividing their covariance by the product of their respective standard deviations (cf. Berman 2016, 168).

$$\text{Pearson Correlation } (\rho_{xy}): \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2}\sqrt{\sum(y_i - \bar{y})^2}}$$

Additionally, the numerator of the normalization fraction represents the computation of the mean between the correlations absolute values which are in relation with the same independent variable (e.g. blueFirstBlood and redFirstBlood), as red team associated variables tend to suggest a negative correlation towards the blueWin variable when increased. The average of a pairwise correlation represents a portion of the win probability. Since the sum of all averages exceeds the value 1, normalization for each average must be applied.

Conversely, the right term of the sum's inner product determines the amount of the portion which will be assigned the win probability through dividing the value of a blue team correlated independent variable by the total value of the independent variable. Also, both variables b and r will be incremented by 1 to prevent division by zero.

After adding up all percentages which contribute to the chances of the blue team winning, a probability is obtained.

6.1 Results of the *PCCBA*

The subsequent table illustrates the amount of time the algorithm required to process the different datasets. The time and space complexity of the program both go by $O(n)$.

Dataset	**Time Intervals (60 435 games)**	**Timestamps (25 082 games)**	**Time Intervals (25 082 games)**
Time	3min 36s 41ms	1min 30s 289ms	1min 30s 374ms

Table 6: Time required to handle different datasets

Additionally, following table and graph deliver the accuracies which can be deduced from the respective provided datasets that contain either time interval or timestamp information.

Dataset	**20%/10th minute**	**40%/14th minute**	**60%/20th minute**	**80%/27th minute**	**100%/end**
1st dataset	61.68%	68.64%	75.33%	83.76%	93.84%
2nd dataset	59.01%	62.51%	67.84	80.75%	91.25%
3rd dataset	56.20%	60.52%	66.62%	75.45%	91.25%

Table 7: *PCCBA* accuracies

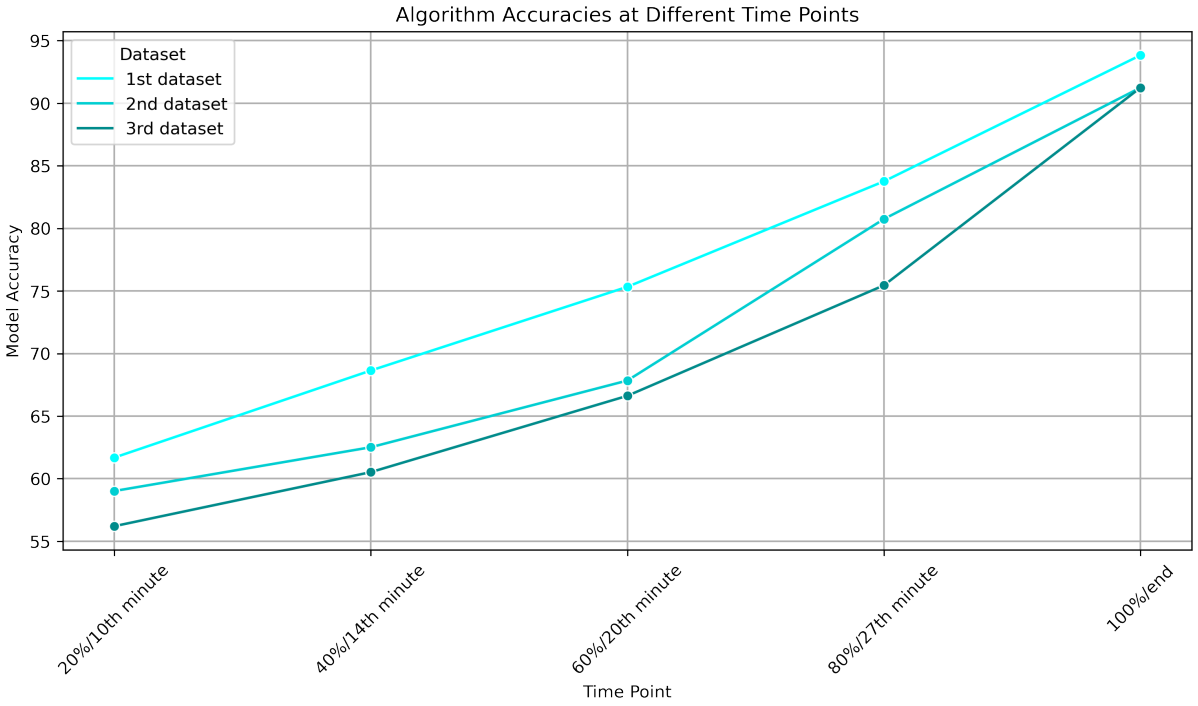


Figure 14: Accuracies of *PCCBA*

Similar to previous findings, the performance of the algorithm, when applying the first dataset was best, while the second dataset shows inferior results, and the third dataset attains the worst scores. Moreover, the line attributed to the first dataset expresses linear behaviour, as the other sets of data receive a surge in terms of accuracy after the 20-minute mark or 80% interval. The continuous increase in accuracy after every time point indicates that the algorithm is suitable for win prediction tasks.

6.2 Comparison with other models

In order to compare the Pearson correlation-based algorithm with other Machine Learning models, following three plots have been made to visualize overall accuracies. As in previous instances, the first plot includes data deduced from the first dataset, while the second plot correlates with the second dataset and the third plot with the third dataset.

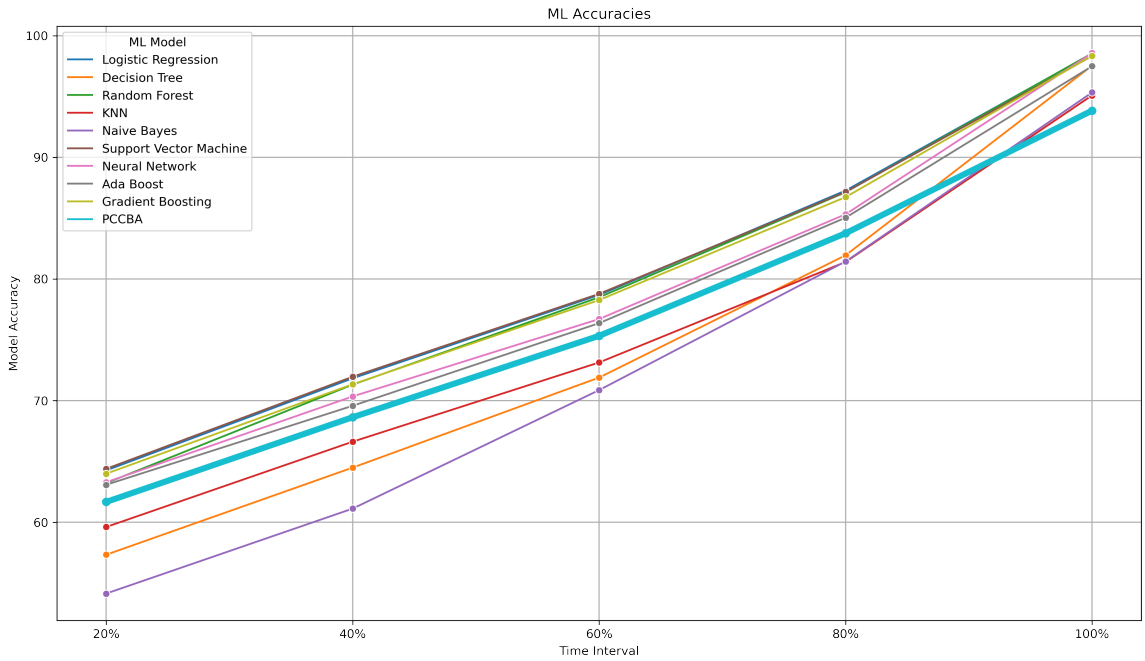


Figure 15: First dataset accuracies with *PCCBA*

In the first dataset the *PCCBA* performs mediocre when compared to other models and ends up scoring lowest at the last time interval. However, the algorithms' accuracies were consistently higher than those of KNN, Decision Tree and Naive Bayes throughout the first four time points.

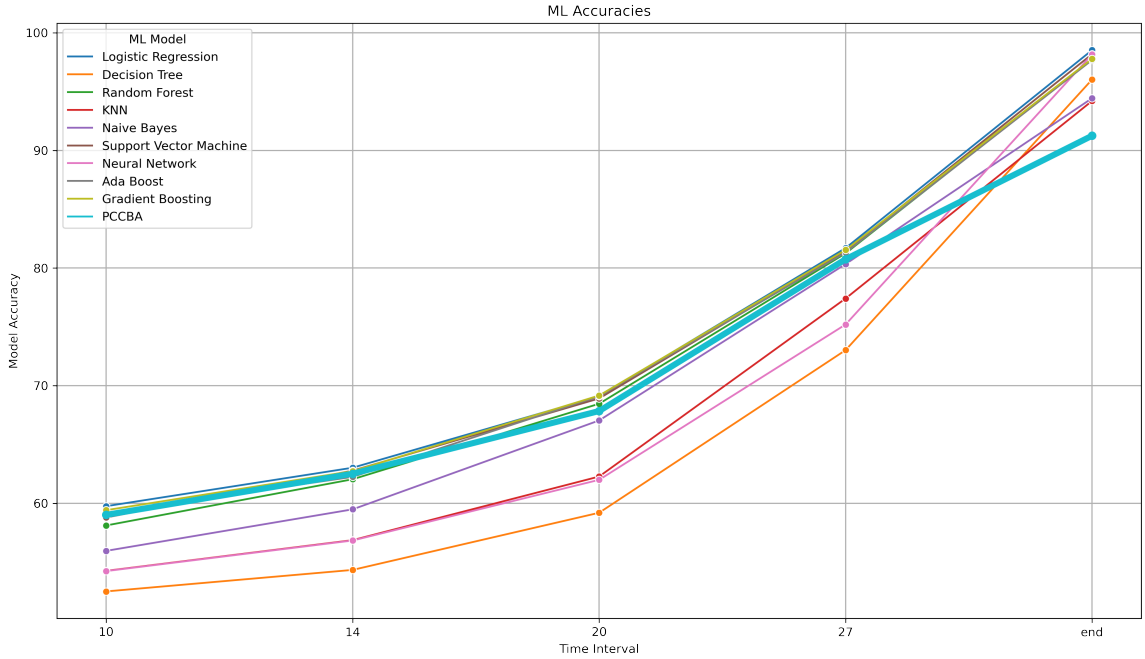


Figure 16: Second dataset accuracies with *PCCBA*

Regarding the second dataset, as the *PCCBA* can keep up with other Machine Learning models that

returned the highest accuracies (e.g. Logistic Regression, Gradient Boosting and SVM), it ends up performing poorly at the last minute mark. In addition to exceeding the capabilities of KNN, Decision Tree and Naive Bayes until the 4th time point it also scored higher than Neural Network in terms of accuracy.

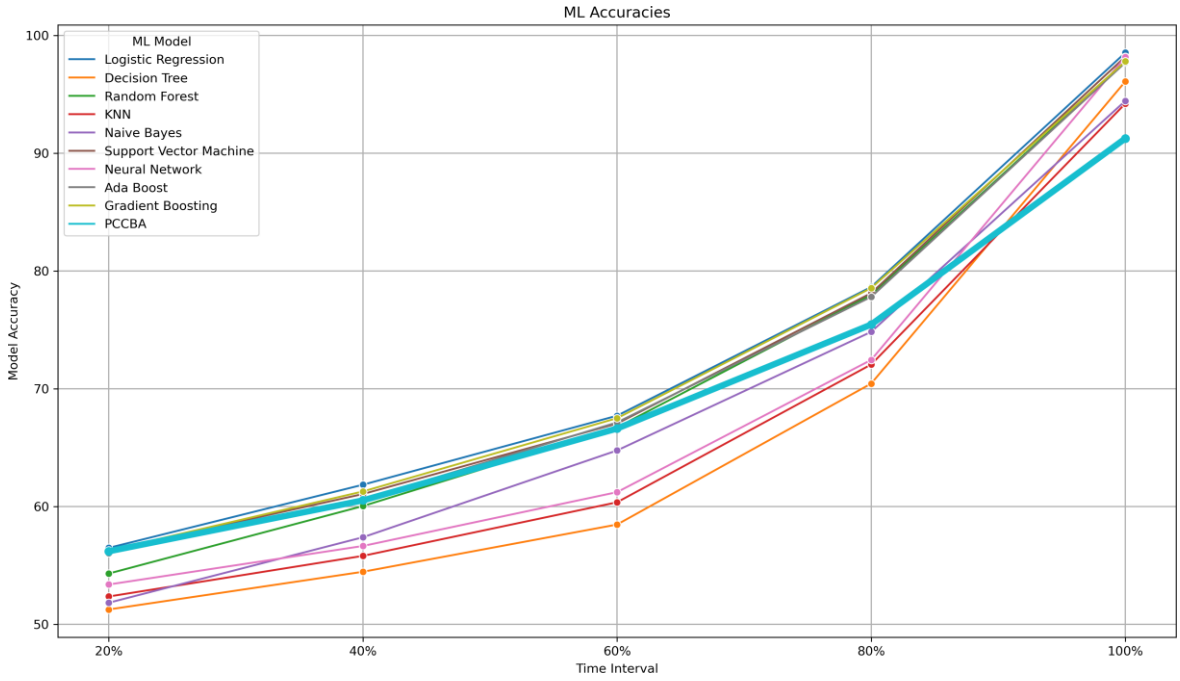


Figure 17: Third dataset accuracies with *PCCBA*

In the third dataset, the *PCCBA* delivers similar results when compared to the second dataset. However, in contrast to the previous dataset, the Pearson correlations' performance already starts to decrement at the 80% time interval.