U.PORTO

Machine Learning & Data Mining

BASKETBALL PLAYOFFS QUALIFICATION

TP1 – G66

Students:

Inês Cardoso	up202005435
Joana Santos	up202006279
Abay Utebaiuly	up202301964

Awards and Prizes (awards_players)

96 awards and prizes received by players.

Coaches Data

163 coaches' data who have managed teams during the specified period.

Player Details

894 player profiles, including player statistics, attributes, and biographical information.

Player-Team Performance

1877 performance data lines of individual players for each team they have played for.

Playoff Series Results

71 results of playoff series, which could include details on matchups, winners, and game outcomes.

Team Performance

143 lines about the performance of teams across different season, involving statistics, rankings, and standings.

Postseason Team Results

81 results of each team during the postseason.

Domain Description

The dataset used in this projects is related to **basketball competitions**, which are divided into two stages: an initial round in which teams fight for victories, and a following playoff with the top-performing teams.

The dataset was compiled over **ten years,** and covers players, teams, coaches, game, and performance measure data.

The primary **goal of this project is predicting next season's playoff-qualifying teams** based on past knowledge acquired by the given dataset.

Business Success Criteria

Prediction Accuracy: The predictive model should achieve a high level of accuracy in forecasting which NBA teams will qualify for the playoffs. Success will be measured by a predefined accuracy threshold, ensuring that the model provides actionable insights.

Profitability: The project should demonstrate a positive impact on team profitability. This includes increased revenue from playoff participation, higher ticket sales, merchandise sales, and enhanced sponsorship opportunities. The success criterion here is a measurable increase in financial metrics tied to playoff qualification.

Player and Coaching Insights: The model should provide valuable insights into player and coaching performance, allowing teams to make informed decisions regarding player recruitment, roster composition, and coaching staff.

Related Business Questions: The answers to related business questions (e.g., player impact, coaching influence) should provide actionable insights that help the organization make strategic decisions and improve overall team performance

Business Understanding

Business Goals

The primary business objective of this project is to **predict with high accuracy which WNBA teams will qualify for the playoffs** in the upcoming season.
This prediction will enable team owners, managers, and sponsors to make informed decisions regarding team management, marketing strategies, and investments. It will also engage and excite fans by providing them with insights into their team's potential success.

Data Mining Goals

1. Gain insights from the data analysis.
2. Make data-driven decisions.
3. Optimize the decision-making process related to basketball playoffs qualification.
4. Working accurate machine model that determine which teams will get to the playoffs

Exploratory Data Analysis

1 - Observation of the datasets

We noticed that there were no repeated data in any dataset.

- **Players Dataset:**
  - Contains information on coaches.
- **Awards Dataset:**
  - Includes coaches' awards.
- **Teams Post Dataset**:
  - A direct correlation is evident in **Figure 2**, where the greater the average minutes played by all teams, the higher the likelihood that a team will make it to the playoffs.
- **Teams Dataset:**
  - New teams are introduced every year.
  - Four teams from each conference qualify for the playoffs.
  - The number of teams in each year is not uniform, as it's possible to see on the graphic from **Figure 1**.
- **Coaches Dataset:**
  - In certain years, there may be more than one coach per team.

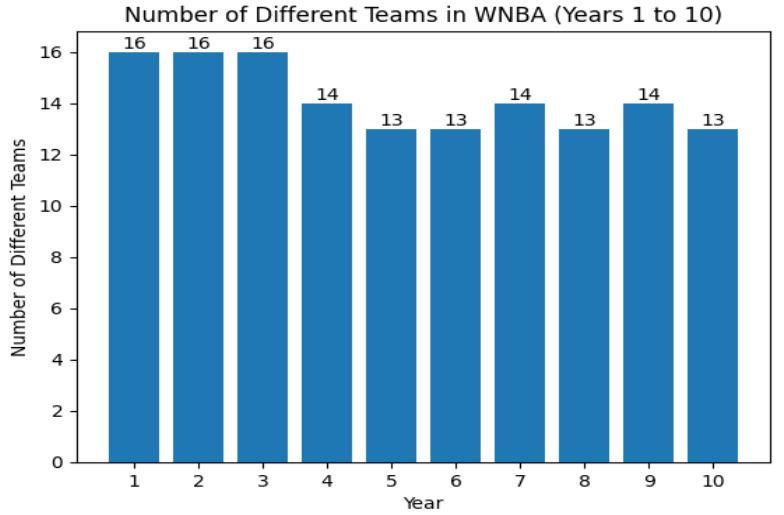


Figure 1 - Number of different teams in WNBA throughout the years

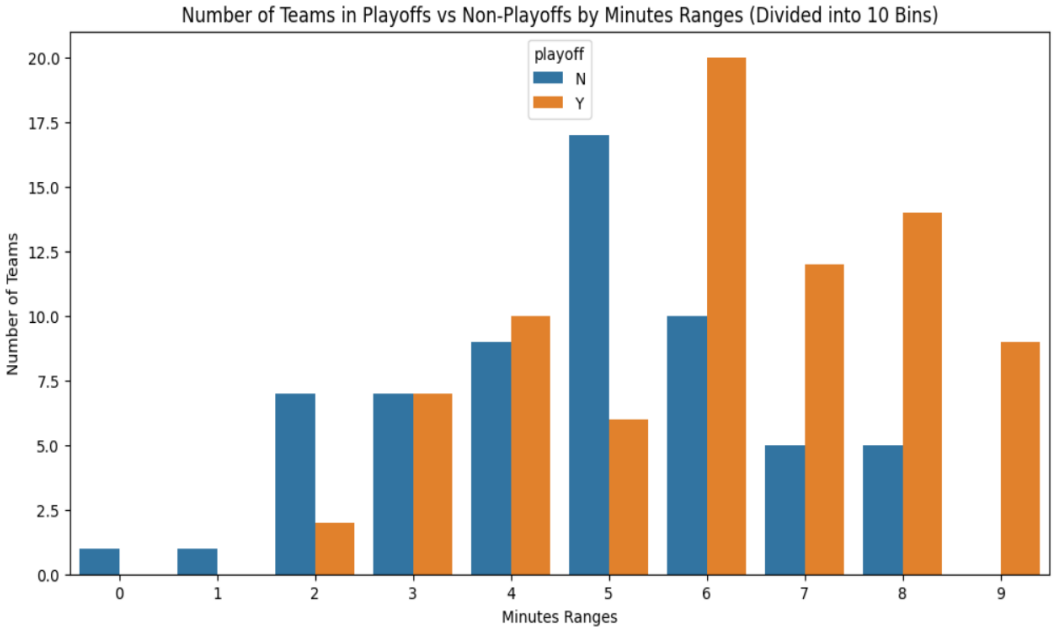


Figure 2 - Number of Teams in Playoffs vs Non-Playoffs by Minutes Ranges

Exploratory Data Analysis

2 - Graphic for outliers and correlation Matrix

For each dataset, a correlation matrix and a graphic to analyze the outliers for each column were made.

- **Correlation Matrices:**

- In the team's dataset "Won" and "Lost" columns can be calculated, respectively, as the sum of "homeW" with "awayW" and "homeL" with "awayL".

- In both **team's dataset** and **players team's dataset**, the rebounds column results from the sum of the oRebounds (offensive rebounds) and dRebounds (defensive rebounds). The same happens in the equivalent post columns

- **Graphic for ouliers:**

- Found **outliers in the weight and height columns of the players dataset.**

- Other outliers found seemed possible for the columns they were in.

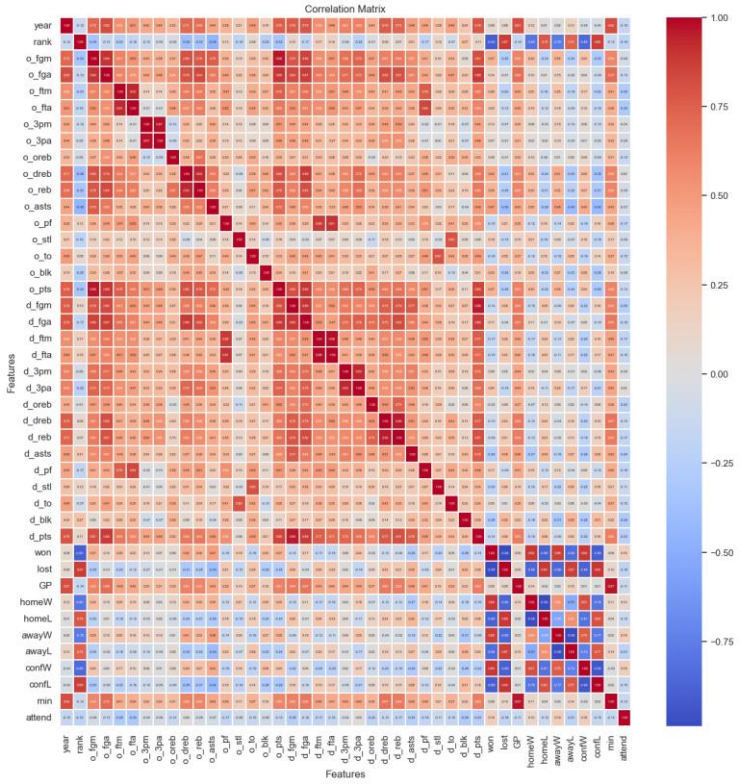


Figure 3 – Correlation matrix for the team's dataset

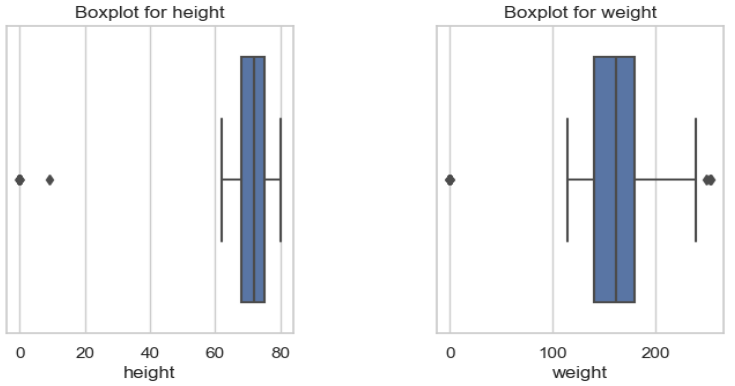


Figure 4 – Graphic for outliers for the player's weight and height

Predictive Data Mining Problem

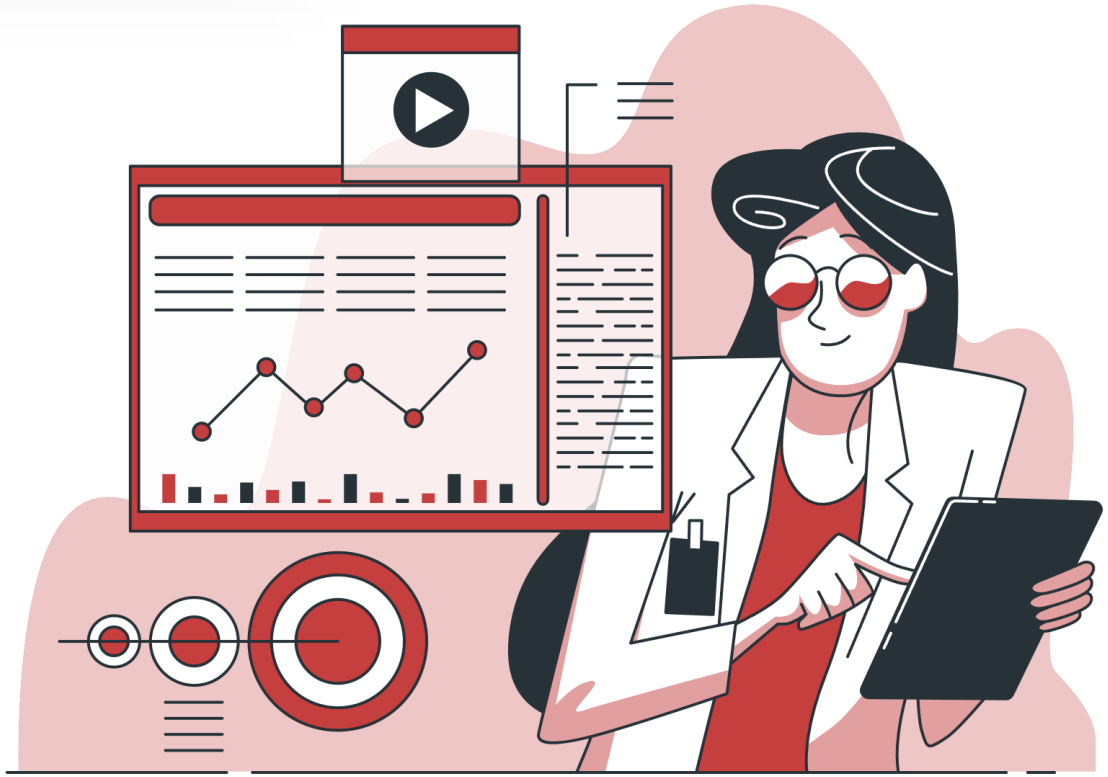
Problem Definition

This is a **supervised classification problem** whose objective is to develop a predictive model that utilizes historical player, team, and game performance data over the past ten years to **forecast the teams that will qualify for the playoffs in the upcoming basketball season**.

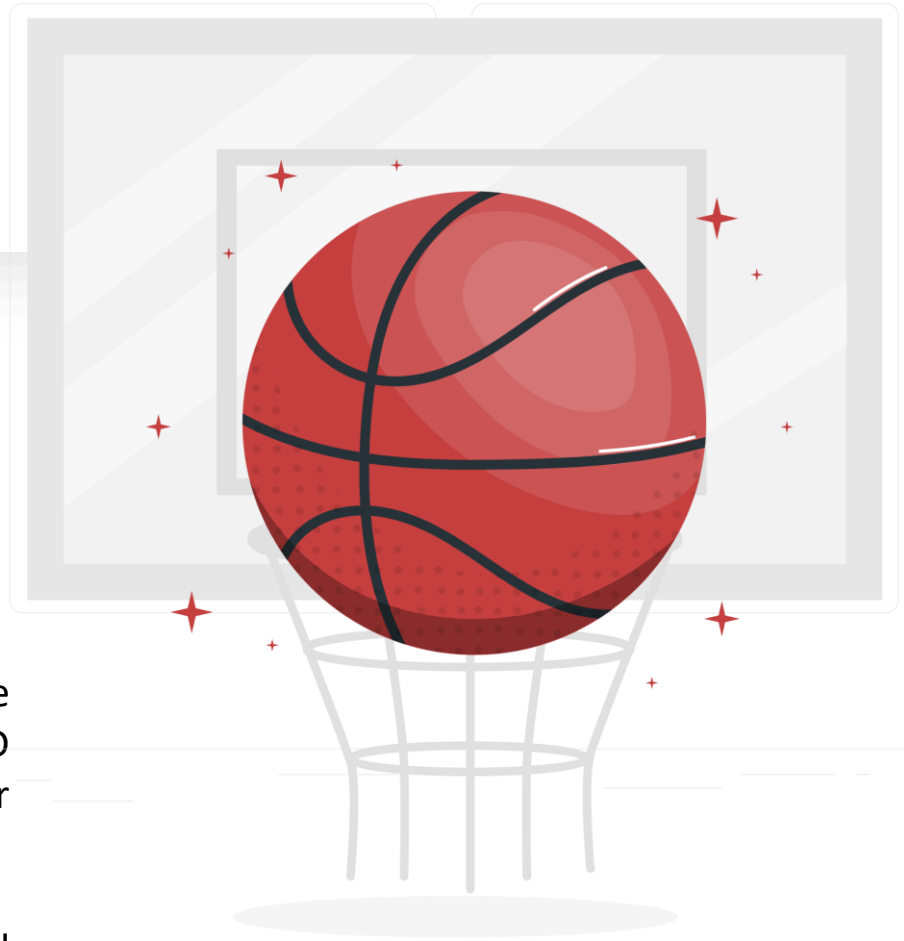
The **model should**:

01 **Consider various performance measures** and factors such as: game outcomes and player and coach statistics to **generate accurate predictions** for the upcoming season's playoff-qualifying teams.

02 Effectively **analyze past patterns** and trends to predict the teams' performance and likelihood of making it to the playoffs.



Data Preparation



1 – Data Cleaning

- **Removed columns with only null values or with just one value.**

- **Removed row entries with only null values or just on value.**
  In the player's dataset, a few entries corresponding to coaches were erroneously present. These ended up being removed since (excluding the ID column) all the others had null values. If a coach becomes a player in a later year, it won't be deleted since it won't have only null values.

- **Removed the outliers** found related to the heights and weights and replaced them with the overall average of all players.

Data Preparation

2 - Data Selection and Data Shift

- **Data Shifting**

The datasets had information related to the end of each year's playoff, that could lead to **data leakage** if used.

Therefore, it was decided to use statistics (sum and average) **from the two years before**.

Team's Post Dataset Preparation:

- Data Shift

Series Post Dataset Preparation:

- Calculated the **number of wins and losses for each team** and year.
- **Data Shift**

Team's Dataset Preparation:

- **Removed** all columns containing **statistics that could be obtainable from the player's dataset**, mainly because new teams lacked initial team statistics.

- Data Shift

Players Teams/ Coaches Preparation:

- **Data Shift -** Computed the **average performance of individuals** in prior years, regardless of the team they were affiliated with.

Awards Dataset Preparation:

- For each player (and coach), we calculated the **number of awards it has received**.
- **Assigned distinct weights** to awards: Excluded the "Kim Perrot Sportsmanship" award and gave double value to those associated with the Most Valuable Player (MVP).
- **Data Shift**

Data Preparation

3 – Data Merge

01

Merged three datasets: the **players' team's dataset**, the **players dataset**, and the **awards dataset**, using the player ID as the key. From the players dataset, we selected only the '**height**' and '**weight**' columns, and from the awards dataset, we extracted the **award count** for each player (and coach).

02

The resulting dataset suffered **aggregation,** by calculating the average of all its columns, grouping them by year and team. This led to the creation of a dataset that contains the **statistical information for all players on a team each years**.
This was then merged with the teams' dataset, by year and team id**.**

03

Additionally, we merged the **coach's dataset** with the **teams' dataset**. In the case of the **coach's dataset**, where multiple coaches were associated with a single team each year, we calculated the average statistics for the coaches of each team and merged this information with the team dataset.

04

We did not merge the teams_post dataset neither the series_post dataset, as their usage led to worse results when testing with models.

Data Preparation

4 – Data Construct (1 / 2)

- **Created a column** named "**performance**" to evaluate **player performance in regular season** based on the following formula:

NEW COLUMN → poins + rebounds + assists + steals + blocks – turnovers – PF

In this formula, positive attributes such as points, rebounds, assists, steals, and blocks are weighted with a positive value, signifying their positive impact on performance, while turnovers and personal faults (PF) are subtracted to account for their negative influence.

- **Created a column** named "**playoff_performance**", that employs the identical formula as "performance" column but uses playoff data for calculation, determining the **player performance in playoff season**.

- **Deleted the columns "homeW", "awayW", "homeL" and "awayL",** since the sum of the first two equals the "Won" column and the sum of the last two equals the value in "Lost"

- **Deleted the columns oRebounds (offensive rebounds) and dRebounds** (defensive rebounds), since their sum correspondend to the column Rebounds. The same happened for the equivalent post columns.

Data Preparation

4 – Data Construct (2 / 2)

- **Created a column** named "**points_precision**" that calculates the precision of the points for each player in regular season, with sequential deletion of the columns used for the creation of it.

points_made = ftMade + 2 * (fgMade – threeMade) + 3 * (threeMade)

points_attempled = ftAttempted + 2 * (fgAttempted – threeAttempted) + 3 * (threeAttempted)

NEW COLUMN → **points_precision** = points_attempted – points_made

- **Created a column** named "post_points_precision", that employs the same logic as for "points_precision" column but uses playoff data for calculation, determining the precision of the points in playoff season.

- Created 7 new columns: One with the **sum of the number of times that a team in the two previous years went to playoffs,** other 6 with the **sum of the wins and the losses of the different steps of the playoff**, the first round, semis and finals

Experimental Setup

01

Standarizing

Standardizing the data using **MinMax Scaller** from sklearn library

02

Feature Selection

Feature Selection with Manual and also using two different feature selection algorithms :

- **SelectKBest** function from the sklearn library, with ***mutual_info_classif*** as the score function, to select the best features for each model. To determine the optimal number of features for each model, we plotted a graph showing the change in average of the accuracy through the years, with respect to the number of features. Based on this graph, we selected the best number of features for each model.

- **Backwards selection** using SFS with features between (3, 32) with scaled data for the algorithms that needed it.

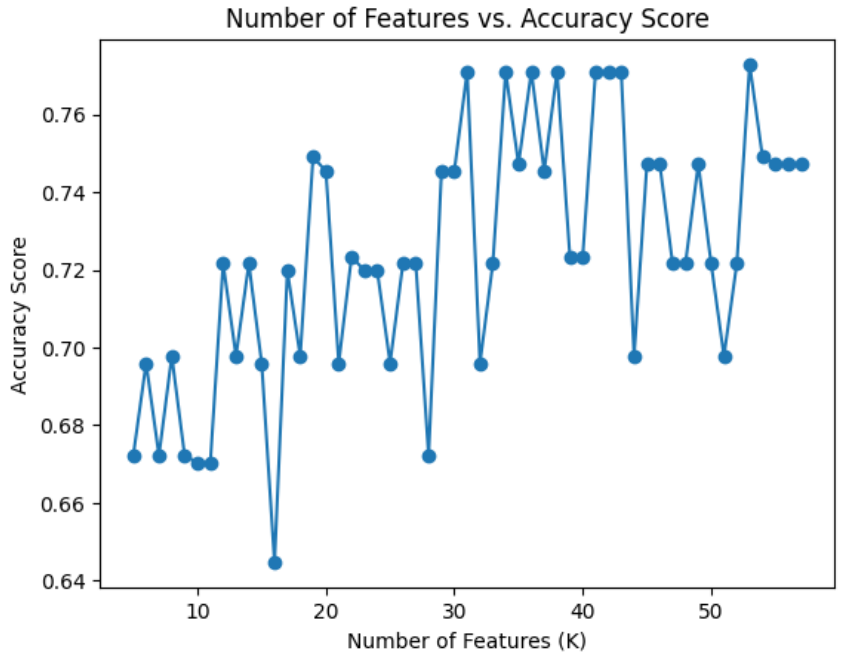


Figure 5 – Variance of the accuracy score with the variance of the number of features or the Voting Machine Classifier

Experimental Setup

03 Hyperparameter Tunning

Hyperparameters tunning with **OUR OWN** implementation of **GridSearch** and **RandomizedGridSearch**

04 Test with different algorithms & Bagging

To split the data for training and testing, we picked a year to test in and used 'n' years back to train.

- To determine the optimal number of years to look back, we plotted a graph showing the average accuracy – calculated from the accuracy training of years 5 to 10 - changing the number of years used for training. Additionally, we created separate graphs for the training and testing data to identify the year that would provide the highest accuracy without overfitting.
- We then calculated the prediction by picking **4 teams from each conference** with the highest probability of going to the playoffs in each one.
- We tried to **test with different algorithms and different combinations of feature selection** to try to achieve the best results.
- **For training and testing purposes, we did not use years one and two as we shifted the data two years back.**
- To enhance the stability of our models, we also tried to implement **the bagging method**. We also thought about other methods, such as boosting. However, bagging seemed most suitable for our problem.

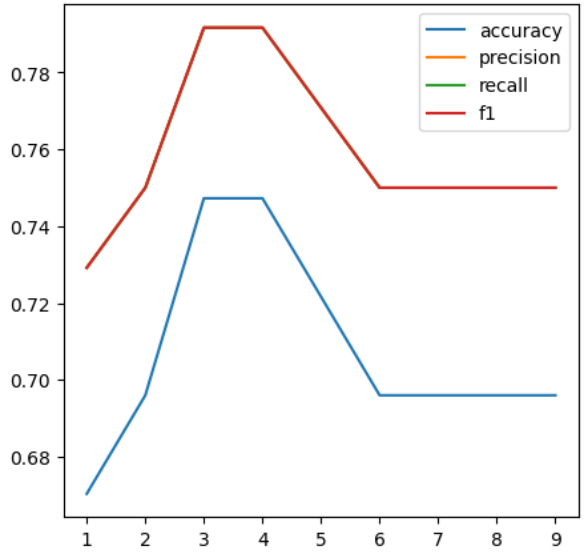


Figure 6 –Average accuracy per years training data for the Voting Machine Classifier
For example: The value of accuracy in "Year 9" means that is the accuracy of using the data from the years before year 9

Experimental Setup

05

Analyze the results

The results were analysed with graphs of the variation in accuracy, performance, recall, and f1 over the years, for different algorithms and combinations. In addition, the confusion matrix and ROC curve for the 10th year were evaluated.

06

Ensemble

After analysing the results, we decided to ensemble the results by creating a **voting system** with the different algorithms

In this step, various combinations were tried to achieve the best results possible.

Results

- We created a **voting mechanism** to **ensemble the results from the four best** models with hyperparameter tunning: KNN, Logistic Regression, MLP Classifier and Naïve Bayes. The outcomes from the voting machine are illustrated in Figure 5, where **the best result is represented by the Green Line, which is using the Kbest algorithm training with the data from the previous 4 years.**
- This voting machine was also used to predict the results for the 11th year, in the Kaggle competition.
- Detailed explanations of the utilization of each of these models, along with the corresponding results, can be found in the annexes of this document.

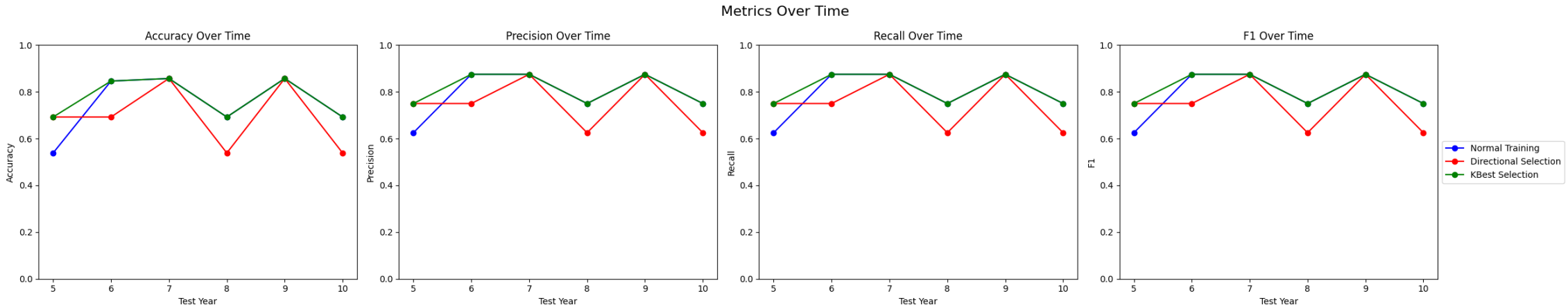


Figure 7 – Ccomparison of the training and testing results of the Voting Machine Classifier under three conditions: without any feature selection algorithm, with backward selection, and with Kbest selection.

BEST RESULT
(4 YEARS BACK AND KBEST)

All years:	Accuracy: 0.77	Precision: 0.81	Recall: 0.81	F1: 0.81
10th year :	Accuracy: 0.69	Precision: 0.75	Recall: 0.75	F1: 0.75

- In Figures 6 and 7, we examine both the **Learning Curve (Training Curve)** and the **Testing Curve** under two scenarios: one utilizing the **Kbest algorithm** and the other **without any feature selection algorithm**. The learning curve serves as a crucial tool for identifying overfitting and underfitting. Notably, our analysis of Figures 7 and 8 reveals the **absence of both overfitting and underfitting** in the depicted curves.

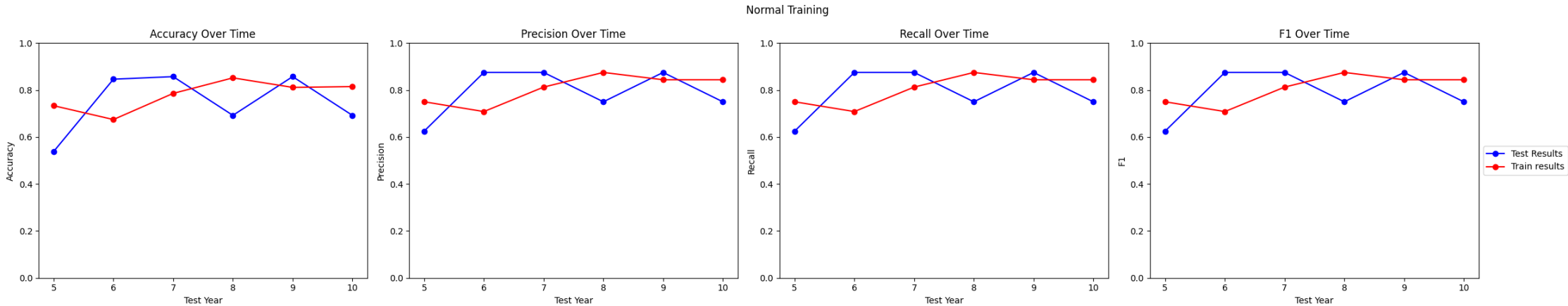


Figure 8 – Training and Testing curves for the Voting Machine Classifier with no feature selection algorithm

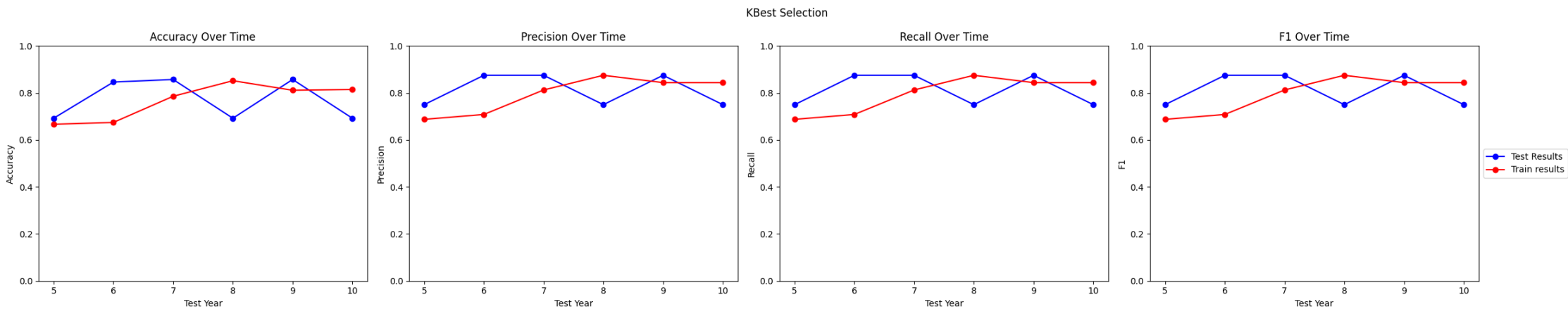


Figure 9 – Training and Testing curves for the Voting Machine Classifier with the Kbest algorithm

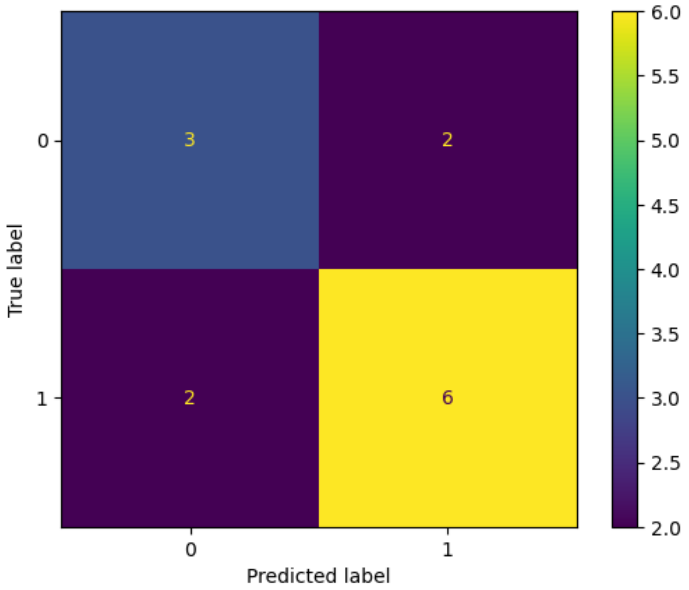


Figure 10 – Confusion Matrix for year 10 of the voting machine with hypertunning and kbest

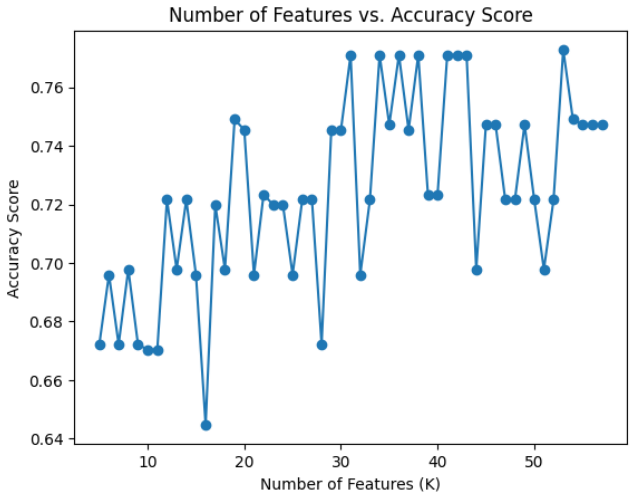


Figure 12 – Variance of the accuracy score with the variance of the number of features for the Voting Machine Classifier

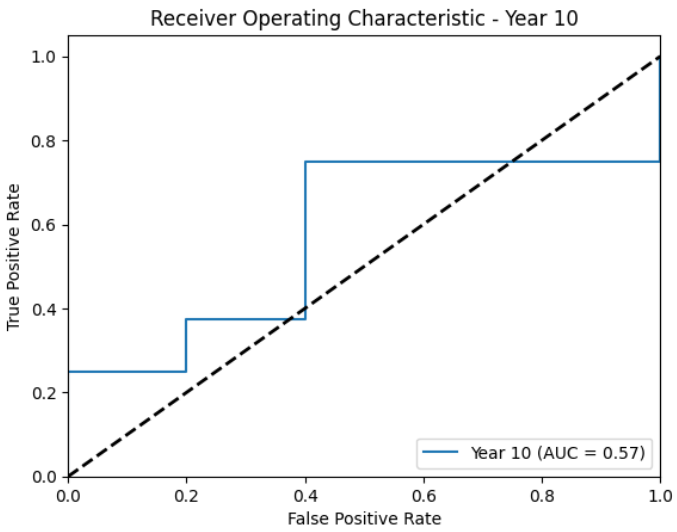


Figure 11 – ROC curve for year 10 of the voting machine with hypertunning and kbest

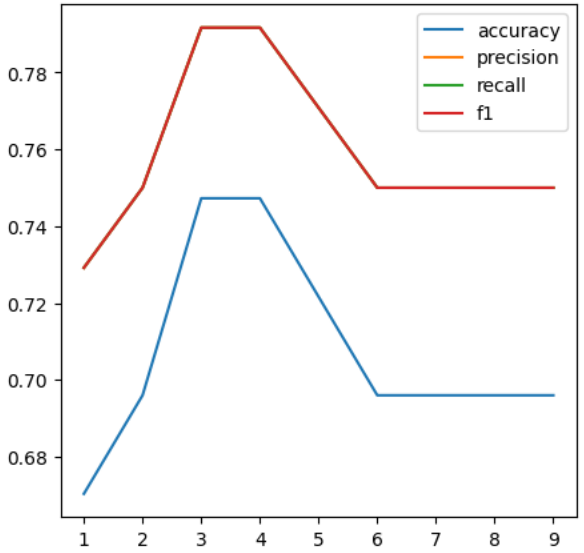


Figure 13 –Average accuracy per years training data for the Voting Machine Classifier

It is possible to deduce that the Ensemble Machine algorithm delivers **stable and high-quality results** by examining the presented graphs on the current and previous slides.

After analysing the graphs on the previous slide, it is clear that the accuracy of the testing with hypertuning ranged from 0.69 to 0.85, indicating a small fluctuation. **The most favourable outcomes were achieved when we used a combination of both kbest and hyper-tuning.** The learning curves also demonstrate a steady improvement in accuracy over time.

Further analysis of Figure 12's graph establishes that this algorithm attains its **optimal performance with approximately 30 features**.

The graph in Figure 13 reveals a distinct pattern: after the algorithm reaches its peak performance when training with 3 and 4 years back. However, the performance drops after that point. This indicates that **utilizing data from extremely old seasons results in poorer outcomes.**

Conclusions

Conclusions

- We **explored various ML models**, and tried **diverse combinations** of years, features, and datasets to **optimize our results**.
- Feature selection improved results in certain models but had a detrimental effect on others:
  - For instance, when employing the "Select K Best" method, we observed that the **number of features chosen might not be** universally **optimal for all algorithms**.
  - In the case of the "Backwards" algorithm, even with cross-validation set to 0, it continues to **employ a simplistic test-train split methodology that differs from our** preferred data testing and training approach.
- Additionally, we found that hyperparameter tuning yielded improvements in the results of certain algorithms, but it presented similar challenges to the "Backwards" algorithm in terms of methodology misalignment.
- Ensemble various models allowed to improve the results.

Annexes

Tools used

Tools

- Utilized **Python** for data analysis and modeling, using also the libraries: **pandas, numpy, scikit-learn.**
- Utilized data visualization tools (e.g., **Matplotlib**, **Seaborn**) for exploratory data analysis and model interpretation
- **Jupyter Notebook** and **Google Colab** as the primary platforms for coding and analysis.
- Utilized **Github Issues** for project management.

Data Preparation

1 - Data Selection and Data Shift

- **Data Shifting**

Three distinct **data-shifting strategies** were **separately applied** to ALL the datasets.

- One approach involved using the **previous year's team results.**
- The other calculated the **two-year mean performance**.
- The last calculated is the **three-year mean performance**.

In the table below, we can see the results from the first pipeline. As we can see, the best results were obtained when using two years.

	1 year back approach				2 years back approach (with hypertunning)				3 years back approach (with hypertunning)			
	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
KNN	0.65	0.71	0.71	0.71	**0.67**	**0.73**	**0.73**	0.73	0.70	0.75	0.75	0.75
Naive	0.75	0.79	0.79	0.79	**0.75**	**0.79**	**0.79**	0.79	0.70	0.75	0.75	0.75
Logistic Regression	0.75	0.79	0.79	0.79	**0.77**	**0.81**	**0.81**	0.81	0.70	0.75	0.75	0.75
MPL classifier	0.65	0.71	0.71	0.71	**0.75**	**0.79**	**0.79**	0.79	0.72	0.77	0.77	0.77

Ideas explored

1 – Trained models with conferences together and separated

Our first approach was to separate the conferences into two datasets, train with each of the datasets, join the predictions and then calculate the accuracy.

However, the problem of overfitting due to having less data to train appeared. To fix this problem we trained with both conferences and then picked four from each conference to go to the playoffs.

It's difficult to make a general statement about whether the results improved or worsened after this change, as it depends on the specific algorithm used. However, we did observe that some training curves that previously showed signs of overfitting were no longer exhibiting this issue, as we can see in figure 9 and 10.

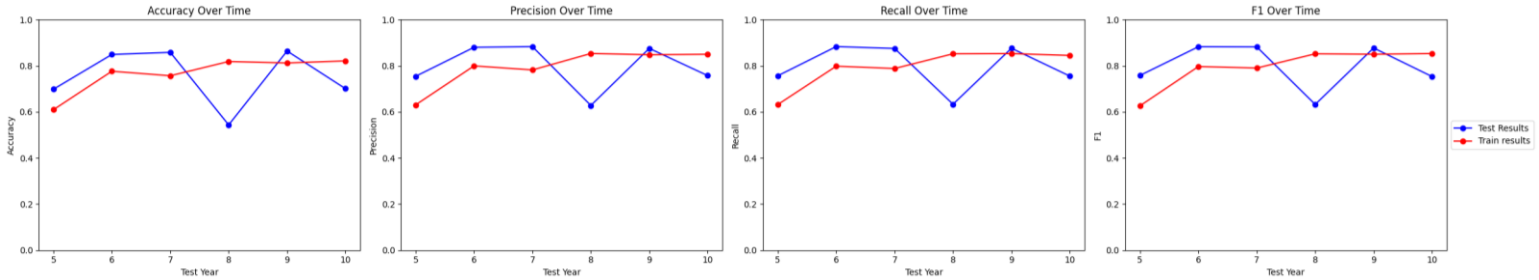


Figure 14 – MPL trained with conferences together

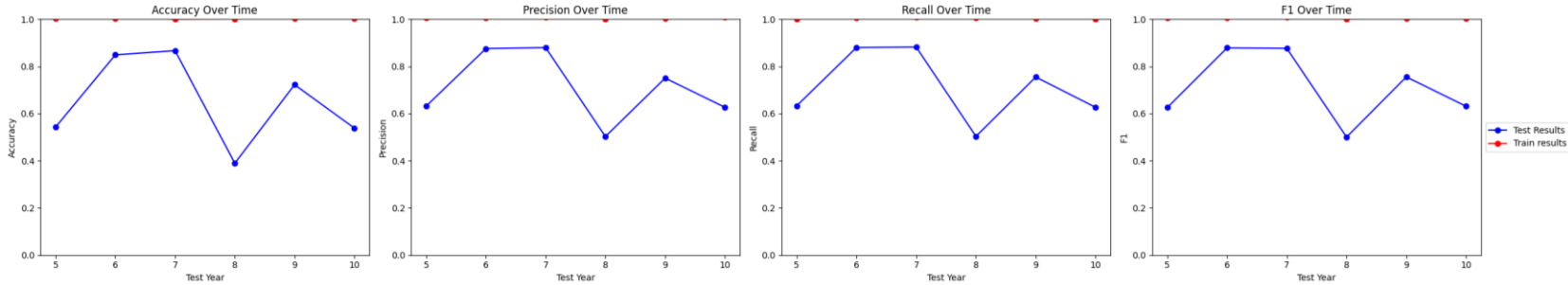


Figure 15 – MLP Trained with conferences separate

Ideas explored

2 – Bagging

We experimented with bagging in all our algorithms, with and without hyper tuning. However, this did not yield any significant improvement and, in some cases, resulted in worse outcomes. Therefore, we decided against it.

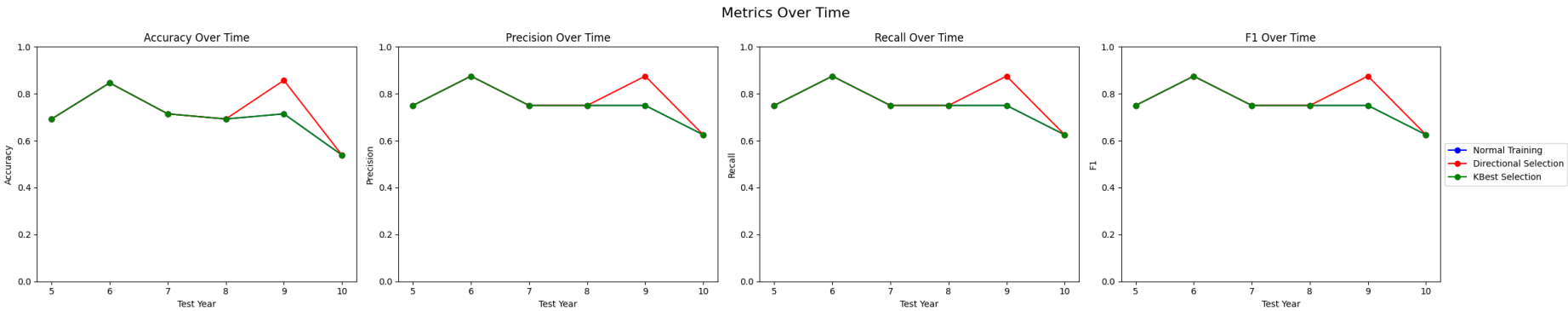


Figure16 – KNN without bagging

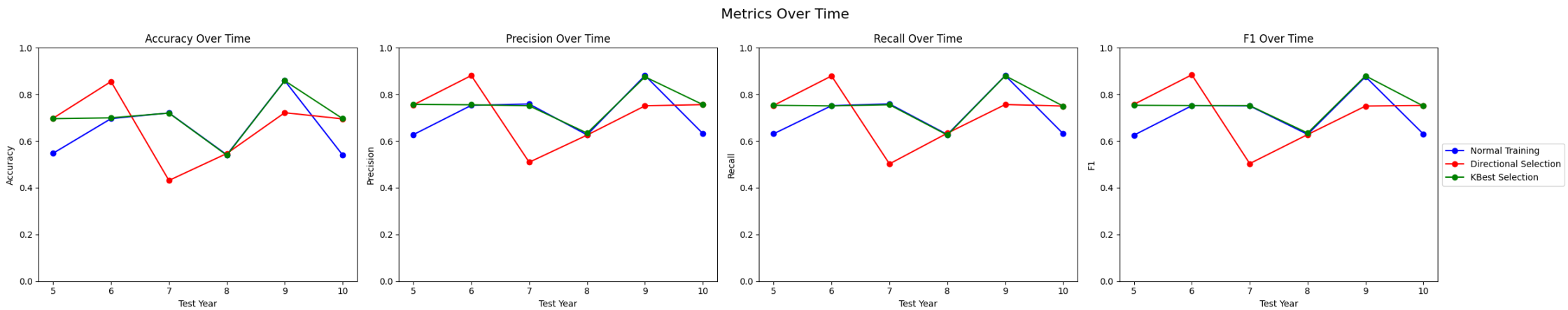


Figure 17 – KNN using bagging

Ideas explored

3 – Merge of the Players dataset

When merging the players with the teams' dataset, we calculated the average of each attribute for all team players in that year.

We attempted to **select the top 3 players** using two different approaches:

- Based on the **number of minutes** they played and

- Based and on their **performance** (which formula was explained earlier in this document). However, this approach did not yield any significant improvements or led to worse results when training the models.

Therefore, it was decided to stick with the original idea of calculating the average for all team players

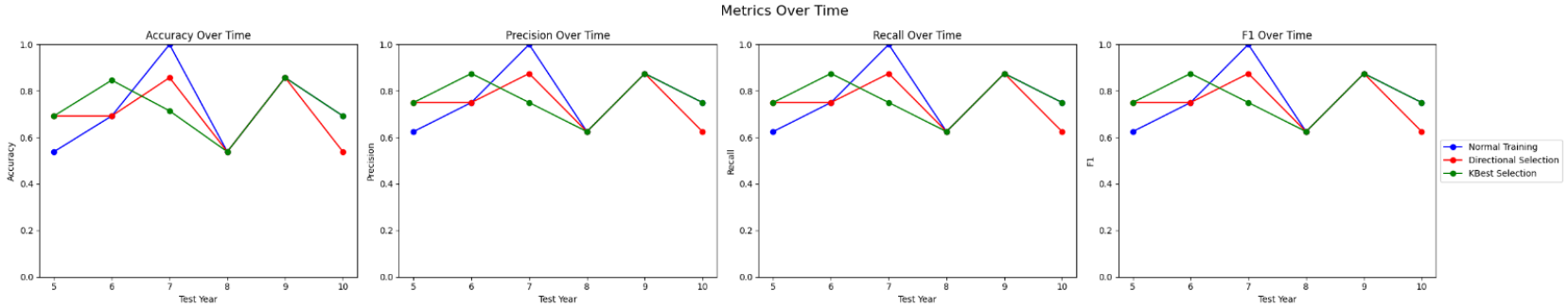


Figure 18 – Results of the voting machine using data from the top 3 players based on their performance.

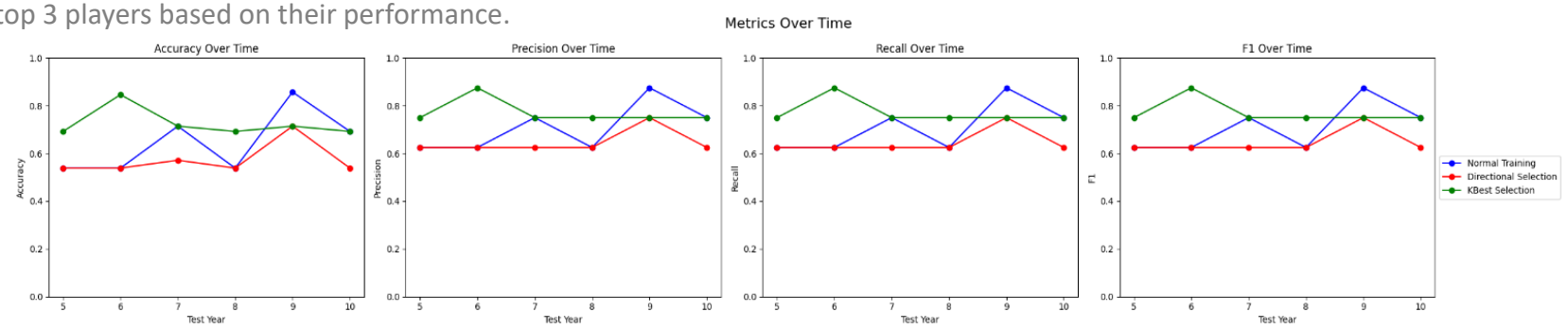


Figure 19 – Results of the voting machine using data from the top 3 players based on their number of minutes played.

Ideas explored

3 – Merge of the Players dataset

The idea of attempting to select the top 3 players using the two different approaches was driven by the creation of graphics that enable the observation of the correlation between the probability of reaching the playoffs based on the team's average performance, and also based on the average minutes played by each team. As it's possible to see on the figures of this slide, there is a strong connection, and **the higher the performance or the minutes played, the higher the probability of reaching the playoffs.**

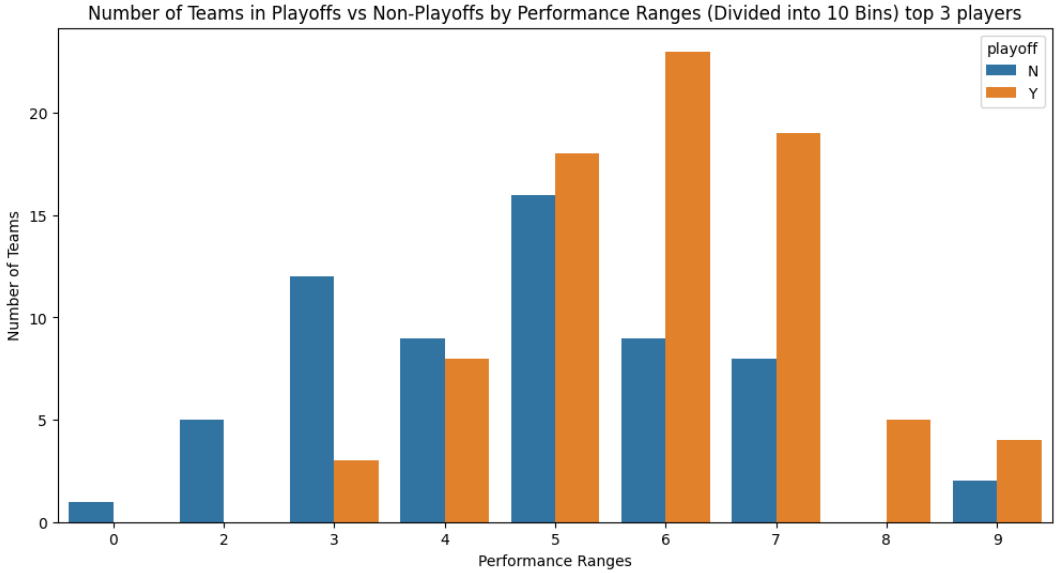


Figure 20 - Number of Teams in Playoffs vs Non-Playoffs by the performance of the top 3 players

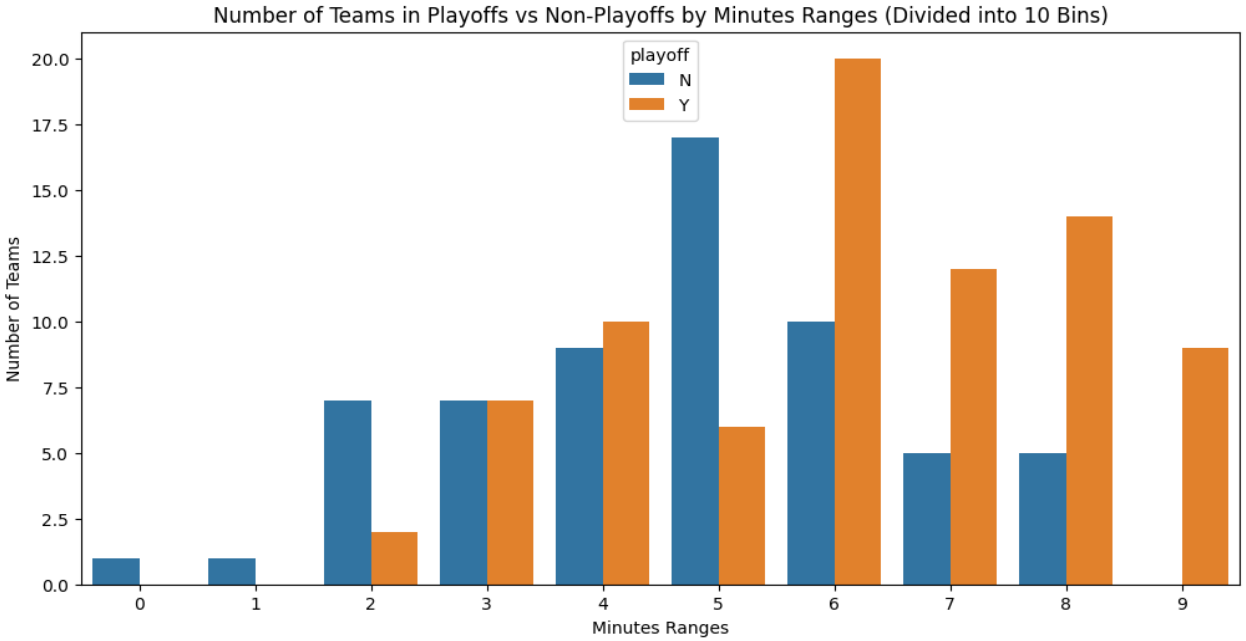


Figure 21 - Number of Teams in Playoffs vs Non-Playoffs by Minutes Ranges

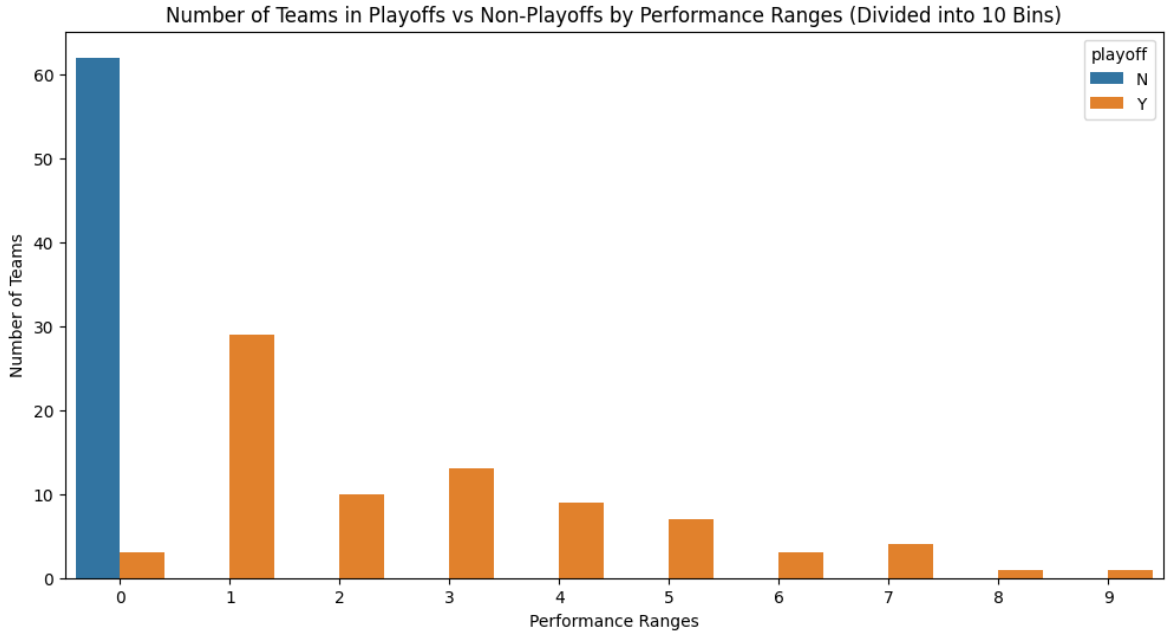


Figure 22 - Number of Teams in Playoffs vs Non-Playoffs by Performance Ranges

Ideas explored

4 – Data Shift of the Players and Coaches

When shifting the data from both the players and coaches, we used two different approaches. Firstly, we shifted the data and calculated the average of their performance in the two years back, regardless of which team they were in during that time. This means that even if a player was in a different team one year and then moved to another team the following year, we still have an idea of their performance and how good they are.

We had to consider, however, that a player's performance is dependent on the team they play with. Therefore, we experimented with a second approach where we analysed data from two years prior while taking into account the team the player was on. However, this approach yielded worse results, so we ultimately decided to stick with the first approach.

Algorithms Chosen

The data mining challenge addressed in this document pertains to a **supervised classification problem**. Therefore, the algorithms chosen to complete the predictive task needed to be **suitable to solve classification problems**.

While several algorithms meeting these criteria were experimented with, the ones exhibiting superior performance included: **KNN, Naïve Bays, Logistic Regression and MLP Classifier.**

KNN

The Algorithm

- **The k-nearest neighbour' algorithm**, also known as KNN, is a **supervised** learning classifier that can be used for either **regression or classification** problems but is more frequently used in classification problems such as the one presented in this document.
- This algorithm forecasts outcomes for new or testing data by assessing their resemblance to known data samples from the training set. This approach assumes that data with comparable characteristics tend to cluster together, **utilising distance measures as a fundamental metric**.
- Bellow, in **figure 13**, it's possible to see the results of KNN without hypertunning and training with data from the previous 3 years.
- This algorithm needed feature scaling.

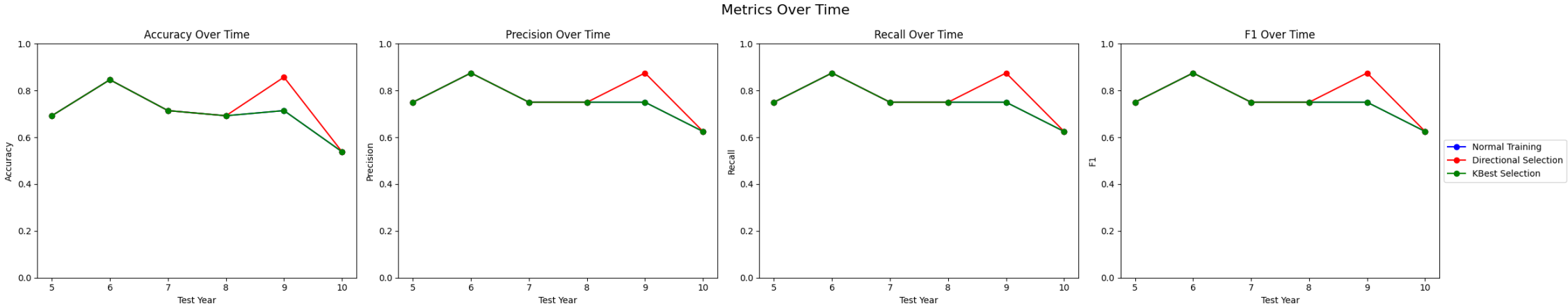


Figure 23 – Results of Testing with the KNN algorithm without hypertunning and training with data from the previous 3 years under three conditions: without any feature selection algorithm, with backward selection, and with Kbest selection

KNN

Below we can see the test and train curves of the KNN results with hyperparameter tuning, both with and without k-best feature selection.

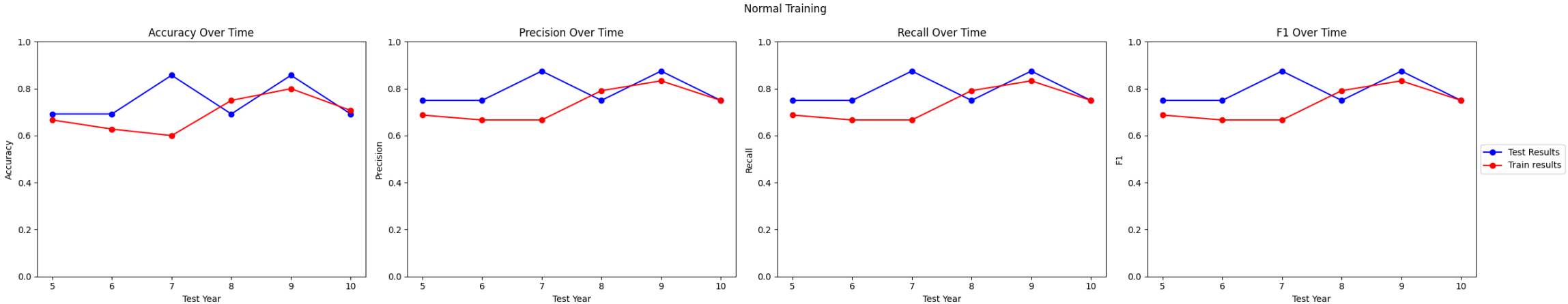


Figure 24 – Training and Testing curves for the KNN with no feature selection algorithm

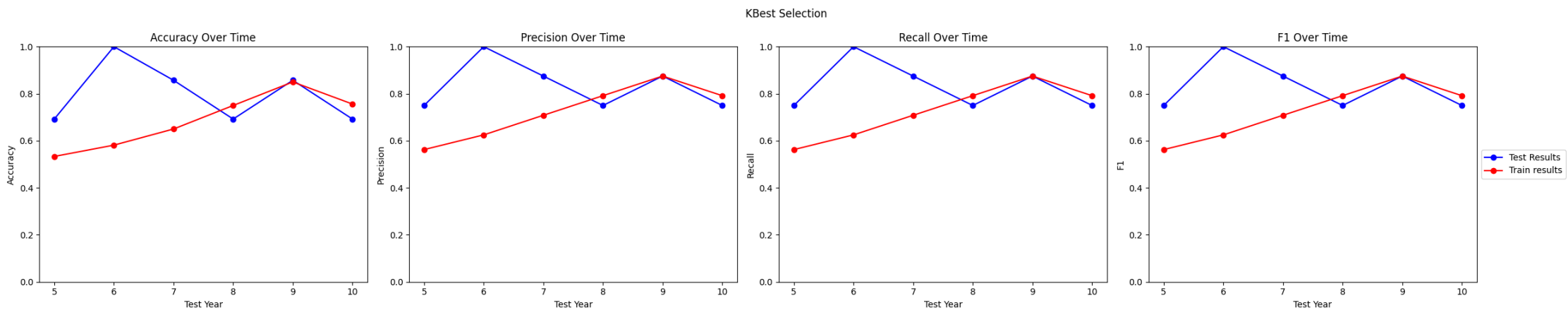


Figure 25 – Training and Testing curves for the KNN with the KBEST algorithm

KNN

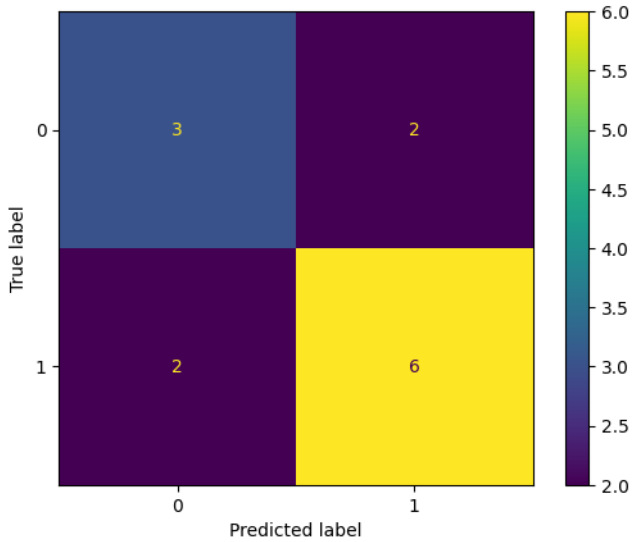


Figure 26 – Confusion Matrix for year 10 of knn with hypertunning and kbest

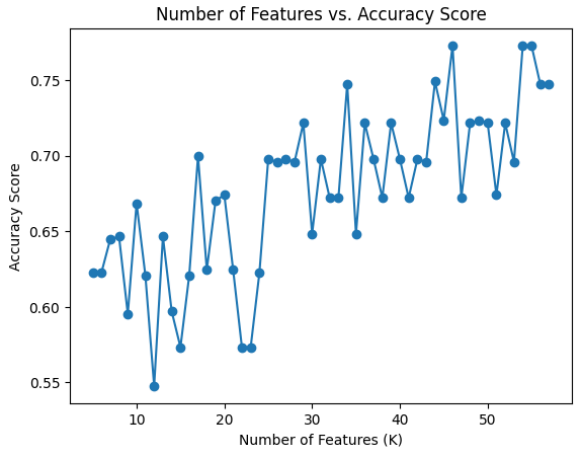


Figure 28 – Variance of the accuracy score with the variance of the number of features for knn

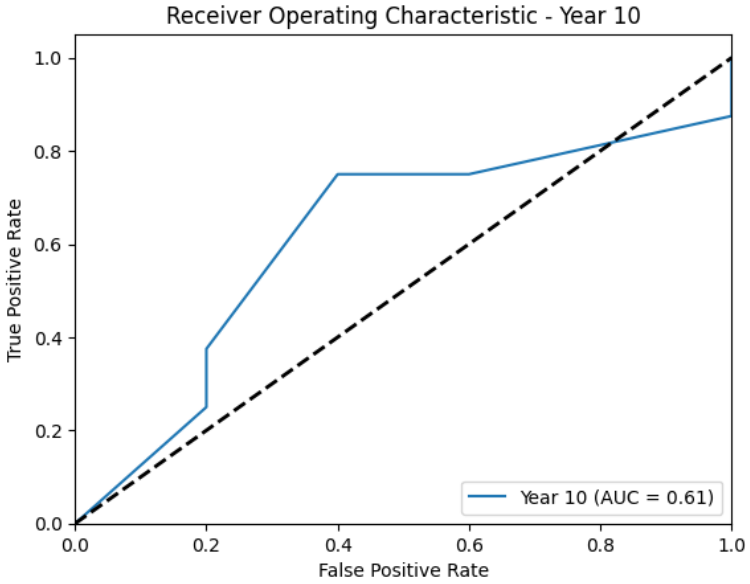


Figure 27 – ROC curve for year 10 of knn with hypertunning and kbest

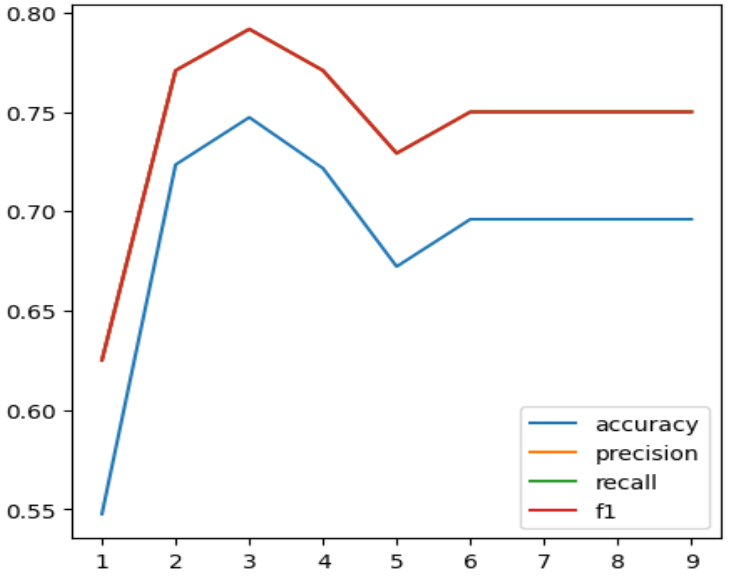


Figure 29 –Average accuracy per years training data for knn

Upon examining the KNN results across the presented graphs on this slide and on the previous two slides, it's possible to deduce that this algorithm demonstrates a **reasonable performance**, although not perfect, and also, it does **not present signs of overfitting or underfitting**, as it's possible to see by the learning curves on those graphs.

It's also possible to say that KNN provided **relatively stable results**. Looking at the graphs provided on the previous slide, it's possible to see that for the testing with hyppertunning, the accuracy only varied between 0.69 and 0.85, reflecting a relatively **small variance**.

When examining the graph from figure 28, it's concludable that KNN **behaves better with the increase of the number of features**.

When examining the graph from figure 29, it's noticeable that the algorithm **performs better when trained with data from limited years back -** performance peaks around year 3, but beyond year 4, there's a noticeable decline.

Naïve Bayes

The Algorithm

- Naive Bayes relies on applying **Bayesian inference** with the strong assumption of independence among variables. While this assumption may **not always hold true in practice**, the algorithm still **performs well in various real-world applications**.

- Therefore, given that Naive Bayes is a **supervised learning classifier** specifically designed for **classification tasks,** such as text classification, it proves to be suitable for the problem outlined in this document. Moreover, due to its good performance, we have chosen this algorithm for our application.
- Bellow, in **figure 17**, it's possible to see the results of Naïve Bayes without hypertunning and training with data from the previous 4 years.

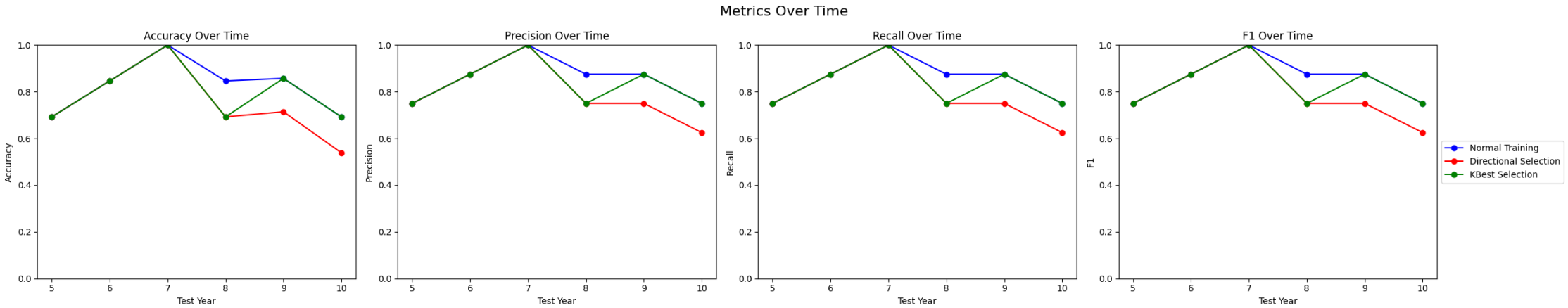


Figure 30 – Results of Testing with the Naive Bayes algorithm without hypertunning and with training data from the previous 4 years under three conditions: without any feature selection algorithm, with backward selection, and with Kbest selection

Naïve Bayes

Below we can see the test and train curves of the Naïve Bays results with hyperparameter tuning, both with and without k-best feature selection. As you can see the accuracy stayed the same with and without hypertunning as well as with and without kbest

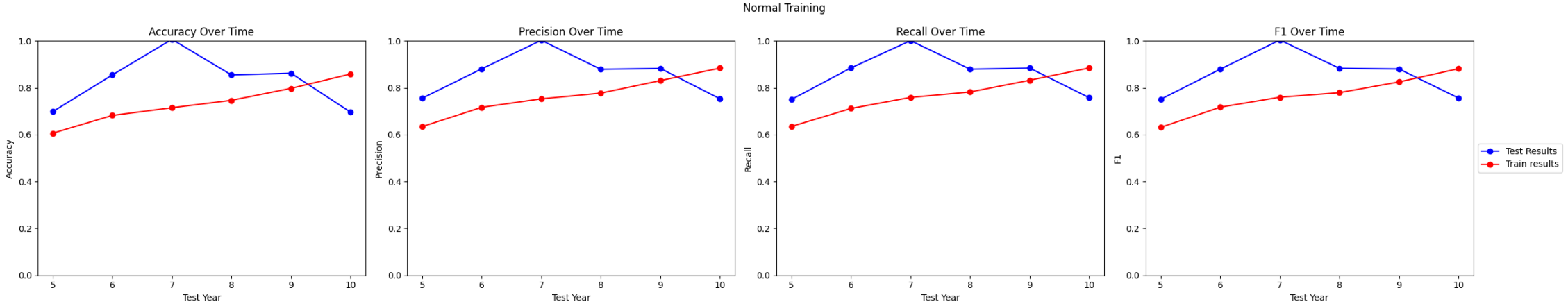


Figure 31 – Training and Testing curves for the Naive Bayes with no feature selection algorithm

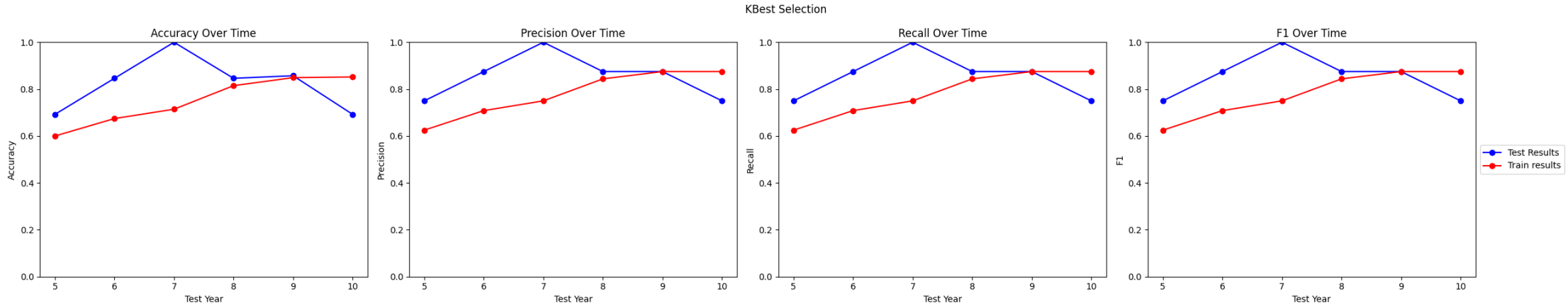


Figure 32 – Training and Testing curves for the Naive Bayes with the KBEST algorithm

Naïve Bayes

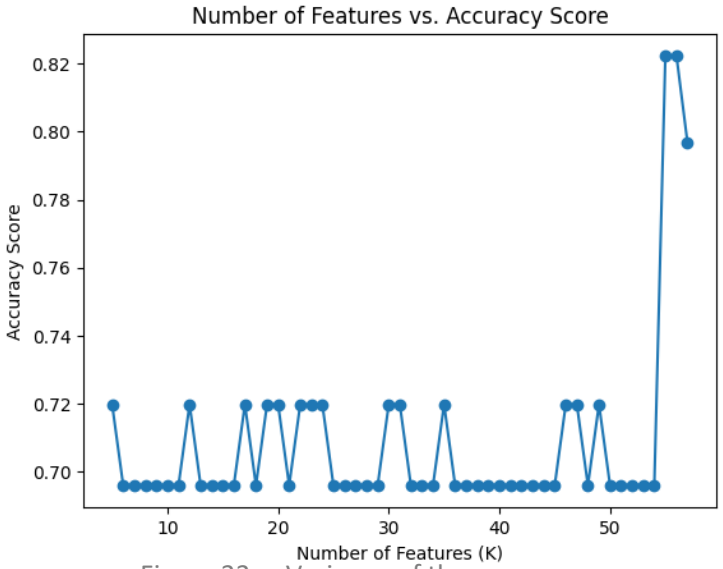


Figure 33 – Variance of the accuracy score with the variance of the number of features of naïve bayes

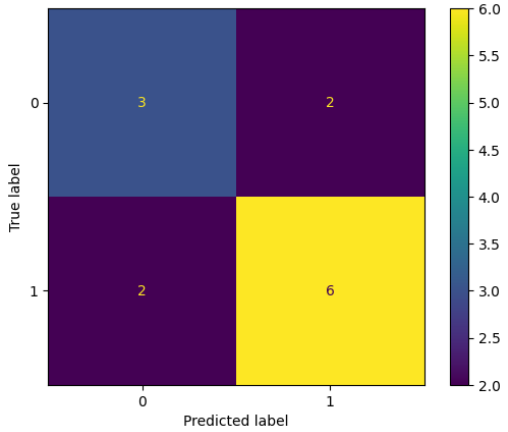


Figure 35 – Confusion Matrix for year 10 of naive bayes with hypertunning and kbest

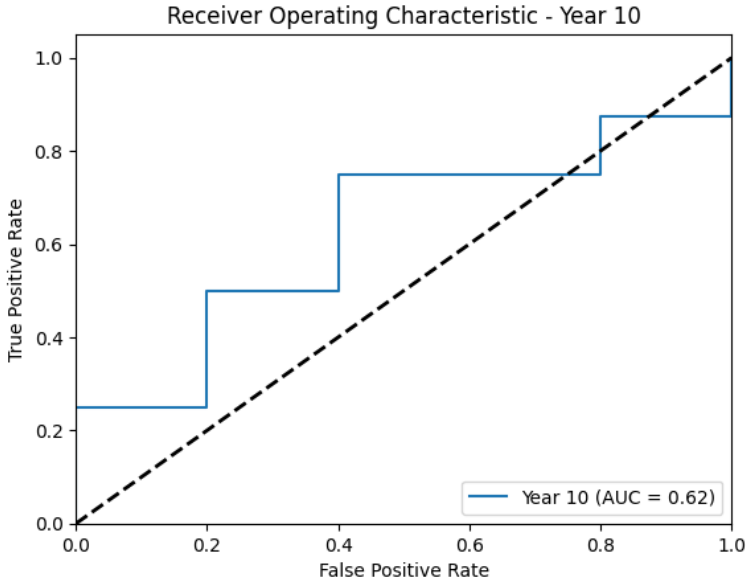


Figure 34 – ROC curve for year 10 of naive bayes with hypertunning and kbest

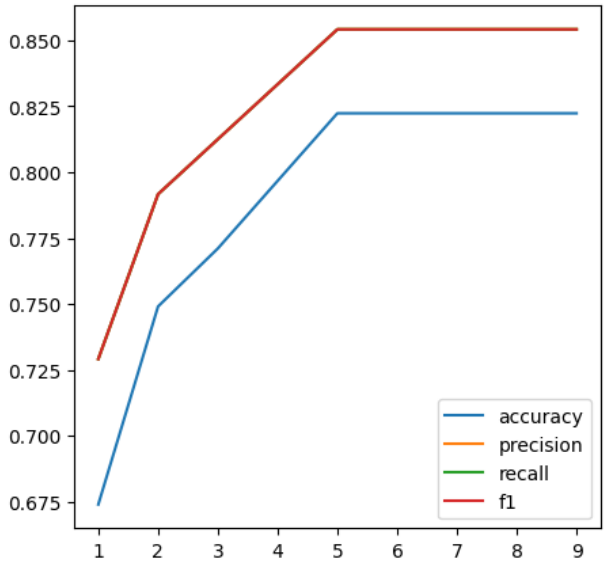


Figure 36 –Average accuracy per years training data for naïve bayes

Upon examining the Naïve Bayes results across the presented graphs on this slide and on the previous two slides, it's possible to deduce that this algorithm demonstrates a **reasonable performance**, although not perfect, and also, it does **not present signs of overfitting or underfitting**, as it's possible to see by the learning curves on those graphs.

It's also possible to say that Naive Bayes provided **relatively stable results**. Looking at the graphs provided on the previous slide, it's possible to see that for the testing with hyppertunning, the accuracy only varied between 0.69 and 0.85, reflecting a relatively **small variance**.

When examining the graph from figure 36, it's noticeable that the algorithm **performs better when trained with more data from years back** - performance peaks around year 5, and stays stable with that peak value beyond that year.

Logistic Regression

The Algorithm

- Logistic Regression is a **supervised learning algorithm** particularly effective in **binary classification tasks**, just like the presented problem in this document.
- This algorithm learns a linear relationship between the input features and the log-odds of the output belonging to the positive class. The logistic function is then applied to transform the log-odds into probabilities between 0 and 1.
- Bellow, in **figure 22**, it's possible to see the results of Logistic Regression without hypertunning and training with data from the previous 5 years.

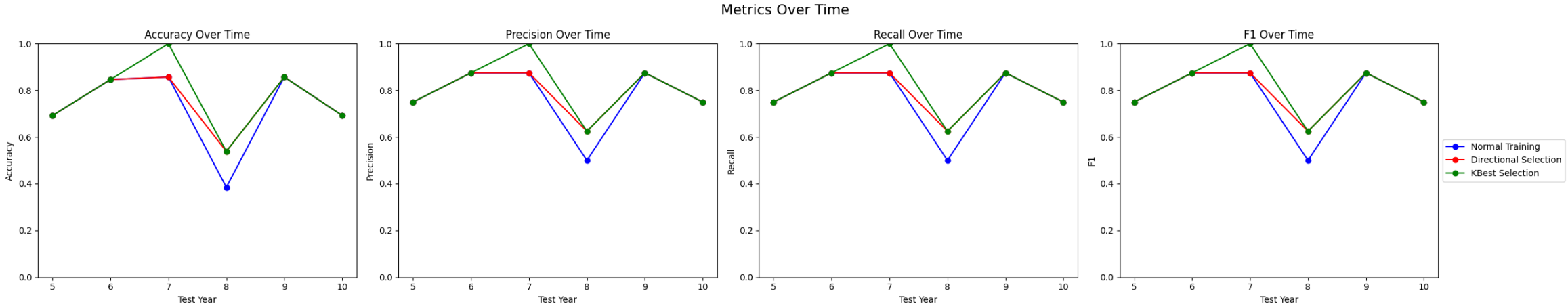


Figure 37 – Results of Testing with the Logistic Regression algorithm without hypertunning and with training data from the previous 5 years under three conditions: without any feature selection algorithm, with backward selection, and with Kbest selection

Logistic Regression

Below we can see the test and train curves of the KNN results with hyperparameter tuning, both with and without k-best feature selection.

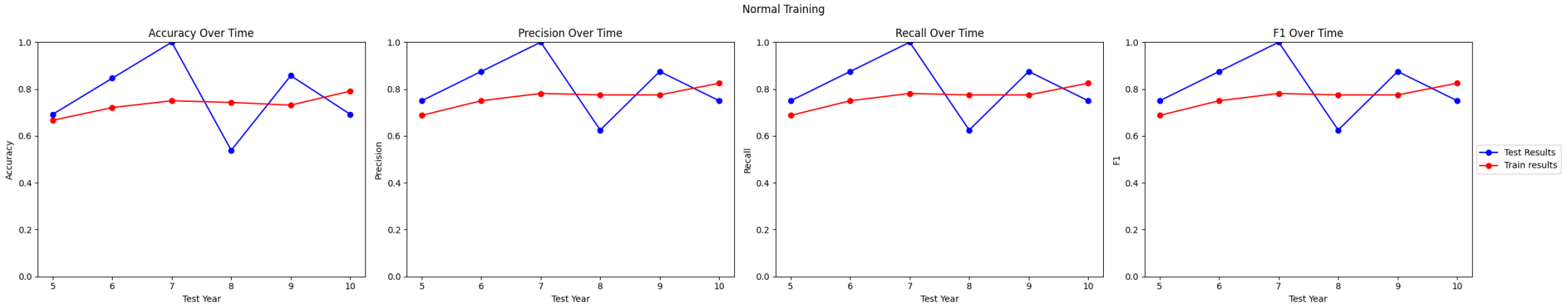


Figure 38 – Training and Testing curves for the Logistic regression with no feature selection algorithm

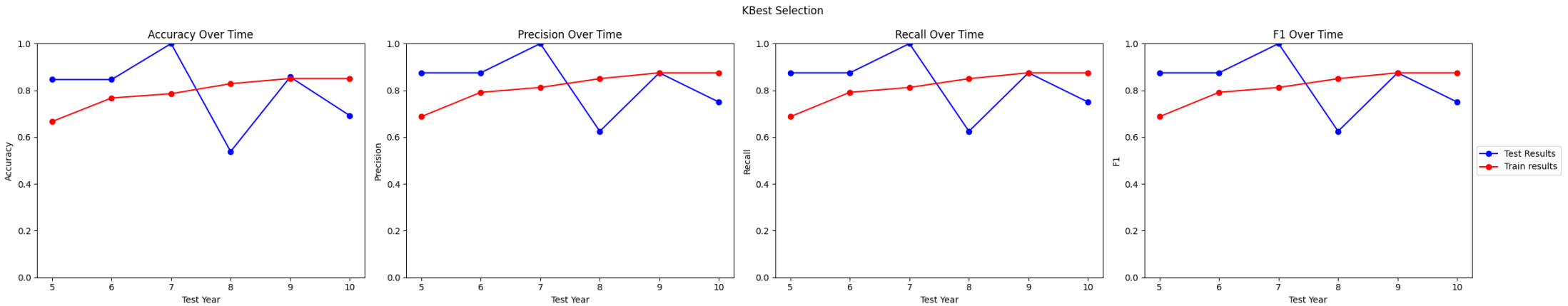


Figure 39 – Training and Testing curves for the Logistic Regression with the KBEST algorithm

Logistic Regression

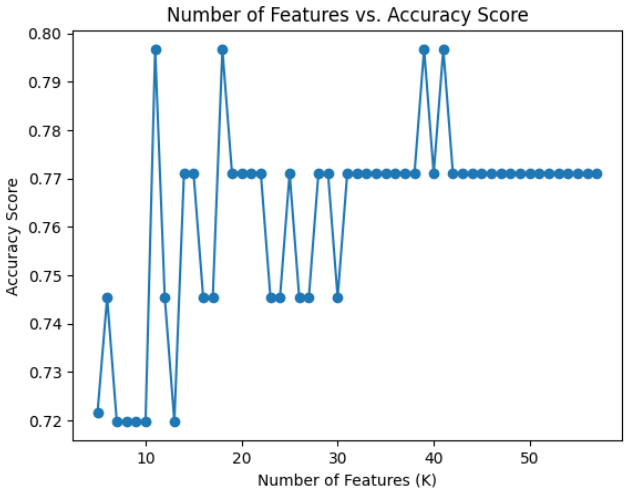


Figure 40 – Variance of the accuracy score with the variance of the number of features of logistic regression

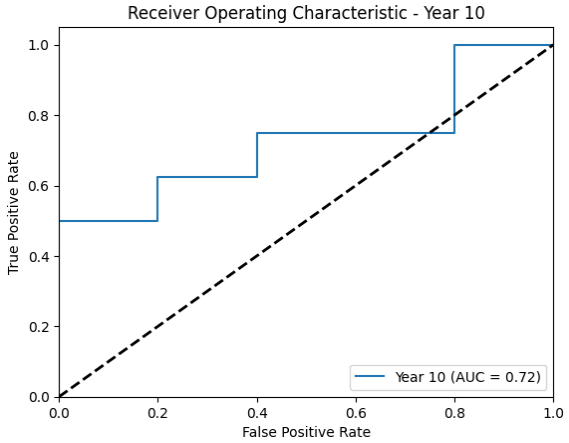


Figure 41 – ROC curve for year 10 of logistic regression with hypertunning and kbest

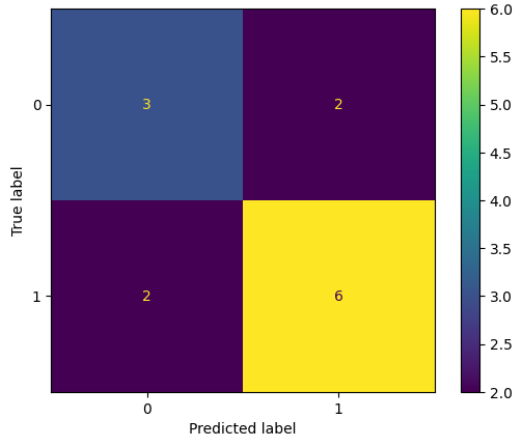


Figure 42 – Confusion Matrix for year 10 of logistic regression machine with hypertunning and kbest

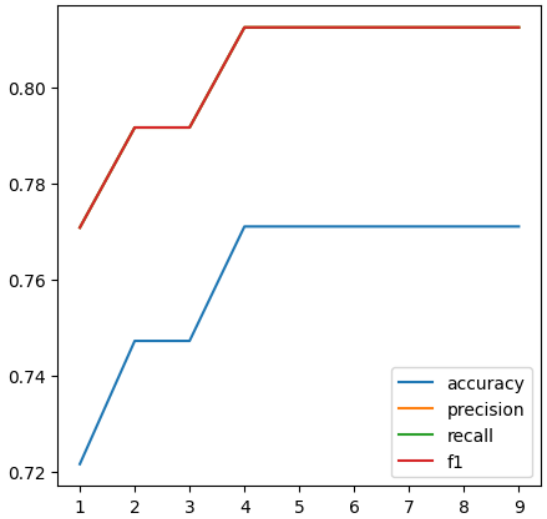


Figure 43 –Average accuracy per years training data for the Voting Machine Classifier

Upon examining the Logistic Regression results across the presented graphs on this slide and on the previous two slides, it's possible to deduce that this algorithm demonstrates a **reasonable performance**, although not perfect, and also, it does **not present signs of overfitting or underfitting**, as it's possible to see by the learning curves on those graphs.

Logistic Regression provided **the most unstable results when compared to the other algorithms used in the ensemble machine.**

Upon analysing the graph in Figure 40, it can be concluded that Logistic Regression performs more effectively with an increased number of features, achieving particular stability when the number of features exceeds 20.

When examining the graph from Figure 43, it's noticeable that the algorithm **performs better when trained with more data from years back -** performance peaks around year 4 and stays stable with that peak value beyond that year.

MLP

The Algorithm

- The Multi-Layer Perceptron, also known as MLP, is a **supervised learning algorithm** also **very effective in classification tasks**, just like the presented problem in this document.
- This algorithm is a type of feedforward neural network with multiple layers, including an input layer, one or more hidden layers, and an output layer. It learns from a labeled dataset during the training phase, where the input data is associated with corresponding output labels.
- Bellow, in **figure 27**, it's possible to see the results of MPL without hypertunning and training with data from the previous 4 years.
- **This algorithm needed feature scaling.**

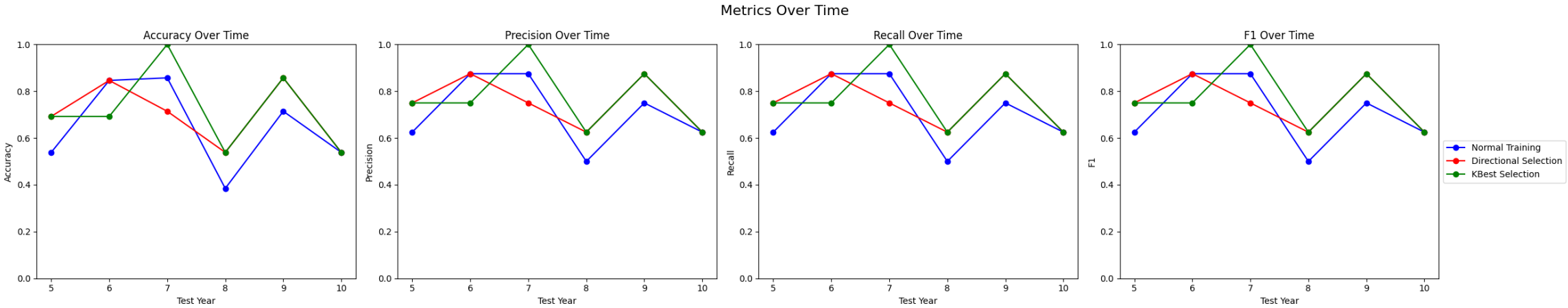


Figure 44 – Results of Testing with the MPL algorithm without hypertunning and training with data from the previous 4 years under three conditions: without any feature selection algorithm, with backward selection, and with Kbest selection

MLP

Below we can see the test and train curves of the MLP results with hyperparameter tuning, both with and without k-best feature selection.

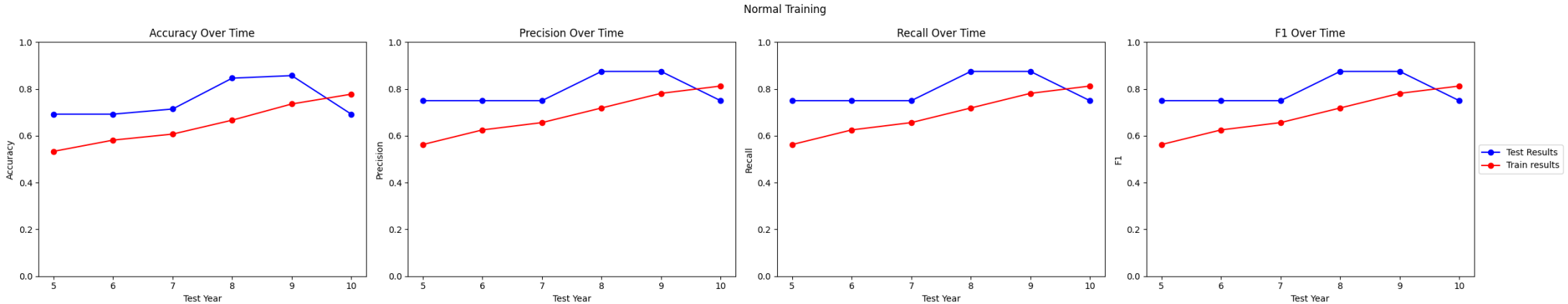


Figure 45 – Training and Testing curves for MLP with no feature selection algorithm

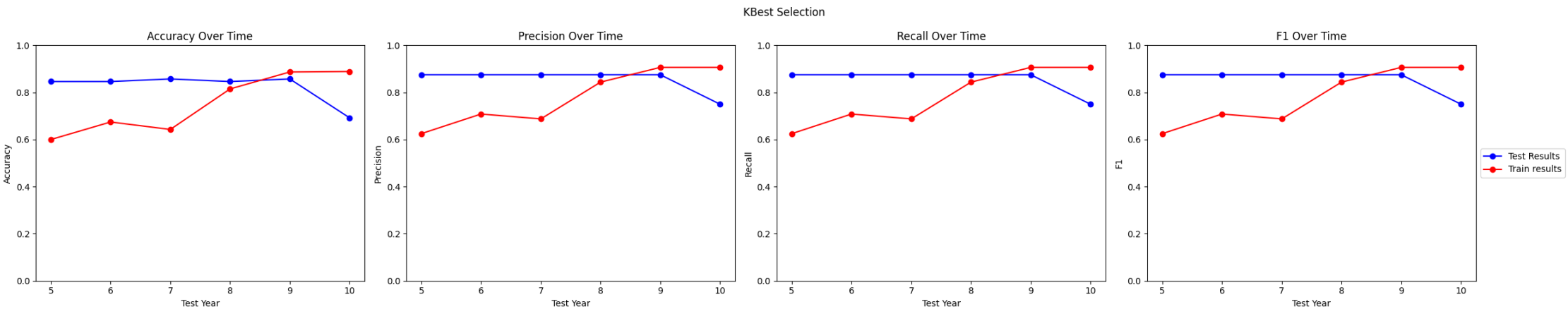


Figure 46 – Training and Testing curves for MPL with the KBEST algorithm

MLP

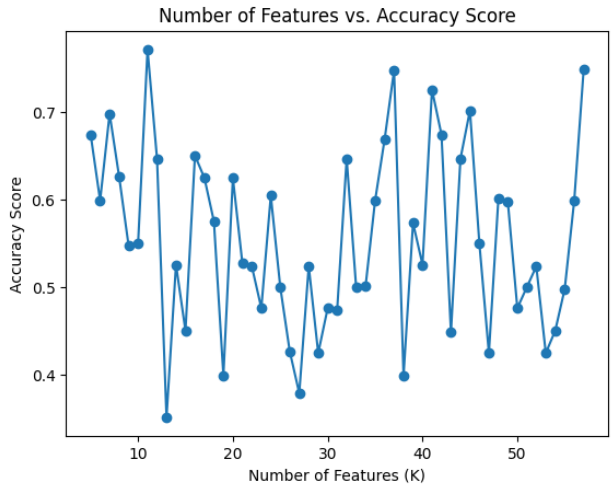


Figure 47– Variance of the accuracy score with the variance of the number of features for MLP

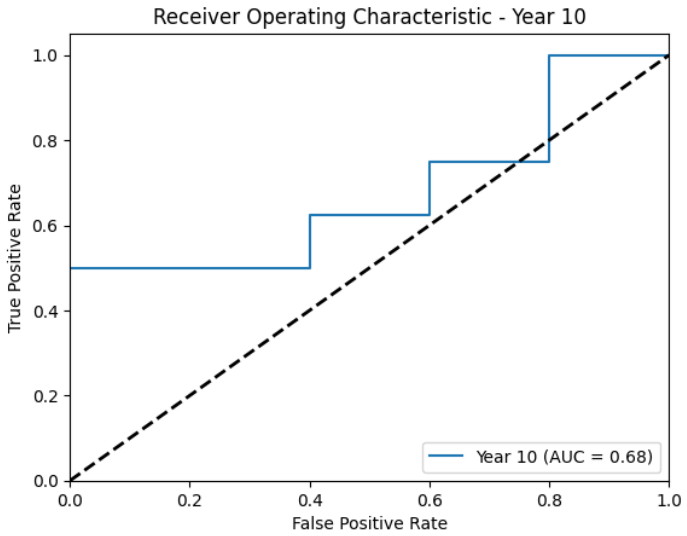


Figure 49 – ROC curve for year 10 of MLP with hypertunning and kbest

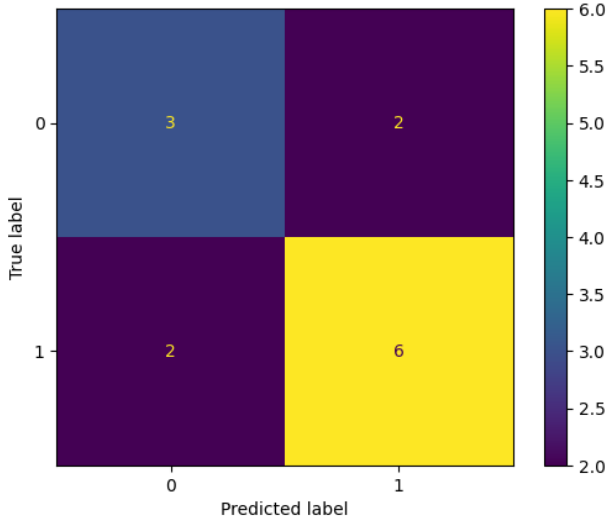


Figure 48 – Confusion Matrix for year 10 oF MLP with hypertunning and kbest

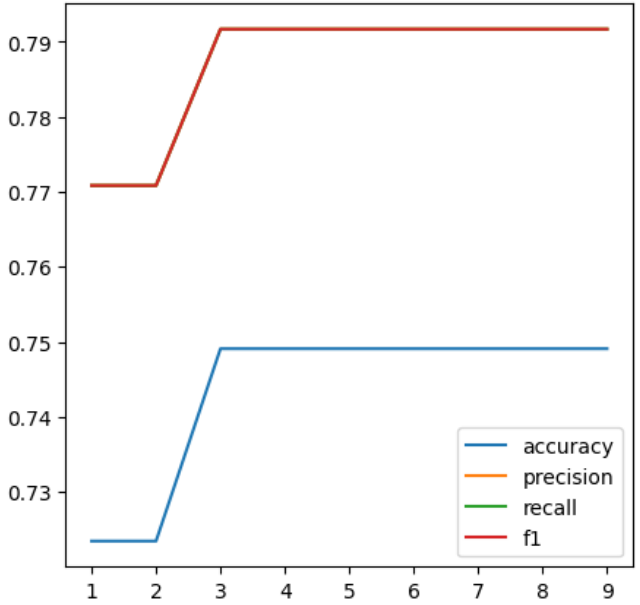


Figure 50 –Average accuracy per years training data for MLP

Upon examining the MLP results across the presented graphs on this slide and on the previous slide, it's possible to deduce that this algorithm presents **good and stable results**.

Based on the graphs presented on the previous slide, it is evident that the accuracy for the testing with hypertuning varied between 0.69 and 0.85, indicating a relatively small variance. The best results were achieved through the combination of both kbest and hypertuning. The learning curves also indicate a stable increase in accuracy.

Further analysis of Figure 47's graph establishes that this algorithm attains its **optimal performance with approximately 10 features**.

Examining the graph in Figure 50 reveals a distinct pattern: after the algorithm trained with three years back, it stagnates on a peak. This implies that, beyond this point, the increased volume of training data has a reduced impact on the algorithm's performance.

Random Forest

Random Forest was one of the algorithms that we also tested with, although yielded good results, **had overfitting** so it was discarded.

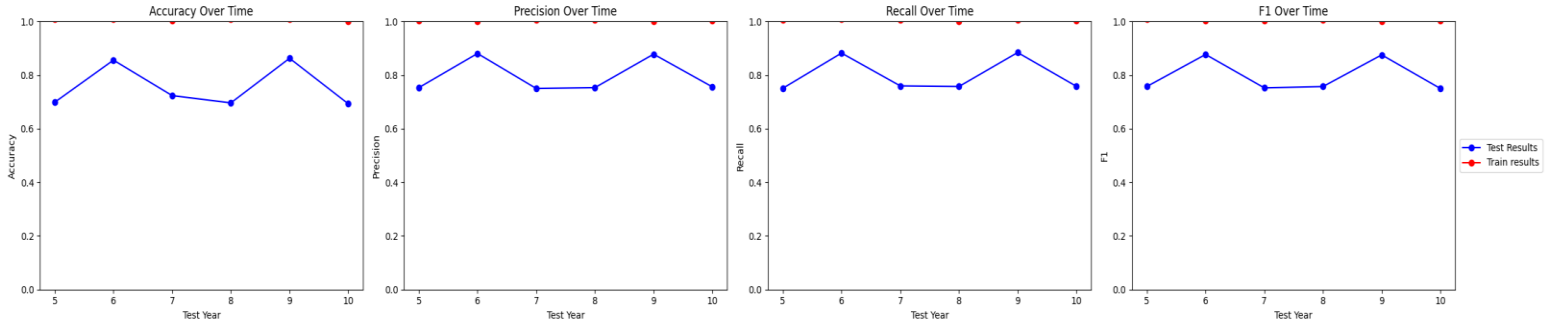


Figure 30 – Training and Testing curves for the Random Forest with no feature selection algorithm

LightGBM

LightGBM was one of the algorithms that we also tested with. As it's possible to see in figure 31, the results weren't the best. Which makes sense, since this algorithm uses the boosting method, and as we have predicted earlier, for the case of our problem, bagging would be more suitable.

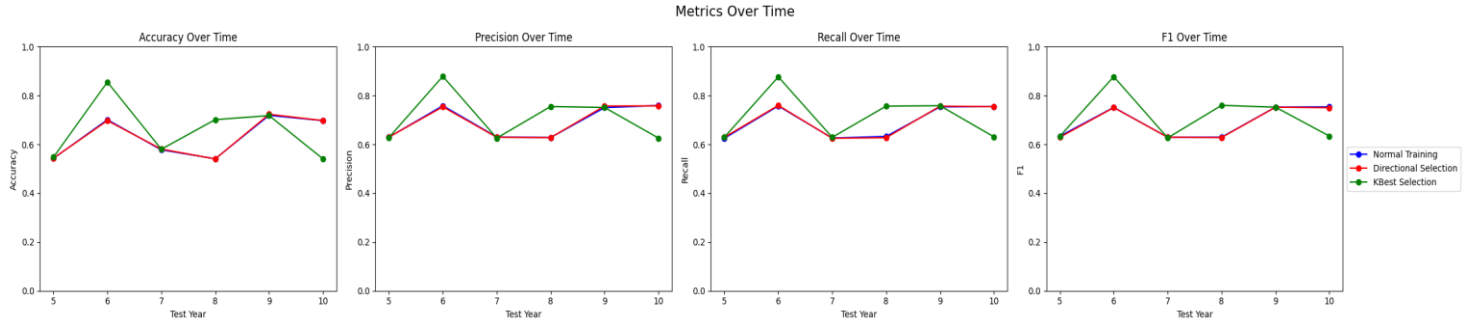


Figure 31 – Training and Testing curves for the LightGBM algorithm with no feature selection algorithm