Predicting the Performance of Football Teams in the Italian Serie A League Using Neural Networks and Machine Learning Methods through Historical Data Analysis

Bachelor's thesis

Degree Programme in Computer Applications

Spring 2024

Masoud Amininiaki

HAMK
Häme University
of Applied Sciences

Degree Programme in Computer Applications
Author Masoud Amininiaki
Subject Predicting the Performance of Football Teams in the Italian Serie A League Using Neural Networks and Machine Learning Methods through Historical Data Analysis
Supervisors Mazhar Mohsin

Abstract
Year 2024

The aim of this thesis is designing several prediction models based on machine learning algorithms and neural networks methods to predict the performance of football teams, their final ranking and the champion of Serie A football league based on the available historical data of the football matches. In this regard, seven predictive models were created in Jupyter Notebook which is an environment based on Python programming language. The models were analytically and comprehensively compared in the case of errors and accuracies. So the best predictive model was introduced. The ranking provided by the betting websites was compared with the ranking of the teams based on the predictive models of this thesis. The importance of the variables used in the predictive models on the target variable was determined and it was defined which feature has the greatest impact on the prediction models.

At the first step in the modeling, the data required for use in the predictive models was collected from different sources and entered in two csv files. One of them is for the data of seven past football leagues and another includes the data of the current Serie A football league (2023-2024). After data processing, seven prediction models which are Linear Regression (LR), Decision Tree (DT), Gradient Boosting Machine (GBM), Neural Networks (NN), Random Forest (RF), Support Vector Machine (SVM) and Extreme Gradient Boosting (XGB) based on the machine learning and neural networks methods were designed in Python. These models were validated after training and testing. The method of validation used in this research was 5 fold cross-validation. The outcomes of these seven prediction models were compared analytically. Also, the results related to the errors (Mean Absolute Error and Mean Squared Error) and the accuracy criteria for each model were presented in appropriate tables and graphs.

After comparing the analysis of the predictive models and presenting detailed examination of each predictive model, it was found that the LR, GBM, RF, DT and XGB models are suitable models for prediction the performance of football teams and they have been able to correctly predict the team ranking of the current league more than the other models. The most important feature was average goals per game. Among those 5 suitable models, GBM and Linear Regression had the most accurate predictions.

Keywords Machine Learning, Neural Networks, Performance of Football Teams, Predictive Models, Python.

Pages 43 pages and appendix 1 page

Glossary

DT	Decision Tree
GBM	Gradient Boosting Machine
LR	Logistic Regression
MAE	Mean Absolute Error
MSE	Mean Squared Error
NN	Neural Networks
RF	Random Forest
SVM	Support Vector Machine
XGBoost	Extreme Gradient Boosting
XGB	Extreme Gradient Boosting

Content

Figures, program codes and tables

Appendices

1 Introduction

Football is the most popular sport in the world, and a significant amount of wealth is circulating in this popular sport in different ways. Football is very popular in Italy and this money-making industry is followed in a professional and modern way in this country. There is a famous sentence about football which says that football is unpredictable. Football clubs spend a lot of money to make their team succeed in football matches. They all use all available tools to win in this way and also welcome new tools and methods that help the development of their desired club team. Besides the football club betting companies are always trying to make predictions closer to reality and try to improve the betting factors of football league matches.

The aim of this thesis is to present some machine learning models for predicting the performance of football teams of the Italian Serie A league, their rankings and the champion of the current league (2023-24) based on the detailed historical data which were collected from 7 past leagues such as averages of points, goals, shots on target, etc. Implementing the historical data was done using some methods of machine learning and neural networks algorithms like Random Forest via Jupyter Notebook which is based on the programming language of the software package of Python. Besides, comparing the predictive models and choosing the best and the most accurate model for predicting the performance and rankings of the football teams ahead of the Italian Serie A league in the 2023-2024 season has been done.

The thesis ahead is both research based and practical based. It focuses on simulating several predictive models of the performance of football teams. So, gambling companies can use the predictive models to improve the factors of betting. For instance they can improve the coefficient of the champion of the football league. Also football clubs can implement the prediction models to have a better performance in the football league. For example they can improve choosing the desired tactics and line-up players.

This thesis will explore the best prediction model for performance of football teams by answering the following questions:

- Which of the predicting models is the best model for predicting the performance of football teams?
- Is the predicting model of this thesis more accurate than existing predicting models of the betting companies?
- Which of the characteristics of the football teams has a greater impact on the prediction model of the performance of football teams?

2 Basics of football prediction

Professional sports competitions are generally unpredictable. This is the reason why group sports competitions are attractive. As the most popular sport in the world, football is one of the most unpredictable sports. In football matches, many factors affect the outcome of the game such as the conditions of the field, weather, refereeing mistakes, etc. That's why there may be situations in the game where the weaker football team defeats the stronger football team. Therefore, it is not possible to make a definite prediction of football matches, if this happened, betting companies would go bankrupt and football matches would no longer be attractive and popular. But there are ways to predict football matches with high accuracy. (Sjöberg, 2023, p. 1). So suitable prediction models in football matches can be designed with the machine learning and artificial intelligence methods.

2.1 An overview of the importance of football prediction

In recent decades, football has continued to draw worldwide attention from people of various ages and social situations. The game result is, like in most other sports, is usually used to assess the performance and success of a team or a single player. Given the importance of the score, it is hardly surprising that many gamblers try to guess the result of football matches. The motivation is on the one hand driven by the admiration of other enthusiasts, on the other hand monetary rewards serve as an incentive system (Stubinger et al., 2019, p. 1).

Football matches are popular among people of all ages. In recent years, football matches have been very popular from the point of view of generating income. Betting companies have made a lot of money on this issue. For this reason, predicting and checking team performance has been very important for users of betting sites. The act of predicting itself is exciting even if the bet doesn't happen. It is natural that the presence of money as a reward increases the motivation of football fans (Tiwari et al., 2020, p. 229). Figure 1 shows that the trend of investing in betting on different sports has been increased.

Figure 1 Sports betting market size from 2021 to 2030 (visionresearchreports, 2024)

2.2 Advances in machine learning algorithms and Neural Networks

In the past, it has been difficult to predict anything, especially the results of football matches. This difficulty had many causes, for example, there was no complete historical data because the proper tools for this work were not available. But today, with the advancement of data science, appropriate tools have been provided to predict the results of football matches. Using predictive models based on machine learning algorithms and neural networks increases the accuracy and reliability of predictive models.

In recent decades, a lot of historical data related to all sports, including football, is available online. There are many sites and tools online that researchers and enthusiasts can use to easily design and develop their prediction models (Sjöberg, 2023, p. 1).

Important milestones in the development of machine learning algorithms from the past to the present:

- 1950s-1960s: Appearance of machine learning methods, focusing on pattern detection.
- 1980s: Birth of symbolic reasoning and proficient applications.
- 1990s: Shift to methods based on statistics and probability.
- 2000s: The emergence of big data and the rise of neural networks.
- 2010s: Deep learning explained the possibilities which is leading to developments in image and speech detection of image and speech.

The progress paths of machine learning can be summarized in the following cases:

- Explainable AI: Request for transparency in AI decision making.
- Edge Computing: Make machine learning abilities closer to the data resources.
- AutoML: Automated machine learning workflows for non-experts.
- Federated Learning: Collaborative training models without centralized data.

Machine learning has creative uses that go beyond its traditional uses.:

- Generative Adversarial Networks (GANs): Production of real content.

- Natural Language Processing (NLP): Creating sticking text.

- Teaching machines to watch: Power of computer vision (Ileaf Solutions, 2023)

2.3 Steps to obtain a suitable prediction model

For predicting the performance of football teams, at first, information and statistics related to the past should be collected. Many features affect the prediction of the performance of football teams, including averages of points, goals, shots on target, possession, pass accuracy, corners, etc. Gomes et al., (2015, p. 349) stated the operation is composed by nine steps that can be summarized in the next five steps:

• Selection: In this step, it is determined what data should be collected and the desired sources for data collection are determined. Also, the target variable and features are determined. The number of excel or csv files is known and data collection is done from different sources;

• Pre-processing: After data collection, data cleaning will be done. For example in this phase, meaningless values are replaced with suitable meaningful numbers;

• Transformation: In this step, to increase the quality of the data and have a better understanding of the data set, data conversion is done according to the dependence between them;

• Data Mining: This phase is done in order to find patterns and interesting points in the dataset;

• Evaluation: In this part, the models are reviewed and analysed and evaluated, and the model that provides the best result according to the set goal is selected.

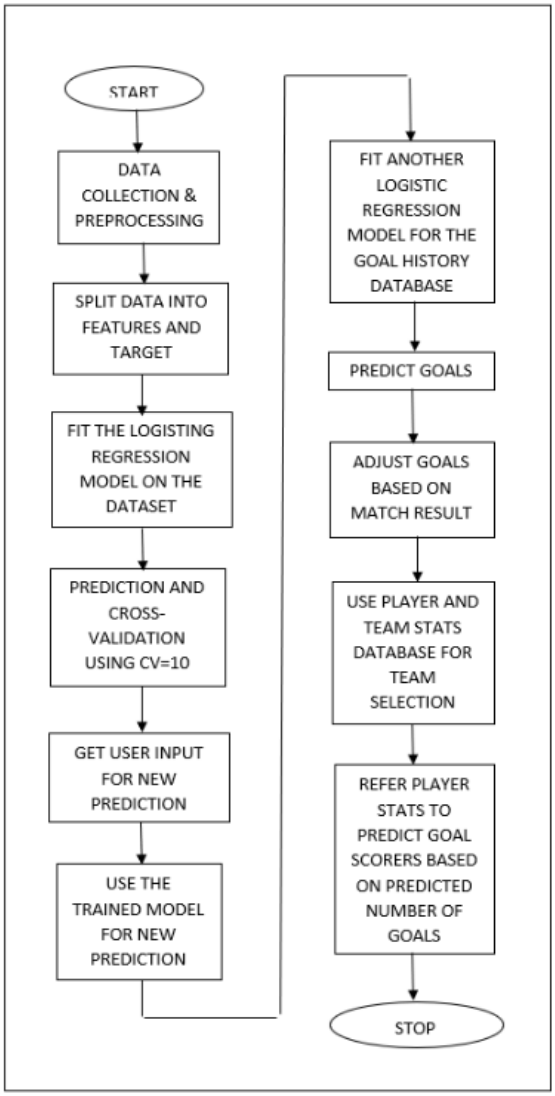
After collecting data, analyzing them and determining a set of features and data will be used for modeling using machine learning and neural networks methods. The desired historical data will be implemented using several machine learning algorithms. Zaveri et al., (2018, p. 164) stated that the prediction models are designed based on different algorithms of machine learning and artificial intelligence such as Logistic Regression, Random Forest, Artificial Neural Network, Linear SVM and Naïve Bayes using Python Sci-Kit Learn library. The training and testing of the data was done and cross validation method was chosen for validate the prediction models.

In the science of machine learning (Machine Learning), the issue of designing machines that learn by using the examples given to them and their own experiences is discussed. In fact, in this science, an attempt is made to design a machine in such a way that it can learn and act without explicitly

planning and dictating each and every action by using algorithms. In machine learning, instead of programming everything, data is fed to a generic algorithm, and it is this algorithm that builds its own logic based on the data it is fed. Examples of machine learning include medical diagnosis, face recognition, weather forecast, etc (El Naqa & Murphy, 2016).

Since the past few decades, when computers have made it possible to implement computational algorithms, in the direction of simulating the computational behavior of the human brain, many research works have been started by computer science experts, engineers and also mathematicians, whose results are in a branch of artificial intelligence science. And it is classified in the subsection of computational intelligence under the title of "Artificial Neural Networks" or ANNs. In the topic of artificial neural networks, many mathematical and software models inspired by the human brain have been proposed, which are used to solve a wide range of scientific, engineering and applied problems in different fields. (Rahman, 2020). Figure 2 shows a good example of implementation of machine learning and neural networks methods.

Figure 2 Flow Diagram of implementation of machine learning methods (Zaveri et al., 2018, p. 164)



The main goal of machine learning and neural network methods is to use historical data from past matches to train and validate a model for predicting future performance of football teams. Historical data provides valuable insights into the performance trends of teams and players over time and allows us to identify key factors that contribute to match outcomes. Through rigorous data analysis techniques, we seek to extract meaningful features and patterns that can improve our predictive model design and optimization.

3 Modeling framework

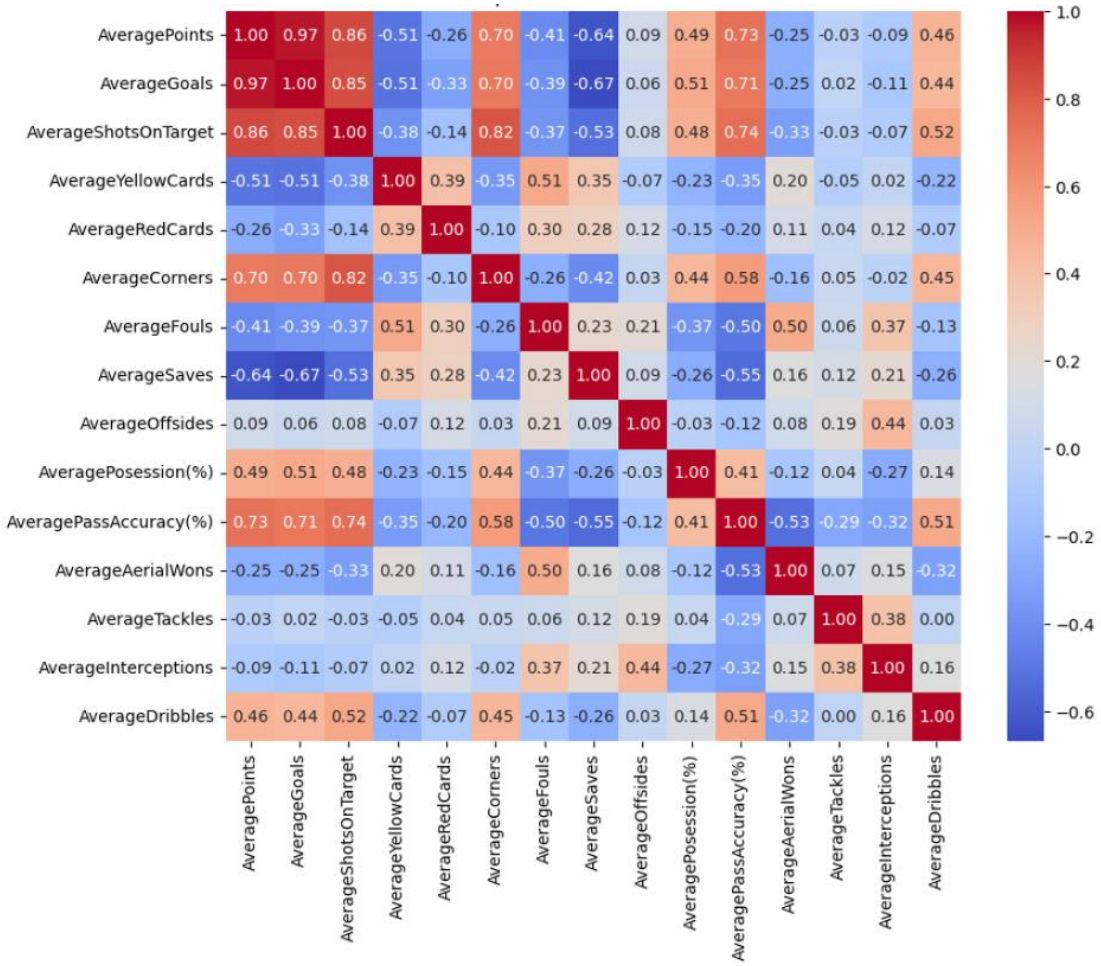
The aim of this thesis is to present several models for predicting the performance of football teams of the Italian Serie A league. First, research was done on the websites that provide the statistics of football matches. This statistic includes averages of feature per game. The feature are points, goals, shots on target, possession, pass accuracy, corners, etc. In this thesis, detailed information will be collected from the last 7 Italian Serie A leagues. The next stage of modeling is based on the historical data collected. So implementing the historical data was done using some methods of machine learning and neural networks methods like Linear Regression, Random Forest, Support Vector Machines (SVM), XGBoost, etc. Programming has been done by Jupyter Notebook which is based on the programming language of the software package of Python. Then the prediction models were tested and validated. Final step is choosing the best model for predicting the performance of football teams ahead of the Italian Serie A league in the 2023-2024 season.

3.1 Data collection and preprocessing

First, historical data related to past football matches in the Italian Serie A has been collected. Each league includes 20 football teams. The collected data from 7 last seasons (from 2016-17 until 2022-23) includes: Season, team, average points per game, average goals per game, average shots on target per game, average yellow cards per game, average red cards per game, average corners per game, average fouls per game, average saves per game, average offsides per game, average possession (%) per game, average pass accuracy (%) per game, average aerial wins per game, average tackles per game, average interceptions per game and average dribbles per game. The mentioned data were retrieved from several websites which present statistics of football matches. The data were entered in to a csv file. The values of historical data were entered into the csv file with 140 rows and 17 columns.

Then the data had to be preprocessed. In this way, the data was sorted, and empty numbers had to be filled with zero, the meaningless numbers had to be meaningful and the non-numerical information had to be converted into numbers. The collected data has been investigated and it was without zero numbers or non-numerical information. Data preprocessing can be done using some Python libraries like NumPy, Scikit-learn, Keras and Pandas. For better understanding of the collected data some related tables and graphs like boxplots and scatterplots were investigated. Figure 3 shows the Hitmap plot of correlation between variables. Another csv file which is statistics of current league (2023-24) was created. This csv file includes data for 33 matches of Serie A league 2023-24. It means 5 matches remain. In the csv file, the target variable which is average points per game was removed from the data.

Figure 3 Hitmap plot of correlation coefficients of variables



3.2 Designing predictive models in Python

After collecting data, the model for predicting the performance of football teams will be designed using machine learning algorithms and neural networks. This is a very important step in the thesis because if the presented model is not accurate, the correct results will not be obtained. As mentioned the target variable in this thesis is average points per game. Various prediction models were built based on different machine learning algorithms and neural networks methods, and finally the best model that provides more accurate outcomes was selected among them.

3.2.1 Importing the necessary libraries

Seven predictive models based on machine learning algorithms has been designed in Jupyter Notebook. The first step of programming in Python begins with importing the necessary libraries in Python. Some libraries used in the methods include Pandas, Numpy, Matplotlib, and Sklearn. Pyplot, Model_selection, Ensemble and Metrics are some modules which were used in the programming. Such functions used are Train_test_split, Mean_absolute_error, Mean_squared_error and R2_score

and the classes of RandomForestRegressor, GradientBoostingRegressor, SVR, Sequential, Adam, Dense, LinearRegression, DecisionTreeRegressor and XGBRegressor were imported.

3.2.2 Loading historical data

After calling the required Python libraries, the database must be loaded. The name of the database that contains information about the last 7 Serie A football leagues is Historical_Data.csv. After loading the data and calling them from the csv file, the data was cleaned. In this way, the columns related to two variables, Team and Season, were removed because they are not useful for modeling. Then points (.) were replaced by commas (,) with throughout the data frame and convert all values to float.

3.2.3 Train the historical data and evaluate of the predictive models

At the beginning of this stage, the data is divided into features and the target variable which is Average Points. Then 80% of the features are considered for training and 20% for testing. In this way, the model is trained and built. Evaluation of the models has been done on test sets. Then Mean Absolute Error, Mean Squared Error and R-Squared were calculated which shows the accuracy and efficiency of the model.

3.2.4 Validate and create the predictive models

For validation of the models in this thesis, the K-fold cross-validation method was used. In this method, the data is divided into 5 parts, means K is equal to 5 in implementing K-fold cross-validation. After the evaluation, it may be necessary to optimize the model or adjust the parameters. Finally a precise model with a high accuracy is presented to predict future performance of Italian Serie A football teams, ensuring reliable predictions for stakeholders in the football community, thereby facilitating informed decision-making. So the chosen predictive model was used to present champions of current league (2023-24) and also it shows the league ranking by scores which the football clubs will gain. Actually the predictive model forecasted the gained points by each team.

3.2.5 Analyze the predictive models

After creating the predictive models in Python, the models has been compared and analyzed. Permutation importance for the seven models was retrieved. It defines which feature has the greatest impact on the target variable. Some suitable graphs and plots like scatter plots and line charts of actual vs predicted values were designed and presented.

4 Implementation models programming and analytical comparison of models

In this section, a more comprehensive review of the codes is discussed and the outputs of the programming execution are presented. Also differences between programming of the models will be discussed. Finally the practical application of these prediction models in the real world will be investigated.

4.1 Python programming for the predictive models

In the following sections, the programming of all seven models in Jupyter Notebook will presented and discussed. The seven predictive models are XGBoost, SVM, GBM, Neural Networks, Decision Tree, Random Forest and Linear Regression.

4.1.1 Extreme Gradient Boosting

The Program code 1 shows the implementation of necessary libraries in the predictive model XGBoost.

Program code 1 Python programming of implementation of libraries, modules and classes in XGB

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, cross_val_score, KFold
from xgboost import XGBRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.inspection import permutation_importance
```

After implementation the necessary libraries, module and classes, the data has been spliced into the train and test data. Program code 2 shows a part of Python programming of XGBoost algorithm related to reading the csv file and splitting the data into train and test data.

Program code 2 Reading and splitting the historical data in XGB model

```
# Load the data
data = pd.read_csv("Historical_Data.csv", delimiter=';')

# Drop the 'Season' and 'Team' columns as they are not required for the analysis
data = data.drop(columns=['Season', 'Team'])

# Replace commas with periods throughout the DataFrame and convert all values to float
data = data.replace(',', '.', regex=True)
data = data.astype(float)

# Split the data into features and the target variable
X = data.drop(columns=['AveragePoints'])  # Features
y = data['AveragePoints']  # Target variable

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Program code 3 shows Python programming of create and train of XGB method. In the case of scaling, there is no need for scaling in tree-based models like XGBoost.

Program code 3 Create and train XGB model in Jupyter Notebook

```
# Create and train the XGBoost model
# No need for scaling in tree-based models
model = XGBRegressor(objective='reg:squarederror', n_estimators=100, learning_rate=0.1, random_state=42)
model.fit(X_train, y_train)
```

As can be seen in the Program code 4 the evaluation of the model XGBoost has been done in Python. The criteria of the evaluation on the test set is Mean Absolute Error, Mean Squared Error and R-Squared.

Program code 4 Evaluation of XGB on the test set

```
# Evaluate the model on the test set
y_pred = model.predict(X_test)
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Absolute Error on Test set: {mae}')
print(f'Mean Squared Error on Test set: {mse}')
print(f'R-squared on Test set: {r2}')
```

Program code 5 presents that the cross-validation K-Fold has been performed to validate the model XGBoost. The value of K has been taken 5. Also the criteria of evaluation for that was done.

Program code 5 Python programming of XGB model validate

```
# Perform K-Fold Cross-Validation
kf = KFold(n_splits=5, shuffle=True, random_state=42)

# Cross-Validated R-squared
cv_r2 = cross_val_score(model, X, y, cv=kf, scoring='r2')
print(f'Cross-Validated R-squared: {np.mean(cv_r2)}')

# Cross-Validated Mean Squared Error
cv_mse = cross_val_score(model, X, y, cv=kf, scoring='neg_mean_squared_error')
print(f'Cross-Validated Mean Squared Error: {-np.mean(cv_mse)}')

# Cross-Validated Mean Absolute Error
cv_mae = cross_val_score(model, X, y, cv=kf, scoring='neg_mean_absolute_error')
print(f'Cross-Validated Mean Absolute Error: {-np.mean(cv_mae)}')
```

Now the predictive model XGB has been created through the past data. The predictive model is ready to predict the performance of football teams in the current season (2023-24). According to the Program code 6 , programming of create the prediction model XGB to define the champion of the league and also teams ranking for the current season was done.

Program code 6 Creating prediction model XGB for predict the champion and standings of the current league

```
# Load current season data
current_season_data = pd.read_csv("Current_Season_Data.csv", delimiter=';')

# Drop the 'Season' and 'Team' columns as they are not required for the analysis
current_season_data = current_season_data.drop(columns=['Season', 'Team'])

# Replace commas with periods (decimal point) throughout the DataFrame and convert all values to float
current_season_data = current_season_data.replace(',', '.', regex=True)
current_season_data = current_season_data.astype(float)

# Predict using the trained model
X_current = current_season_data  # No need to drop columns as X_current should include all features
current_season_data['PredictedAveragePoints'] = model.predict(X_current)

# Load current season data with only the 'Team' column
current_season_teams = pd.read_csv("Current_Season_Data.csv", delimiter=';', usecols=['Team'])

# Add the 'Team' column to the current_season_data DataFrame
current_season_data['Team'] = current_season_teams['Team']

# Sort teams based on the predicted points
sorted_teams = current_season_data.sort_values(by='PredictedAveragePoints', ascending=False)

# Display the sorted list of teams with predicted points
print(sorted_teams[['Team', 'PredictedAveragePoints']])

# Identify the predicted champion
predicted_champion = sorted_teams.iloc[0]['Team']
print(f"\nThe predicted champion for the 2023-2024 season is: {predicted_champion}")
```

4.1.2 Support Vector Machine

According to Program code 7, necessary libraries has been implemented for the predictive model SVM.

Program code 7 Python programming of implementation of libraries, modules and classes in SVM

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, cross_val_score, KFold
from sklearn.svm import SVR
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.inspection import permutation_importance
```

After implementation necessary libraries, module and classes, the data has been spliced into the train and test data. Program code 8 shows Python programming of SVM method related to reading the csv file of past data and then splitting the data into train and test data.

Program code 8 Reading and splitting the historical data in SVM model

```
# Load the data
data = pd.read_csv("Historical_Data.csv", delimiter=';')

# Drop the 'Season' and 'Team' columns as they are not required for the analysis
data = data.drop(columns=['Season', 'Team'])

# Replace commas with periods throughout the DataFrame and convert all values to float
data = data.replace(',', '.', regex=True)
data = data.astype(float)

# Split the data into features and the target variable
X = data.drop(columns=['AveragePoints'])  # Features
y = data['AveragePoints']  # Target variable

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Program code 9 shows Python programming of create, scale and train of SVM method.

Program code 9 Create and train SVM model in Jupyter Notebook

```
# Create and train the SVM model using a pipeline for automatic standardization
model = make_pipeline(StandardScaler(), SVR(C=1.0, epsilon=0.2))
model.fit(X_train, y_train)
```

As can be seen in the Program code 10, the evaluation of the model SVM has been done in Python. The criteria of the evaluation on the test set is Mean Absolute Error, Mean Squared Error and R-Squared.

Program code 10 Evaluation of SVM on the test set

```
# Evaluate the model on the test set
y_pred = model.predict(X_test)
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Absolute Error on Test set: {mae}')
print(f'Mean Squared Error on Test set: {mse}')
print(f'R-squared on Test set: {r2}')
```

Program code 11 presents that the cross-validation K-Fold has been done to validate the model SVM. The value of K was taken 5. Also the precision criteria of evaluation for that was done.

Program code 11 Python programming of SVM model for validation

```
# Perform K-Fold Cross-Validation
kf = KFold(n_splits=5, shuffle=True, random_state=42)

# Cross-Validated R-squared
cv_r2 = cross_val_score(model, X, y, cv=kf, scoring='r2')
print(f'Cross-Validated R-squared: {np.mean(cv_r2)}')

# Cross-Validated Mean Squared Error
cv_mse = cross_val_score(model, X, y, cv=kf, scoring='neg_mean_squared_error')
print(f'Cross-Validated Mean Squared Error: {-np.mean(cv_mse)}')

# Cross-Validated Mean Absolute Error
cv_mae = cross_val_score(model, X, y, cv=kf, scoring='neg_mean_absolute_error')
print(f'Cross-Validated Mean Absolute Error: {-np.mean(cv_mae)}')
```

The predictive model SVM has been performed through the historical data. Now the predictive model is ready to predict the performance of football teams in the current season (2023-24). According to the Program code 12, the programming of create the prediction model SVM was done to define the champion of the league and also teams ranking for the current season.

Program code 12 Creating prediction model SVM for predict the champion and standings of the current league

```
# Load current season data
current_season_data = pd.read_csv("Current_Season_Data.csv", delimiter=';')

# Drop the 'Season' and 'Team' columns as they are not required for the analysis
current_season_data = current_season_data.drop(columns=['Season', 'Team'])

# Replace commas with periods (decimal point) throughout the DataFrame and convert all values to float
current_season_data = current_season_data.replace(',', '.', regex=True)
current_season_data = current_season_data.astype(float)

# Predict using the trained model
X_current = current_season_data  # No need to drop columns as X_current should include all features
current_season_data['PredictedAveragePoints'] = model.predict(X_current)

# Load current season data with only the 'Team' column
current_season_teams = pd.read_csv("Current_Season_Data.csv", delimiter=';', usecols=['Team'])

# Add the 'Team' column to the current_season_data DataFrame
current_season_data['Team'] = current_season_teams['Team']

# Sort teams based on the predicted points
sorted_teams = current_season_data.sort_values(by='PredictedAveragePoints', ascending=False)

# Display the sorted list of teams with predicted points
print(sorted_teams[['Team', 'PredictedAveragePoints']])

# Identify the predicted champion
predicted_champion = sorted_teams.iloc[0]['Team']
print(f"\nThe predicted champion for the 2023-2024 season is: {predicted_champion}")
```

4.1.3 Gradient Boosting Machines

Program code 13 presents the importing of the necessary libraries for the predictive model GBM.

Program code 13 Python programming of implementation of libraries, modules and classes in GBM

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, cross_val_score, KFold
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.inspection import permutation_importance
```

After implementation the necessary libraries, module and classes, the data has been spliced into the train and test data. Program code 14 presents Python programming of GBM related to reading the csv file of the past seasons data and then splitting them into train and test data.

Program code 14 Reading and splitting the historical data in GBM model

```
# Load the data
data = pd.read_csv("Historical_Data.csv", delimiter=';')

# Drop the 'Season' and 'Team' columns as they are not required for the analysis
data = data.drop(columns=['Season', 'Team'])

# Replace commas with periods throughout the DataFrame and convert all values to float
data = data.replace(',', '.', regex=True)
data = data.astype(float)

# Split the data into features and the target variable
X = data.drop(columns=['AveragePoints'])  # Features
y = data['AveragePoints']  # Target variable

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Python programming of create, train and evaluation of GBM model is presented in Program code 15. The criteria of the evaluation on the test set is Mean Absolute Error, Mean Squared Error and R-Squared.

Program code 15 Create, train and evaluation of GBM model in Jupyter Notebook

```
# Create and train the Gradient Boosting model
model = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1, random_state=42)
model.fit(X_train, y_train)

# Evaluate the model on the test set
y_pred = model.predict(X_test)
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Absolute Error on Test set: {mae}')
print(f'Mean Squared Error on Test set: {mse}')
print(f'R-squared on Test set: {r2}')
```

According to the Program code 16, the cross-validation 5-Fold has been done to validate the model GBM. Also the evaluation for that was done.

Program code 16 Python programming of GBM model for validation

```
# Perform K-Fold Cross-Validation
kf = KFold(n_splits=5, shuffle=True, random_state=42)

# Cross-Validated R-squared
cv_r2 = cross_val_score(model, X, y, cv=kf, scoring='r2')
print(f'Cross-Validated R-squared: {np.mean(cv_r2)}')

# Cross-Validated Mean Squared Error
cv_mse = cross_val_score(model, X, y, cv=kf, scoring='neg_mean_squared_error')
print(f'Cross-Validated Mean Squared Error: {-np.mean(cv_mse)}')

# Cross-Validated Mean Absolute Error
cv_mae = cross_val_score(model, X, y, cv=kf, scoring='neg_mean_absolute_error')
print(f'Cross-Validated Mean Absolute Error: {-np.mean(cv_mae)}')
```

The predictive model GBM has been implemented through the historical data. Now the predictive model is ready to predict the performance of football teams in the current league (2023-24). According to the Program code 17, the programming of create the prediction model GBM was done to define the champion of the league and also teams standings for the current season.

Program code 17 Creating prediction model GBM for predict the champion and standings of the current league

```
# Load current season data
current_season_data = pd.read_csv("Current_Season_Data.csv", delimiter=';')

# Drop the 'Season' and 'Team' columns as they are not required for the analysis
current_season_data = current_season_data.drop(columns=['Season', 'Team'])

# Replace commas with periods (decimal point) throughout the DataFrame and convert all values to float
current_season_data = current_season_data.replace(',', '.', regex=True)
current_season_data = current_season_data.astype(float)

# Predict using the trained model
X_current = current_season_data  # No need to drop columns as X_current should include all features
current_season_data['PredictedAveragePoints'] = model.predict(X_current)

# Load current season data with only the 'Team' column
current_season_teams = pd.read_csv("Current_Season_Data.csv", delimiter=';', usecols=['Team'])

# Add the 'Team' column to the current_season_data DataFrame
current_season_data['Team'] = current_season_teams['Team']

# Sort teams based on the predicted points
sorted_teams = current_season_data.sort_values(by='PredictedAveragePoints', ascending=False)

# Display the sorted list of teams with predicted points
print(sorted_teams[['Team', 'PredictedAveragePoints']])

# Identify the predicted champion
predicted_champion = sorted_teams.iloc[0]['Team']
print(f"\nThe predicted champion for the 2023-2024 season is: {predicted_champion}")
```

4.1.4 Neural Networks

Program code 18 presents the importing of the necessary libraries for the predictive model Neural Networks.

Program code 18 Python programming of implementation of libraries, modules and classes in NN

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import Adam
```

After implementation the necessary libraries, module and classes, the data has been spliced into the train and test data. Program code 19 presents Python programming of NN related to reading the csv file of the past seasons data scaling and splitting them into train and test data.

Program code 19 Reading, scaling and splitting the historical data in NN model

```
# Load the data
data = pd.read_csv("Historical_Data.csv", delimiter=';')

# Drop the 'Season' and 'Team' columns as they are not required for the analysis
data = data.drop(columns=['Season', 'Team'])

# Replace commas with periods throughout the DataFrame and convert all values to float
data = data.replace(',', '.', regex=True)
data = data.astype(float)

# Split the data into features and the target variable
X = data.drop(columns=['AveragePoints'])  # Features
y = data['AveragePoints']  # Target variable

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Scale data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Python programming of build, train and evaluation of NN model is presented in Program code 20. The criteria of the evaluation on the test set is Mean Absolute Error, Mean Squared Error and R-Squared.

Program code 20 Create, train and evaluation of NN model in Jupyter Notebook

```
# Build the neural network model
model = Sequential([
    Dense(128, activation='relu', input_shape=(X_train.shape[1],)),
    Dense(64, activation='relu'),
    Dense(32, activation='relu'),
    Dense(1)
])

model.compile(optimizer=Adam(), loss='mse')

# Train the model
model.fit(X_train, y_train, epochs=100, batch_size=10, validation_split=0.2, verbose=1)

# Predict on the test set
y_pred = model.predict(X_test).flatten()

# Evaluate the model
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Absolute Error on Test set: {mae}')
print(f'Mean Squared Error on Test set: {mse}')
print(f'R-squared on Test set: {r2}')
```

According to the Program code 21, the cross-validation 5-Fold has been done to validate the model NN. Also the evaluation for that was done.

Program code 21 Python programming of NN model for validation

```
# K-fold Cross Validation model evaluation
fold_no = 1
maes, mses, r2s = [], [], []
for train, test in kfold.split(X, y):
    # Scaling
    X_train, X_test = X.iloc[train], X.iloc[test]
    y_train, y_test = y.iloc[train], y.iloc[test]
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)
    # Build the neural network model
    model = Sequential([
        Dense(128, activation='relu', input_shape=(X_train.shape[1],)),
        Dense(64, activation='relu'),
        Dense(32, activation='relu'),
        Dense(1)
    ])
    model.compile(optimizer=Adam(), loss='mse')
    # Train the model
    print(f'Training for fold {fold_no}...')
    model.fit(X_train, y_train, epochs=100, batch_size=10, validation_split=0.2, verbose=1)
    # Predict on the test set
    y_pred = model.predict(X_test).flatten()
    # Evaluate the model
    mae = mean_absolute_error(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    maes.append(mae)
    mses.append(mse)
    r2s.append(r2)
    print(f'Mean Absolute Error for fold {fold_no}: {mae}')
    print(f'Mean Squared Error for fold {fold_no}: {mse}')
    print(f'R-squared for fold {fold_no}: {r2}')

    fold_no += 1
```

The predictive model Neural Networks has been implemented through the historical data. Now the predictive model is ready to predict the performance of football teams in the current league (2023-24). According to the Program code 22, the programming of create the prediction model NN was done to define the champion of the league and also teams standings for the current season.

Program code 22 Creating prediction model NN for predict the champion and standings of the current league

```
# Load current season data
current_season_data = pd.read_csv("Current_Season_Data.csv", delimiter=';')

# Drop the 'Season' and 'Team' columns as they are not required for the analysis
current_season_data = current_season_data.drop(columns=['Season', 'Team'])

# Replace commas with periods (decimal point) throughout the DataFrame and convert all values to float
current_season_data = current_season_data.replace(',', '.', regex=True)
current_season_data = current_season_data.astype(float)

# Scale the current season features
X_current_scaled = scaler.transform(current_season_data)

# Predict using the trained model
current_season_data['PredictedAveragePoints'] = model.predict(X_current_scaled).flatten()

# Load current season data with only the 'Team' column
current_season_teams = pd.read_csv("Current_Season_Data.csv", delimiter=';', usecols=['Team'])

# Add the 'Team' column to the current_season_data DataFrame
current_season_data['Team'] = current_season_teams['Team']

# Sort teams based on the predicted points
sorted_teams = current_season_data.sort_values(by='PredictedAveragePoints', ascending=False)

# Display the sorted list of teams with predicted points
print(sorted_teams[['Team', 'PredictedAveragePoints']])

# Identify the predicted champion
predicted_champion = sorted_teams.iloc[0]['Team']
print(f"\nThe predicted champion for the 2023-2024 season is: {predicted_champion}")
```

4.1.5 Decision Tree

As can be seen in Program code 23, the necessary libraries for implementing the predictive model Decision Tree were imported in Jupyter Notebook.

Program code 23 Python programming of implementation of libraries, modules and classes in DT

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, cross_val_score, KFold
from sklearn.tree import DecisionTreeRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.inspection import permutation_importance
```

Then loading the csv file has been done and cleaning the historical data was performed. Then splitting the data to train and test samples was done. Finally the Decision Tree model has been created. Program code 24 shows the programming of the stages of loading, splitting data and training the model.

Program code 24 Python programming of loading, splitting and training in predictive model DT

```
# Load the data
data = pd.read_csv("Historical_Data.csv", delimiter=';')

# Drop the 'Season' and 'Team' columns as they are not required for the analysis
data = data.drop(columns=['Season', 'Team'])

# Replace commas with periods (decimal point) throughout the DataFrame and convert all values to float
data = data.replace(',', '.', regex=True)
data = data.astype(float)

# Split the data into features and the target variable
X = data.drop(columns=['AveragePoints'])
y = data['AveragePoints']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and train the Decision Tree model
model = DecisionTreeRegressor(random_state=42)
model.fit(X_train, y_train)
```

Evaluation of the predictive model DT on the test set and validate the model can be seen in the Program code 25.

Program code 25 Python programming of evaluation and Validation of Decision Tree

```
# Evaluate the model on the test set
y_pred = model.predict(X_test)
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Absolute Error on Test set: {mae}')
print(f'Mean Squared Error on Test set: {mse}')
print(f'R-squared on Test set: {r2}')

# Perform K-Fold Cross-Validation
kf = KFold(n_splits=5, shuffle=True, random_state=42)
cv_r2 = cross_val_score(model, X, y, cv=kf, scoring='r2')
cv_mse = cross_val_score(model, X, y, cv=kf, scoring='neg_mean_squared_error')
cv_mae = cross_val_score(model, X, y, cv=kf, scoring='neg_mean_absolute_error')

print(f'Cross-Validated R-squared: {np.mean(cv_r2)}')
print(f'Cross-Validated Mean Squared Error: {-np.mean(cv_mse)}')
print(f'Cross-Validated Mean Absolute Error: {-np.mean(cv_mae)}')
```

The predictive model Decision Tree has been implemented through the historical data. Now the predictive model is ready to predict the performance of football teams in the current league (2023-

24). According to the Program code 26, the programming of creating the prediction model DT was done to define the champion of the Serie A football league and also teams ranking for the current season.

Program code 26 Python programming of creating prediction model DT for predict the champion and standings of the current league

```
# Load current season data
current_season_data = pd.read_csv("Current_Season_Data.csv", delimiter=';')

# Drop the 'Season' and 'Team' columns as they are not required for the analysis
current_season_data = current_season_data.drop(columns=['Season', 'Team'])

# Replace commas with periods (decimal point) throughout the DataFrame and convert all values to float
current_season_data = current_season_data.replace(',', '.', regex=True)
current_season_data = current_season_data.astype(float)

# Predict using the trained model
X_current = current_season_data
current_season_data['PredictedAveragePoints'] = model.predict(X_current)

# Load current season data with only the 'Team' column
current_season_teams = pd.read_csv("Current_Season_Data.csv", delimiter=';', usecols=['Team'])

# Add the 'Team' column to the current_season_data DataFrame
current_season_data['Team'] = current_season_teams['Team']

# Sort teams based on the predicted points
sorted_teams = current_season_data.sort_values(by='PredictedAveragePoints', ascending=False)

# Display the sorted list of teams with predicted points
print(sorted_teams[['Team', 'PredictedAveragePoints']])

# Identify the predicted champion
predicted_champion = sorted_teams.iloc[0]['Team']
print(f"\nThe predicted champion for the 2023-2024 season is: {predicted_champion}")
```

4.1.6 Random Forest

Inserting the necessary libraries was done in Jupyter Notebook. The imported libraries, modules and classes are presented in Program code 27.

Program code 27 Imported libraries, modules and classes in Jupyter Notebook for RF predictive model

```
import pandas as pd
from sklearn.svm import SVR
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

The past data which is collected in the csv file of Historical_Data has been loaded. After processing the data, splitting and training of Random Forest model was done in Python. Stages of reading, splitting and training of RF is shown in Program code 28.

Program code 28 Python programming of Load, split and train of RF in Jupyter Notebook

```
# Load the data
data = pd.read_csv("Historical_Data.csv", delimiter=';')

# Drop 'Season' and 'Team' columns
data = data.drop(columns=['Season', 'Team'])

# Replace commas with periods in the entire DataFrame
data = data.replace(',', '.', regex=True)

# Convert all values to float
data = data.astype(float)

# Split data into features and target variable
X = data.drop(columns=['AveragePoints'])  # Features
y = data['AveragePoints']  # Target variable

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Model training
model = RandomForestRegressor(random_state=42)
model.fit(X_train, y_train)
```

Validate and evaluate of Random Forest model on the test set is presented in Program code 29. The evaluation criteria in validate sets also was presented.

Program code 29 Python programming of evaluation and validation for RF model

```
# Model evaluation on test set
y_pred = model.predict(X_test)
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f'Mean Absolute Error on Test set: {mae}')
print(f'Mean Squared Error on Test set: {mse}')
print(f'R-squared on Test set: {r2}')

# Implementing K-fold cross-validation
kf = KFold(n_splits=5, shuffle=True, random_state=42)
cv_mse = cross_val_score(model, X, y, cv=kf, scoring='neg_mean_squared_error')
cv_mae = cross_val_score(model, X, y, cv=kf, scoring='neg_mean_absolute_error')
cv_r2 = cross_val_score(model, X, y, cv=kf, scoring='r2')

print(f'Cross-Validated Mean Squared Error: {-np.mean(cv_mse)}')
print(f'Cross-Validated Mean Absolute Error: {-np.mean(cv_mae)}')
print(f'Cross-Validated R-squared: {np.mean(cv_r2)}')
```

The Current_Season_Data.csv file has been loaded and performing the predictive model for forecasting the champion and rankings of the current season of Serie A football league is presented in Program code 30.

Program code 30 Python programming of creating prediction model RF for predict the champion and standings of the current league

```
# Load current season data
current_season_data = pd.read_csv("Current_Season_Data.csv", delimiter=';')

# Drop the 'Season' and 'Team' columns as they are not required for the analysis
current_season_data = current_season_data.drop(columns=['Season', 'Team'])

# Replace commas with periods (decimal point) throughout the DataFrame and convert all values to float
current_season_data = current_season_data.replace(',', '.', regex=True)
current_season_data = current_season_data.astype(float)

# Predict using the trained model
X_current = current_season_data  # No need to drop columns as X_current should include all features
current_season_data['PredictedAveragePoints'] = model.predict(X_current)

# Load current season data with only the 'Team' column
current_season_teams = pd.read_csv("Current_Season_Data.csv", delimiter=';', usecols=['Team'])

# Add the 'Team' column to the current_season_data DataFrame
current_season_data['Team'] = current_season_teams['Team']

# Sort teams based on the predicted points
sorted_teams = current_season_data.sort_values(by='PredictedAveragePoints', ascending=False)

# Display the sorted list of teams with predicted points
print(sorted_teams[['Team', 'PredictedAveragePoints']])

# Identify the predicted champion
predicted_champion = sorted_teams.iloc[0]['Team']
print(f"\nThe predicted champion for the 2023-2024 season is: {predicted_champion}")
```

4.1.7 Linear Regression

Necessary libraries were implemented in Python for the predictive model of Linear Regression. The past information has been loaded and processing of data was done. All stages of reading and processing in Linear Regression are shown in Program code 31.

Program code 31 Python programming of implementation of the libraries and reading the past data in LR

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, cross_val_score, KFold
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.inspection import permutation_importance
import matplotlib.pyplot as plt

# Load the data
data = pd.read_csv("Historical_Data.csv", delimiter=';')

# Drop the 'Season' and 'Team' columns as they are not required for the analysis
data = data.drop(columns=['Season', 'Team'])

# Replace commas with periods throughout the DataFrame and convert all values to float
data = data.replace(',', '.', regex=True)
data = data.astype(float)
```

Splitting the data, scaling, creating and evaluation of Linear Regression model are presented in Program code 32.

Program code 32 Python programming of split, train and evaluation of LR model in Jupyter Notebook

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and train the Linear Regression model using a pipeline for automatic standardization
model = make_pipeline(StandardScaler(), LinearRegression())
model.fit(X_train, y_train)

# Evaluate the model on the test set
y_pred = model.predict(X_test)
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Absolute Error: {mae}')
print(f'Mean Squared Error on Test set: {mse}')
print(f'R-squared on Test set: {r2}\n')
```

According to Program code 33, validation of Linear Regression model was done in Python. The criteria of evaluation are presented.

Program code 33 Python programming of validate of LR model

```
# Perform K-Fold Cross-Validation
kf = KFold(n_splits=5, shuffle=True, random_state=42)

# Cross-Validated R-squared
cv_r2 = cross_val_score(model, X, y, cv=kf, scoring='r2')
print(f'Cross-Validated R-squared: {np.mean(cv_r2)}')

# Cross-Validated Mean Squared Error
cv_mse = cross_val_score(model, X, y, cv=kf, scoring='neg_mean_squared_error')
print(f'Cross-Validated Mean Squared Error: {-np.mean(cv_mse)}')

# Cross-Validated Mean Absolute Error
cv_mae = cross_val_score(model, X, y, cv=kf, scoring='neg_mean_absolute_error')
print(f'Cross-Validated Mean Absolute Error: {-np.mean(cv_mae)}\n')
```

Also the common criteria of evaluation which is MAE and MSE can be retrieved by programming, as can be seen in Program code 33.

The Current_Season_Data.csv file has been loaded and performing the predictive model for forecasting the champion and rankings of the current season of Serie A football league is presented in Program code 34.

Program code 34 Python programming of creating prediction model LR for predict the champion and standings of the current league

```
# Load current season data
current_season_data = pd.read_csv("Current_Season_Data.csv", delimiter=';')

# Drop the 'Season' and 'Team' columns as they are not required for the analysis
current_season_data = current_season_data.drop(columns=['Season', 'Team'])

# Replace commas with periods (decimal point) throughout the DataFrame and convert all values to float
current_season_data = current_season_data.replace(',', '.', regex=True)
current_season_data = current_season_data.astype(float)

# Predict using the trained model
X_current = current_season_data
current_season_data['PredictedAveragePoints'] = model.predict(X_current)

# Load current season data with only the 'Team' column
current_season_teams = pd.read_csv("Current_Season_Data.csv", delimiter=';', usecols=['Team'])

# Add the 'Team' column to the current_season_data DataFrame
current_season_data['Team'] = current_season_teams['Team']

# Sort teams based on the predicted points
sorted_teams = current_season_data.sort_values(by='PredictedAveragePoints', ascending=False)

# Display the sorted list of teams with predicted points
print(sorted_teams[['Team', 'PredictedAveragePoints']])

# Identify the predicted champion
predicted_champion = sorted_teams.iloc[0]['Team']
print(f"\nThe predicted champion for the 2023-2024 season is: {predicted_champion}")
```

4.2 Real-world application and New knowledge

Presenting the model for predicting the outcomes of football matches and forecasting the performance of football teams in the real world also has an important and effective application. Football clubs can use these predictive models to better manage the conditions of the club so that they can get better results. By using this model, they can find out what factors affect better results. Betting companies are one of the beneficiaries of these prediction models. They can improve the coefficients considered for the winner of the league and the champion of the ahead league by using these prediction models. In this way, they can increase their users and attract more financial capital.

This thesis can add new things to the science of predicting performance of football teams and programming prediction models. Some of these items include identifying patterns and factors affecting the outcomes of football matches. Additionally, comparing prediction models based on machine learning and neural networks leads to the improvement of methods and algorithms.

Introducing new techniques to optimize prediction models and assessing the impact of neural networks and machine learning methods on the accuracy and efficiency of predicting of performance of football clubs. It is clear that the results of the investigations have crucial aspects. So it is determined that machine learning algorithms and neural networks have an important effect in increasing the accuracy of prediction models.

5 Analytical comparisons of the predictive models

In this section the analytical comparison between the predictive models will be investigated. The outputs of Python programming in Jupyter Notebook will be presented in some suitable figures and tables. Finally it will be discussed which models will present more accurate results.

5.1 Errors and accuracies

Errors and accuracies of the models on test sets has been calculated and were collected in Table 1. Table 2 presents results of cross-validate for the models. Figure 4 shows an example of programming output to determine errors and precisions for model XGBoost.

Figure 4 Outputs of errors and accuracies of XGBoost in Jupyter Notebook

```
Mean Absolute Error on Test set: 0.11470308898176469
Mean Squared Error on Test set: 0.019705041549863904
R-squared on Test set: 0.9271861731888686
Cross-Validated R-squared: 0.9149064026531555
Cross-Validated Mean Squared Error: 0.019088091653993926
Cross-Validated Mean Absolute Error: 0.10567788948501859
Predicted Average Points for Atalanta 2022-23 which is 1.526: 1.5267175436019897
```

Table 1 Coefficients of precisions on test sets of the predictive models

Model	Mean Absolute Error	Mean Squared Error	R-Squared
XGBoost	0.1147	0.0197	0.9272
SVM	0.1713	0.0442	0.8367
Random Forest	0.1026	0.0161	0.9406
Neural Networks	0.3429	0.1597	0.4097
GBM	0.0859	0.0130	0.9521
Decision Tree	0.1195	0.0232	0.9143
Linear Regression	0.0843	0.0132	0.9513

The model LR and GBM have the least Mean Absolute Errors and Mean Squared Errors. Neural Networks model has the most Mean Absolute Error and Mean Squared Error. LR and GBM model have the most R-Squared and Neural Networks method has the least R-Squared among the 7 models. According to Table 2 LR has the least cross-validate mean absolute error on the sets. LR, GBM and Random Forest have the least cross-validate Mean Squared Error. Logistic Regression

and Random Forest model has the most cross-validated R-squared and Neural Networks method has the least R-Squared among the 7 models.

Table 2 Coefficients of precisions on cross-validate of the predictive models

Model	Mean Absolute Error	Mean Squared Error	R-Squared
XGBoost	0.1057	0.0191	0.9149
SVM	0.1713	0.0465	0.7907
Random Forest	0.1024	0.0171	0.9228
Neural Networks	0.2451	0.0914	0.5847
GBM	0.1009	0.0171	0.9225
Decision Tree	0.1409	0.0326	0.8514
Linear Regression	0.0859	0.0122	0.9437

5.2 Prediction the champion of the league and teams ranking

An interesting comparison between the models is made based on the score of one of the teams in the 2022-23 season, which is presented in Table 3. In this way, the score predicted by the models has been compared with the actual score of the Atalanta team in the 2022-23 season which is 1.526. As a result from table 3, Decision Tree predicted the average points exactly right. Models of XGBoost and Neural Networks also predicted more accurate points against the other models.

Table 3 Predicted average points for Atalanta 2022-23 (actual point: 1.526)

Model	Predicted Average Points
XGBoost	1.5267175436019897
SVM	1.4682053661935504
Random Forest	1.5779999999999996
Neural Networks	1.5257861614227295
GBM	1.5213459382602936
Linear Regression	1.5610016007891614
Decision Tree	1.526

All predictive models have predicted that Inter will champion in current season of football Serie A league (2023-24). But in these predictive models, the rankings of teams is different. Table 4

(XGBoost), Table 5 (SVM), Table 6 (Random Forest), Table 7 (Neural Networks), Table 8 (GBM), Table 9 (Decision Tree) and Table 10 (Linear Regression) offers team rankings and their average points per game in different models.

Table 4 Comparison football clubs ranking in XGBoost

Team	PredictedAveragePoints
Inter	2.406089
Atalanta	1.741986
Milan	1.697171
Bologna	1.694577
Napoli	1.679417
Juventus	1.640942
Fiorentina	1.601727
Roma	1.564412
Lazio	1.516172
Torino	1.221387
Monza	1.172525
Genoa	1.140724
Lecce	1.087924
Verona	1.044414
Udinese	1.025113
Frosinone	1.006519
Sassuolo	0.996819
Caligari	0.981503
Empoli	0.975372
Salernitana	0.511681

As can be seen in the Table 5, as XGB, SVM method also predicted that Inter was championed.

Table 5 Football clubs ranking in SVM

Team	PredictedAveragePoints
Inter	1.935760
Napoli	1.783316
Atalanta	1.708233
Juventus	1.631431
Bologna	1.622650
Milan	1.593793
Fiorentina	1.493278
Roma	1.452127
Lazio	1.419273
Torino	1.407713
Sassuolo	1.315382
Frosinone	1.298340
Monza	1.224759
Caligari	1.200335
Genoa	1.146908
Empoli	1.094247
Verona	1.076337
Lecce	0.955604
Salernitana	0.943392
Udinese	0.935306

In the next page, prediction of team rankings and their average points per game by models RF, NN and GBM will be presented.

Table 6 Football clubs ranking in Random Forest

Team	PredictedAveragePoints
Inter	2.32236
Milan	1.79957
Atalanta	1.74619
Juventus	1.72747
Bologna	1.69442
Roma	1.68650
Napoli	1.64857
Fiorentina	1.59949
Lazio	1.58431
Torino	1.25737
Genoa	1.22457
Monza	1.21014
Lecce	1.05578
Verona	1.02647
Udinese	1.00234
Frosinone	0.99966
Empoli	0.95385
Caligari	0.95063
Sassuolo	0.90079
Salernitana	0.57393

Unlike the SVM model, which predicted the average points per game of the Inter team to be less than 2, the RF and NN models predicted the average points per game to be above 2.

Table 7 Football clubs ranking in Neural Networks

Team	PredictedAveragePoints
Inter	2.80737
Napoli	2.52346
Milan	2.33208
Roma	2.2059
Bologna	1.97284
Genoa	1.87478
Juventus	1.79573
Monza	1.67263
Fiorentina	1.61805
Atalanta	1.48948
Lazio	1.45689
Sassuolo	1.33165
Torino	1.28528
Verona	1.27895
Frosinone	1.17117
Caligari	1.084
Empoli	1.03458
Salernitana	0.986358
Udinese	0.833773
Lecce	0.654993

Model GBM is considered one of the most accurate models of this research.

Table 8 Football clubs ranking in GBM

Team	PredictedAveragePoints
Inter	2.350765
Juventus	1.793615
Bologna	1.782340
Milan	1.717869
Atalanta	1.708415
Napoli	1.704460
Roma	1.654708
Fiorentina	1.628669
Lazio	1.531629
Torino	1.220957
Genoa	1.168519
Monza	1.160887
Verona	1.073760
Lecce	1.073435
Udinese	1.055932
Frosinone	0.998413
Sassuolo	0.992382
Caligari	0.946570
Empoli	0.921588
Salernitana	0.560677

Table 9 Football clubs ranking in Decision Tree

Team	PredictedAveragePoints
Inter	2.400
Atalanta	1.816
Roma	1.737
Bologna	1.737
Juventus	1.711
Milan	1.632
Napoli	1.579
Lazio	1.579
Fiorentina	1.526
Torino	1.289
Monza	1.184
Genoa	1.184
Lecce	1.026
Empoli	1.026
Caligari	1.026
Udinese	0.947
Frosinone	0.789
Sassuolo	0.789
Verona	0.789
Salernitana	0.474

Model LR is among the models that predict more points for the champion team. This comparison is intended against other models.

Table 10 Football clubs ranking in Linear Regression

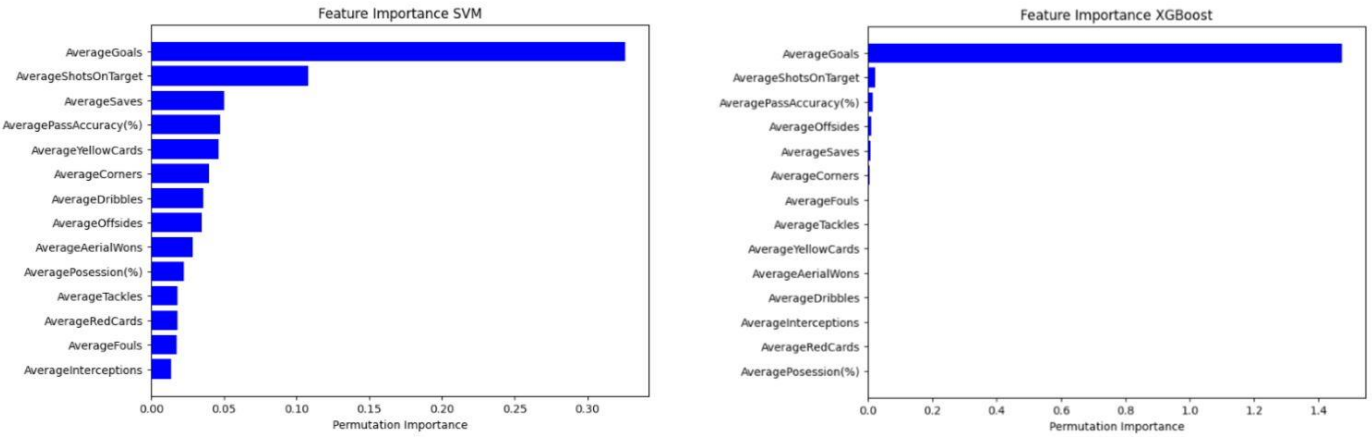
Team	PredictedAveragePoints
Inter	2.559645
Milan	1.940301
Atalanta	1.755112
Napoli	1.694168
Juventus	1.690848
Roma	1.661577
Bologna	1.660489
Fiorentina	1.439957
Lazio	1.430766
Torino	1.359498
Genoa	1.175592
Monza	1.125278
Verona	1.080804
Lecce	0.988354
Udinese	0.981007
Frosinone	0.918297
Caligari	0.915409
Sassuolo	0.861102
Empoli	0.761298
Salernitana	0.493744

5.3 Permutation importance

Feature permutation importance of predictive models has been done and the outcomes are presented in some graphs. Figure 5 shows feature importance for XGBoost and SVM. Figure 6 presents graphs of permutation importance of features in models Random Forest and Neural

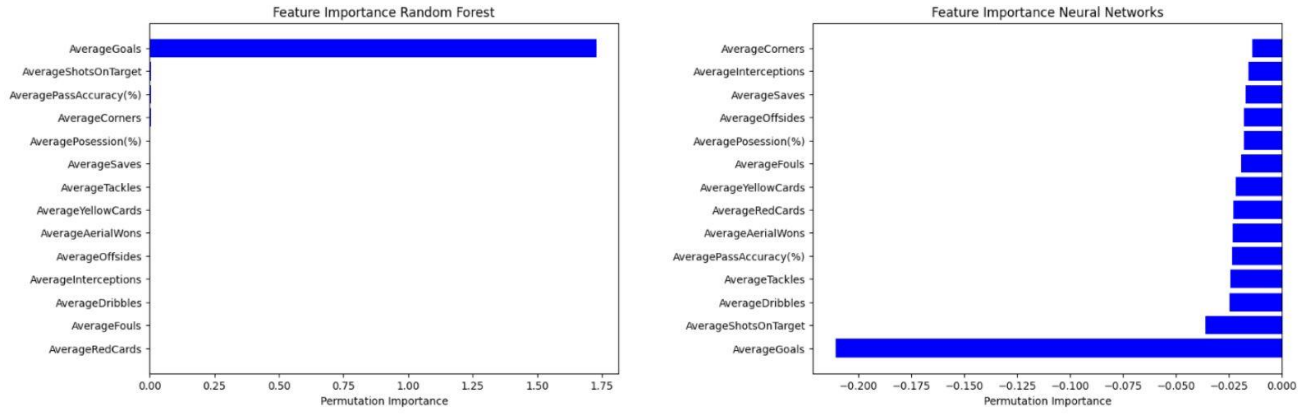
Networks. Graphs of feature importance for models GBM and Decision Tree are presented in Figure 7. Feature importance for Linear Regression is presented in Figure 8. In all models average goals per game is the most effective feature. The other features have less importance against the target variable. It can be said that the red card had the least impact among the features.

Figure 5 Feature importance in models SVM & XGBoost



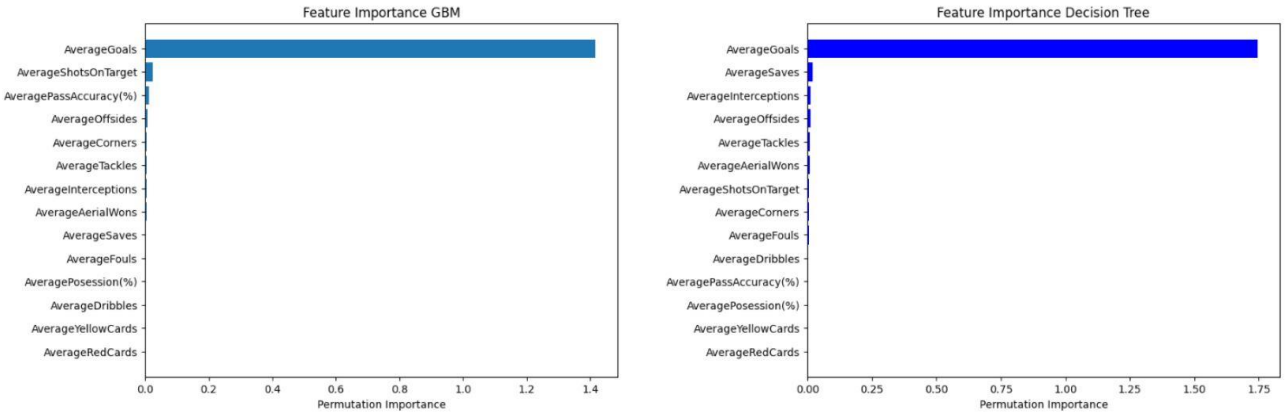
The direction of the graphs in the NN diagram is the opposite of the graphs related to the other models.

Figure 6 Feature importance in models Random Forest and Neural Networks



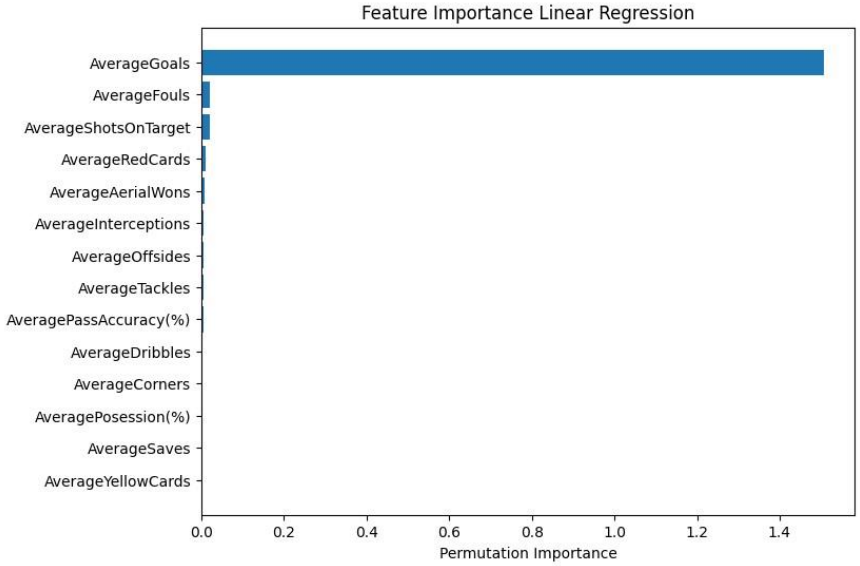
The comparison of the importance of variables for GBM, DT and LR models is presented on the next page. In all models average goals per game is the most effective feature. The other features have less importance against the target variable. It can be said that the red card had the least impact among the features.

Figure 7 Feature importance in models GBM and Decision Tree



It seems that one variable is common among all models as the most important variable.

Figure 8 Feature importance in Linear Regression

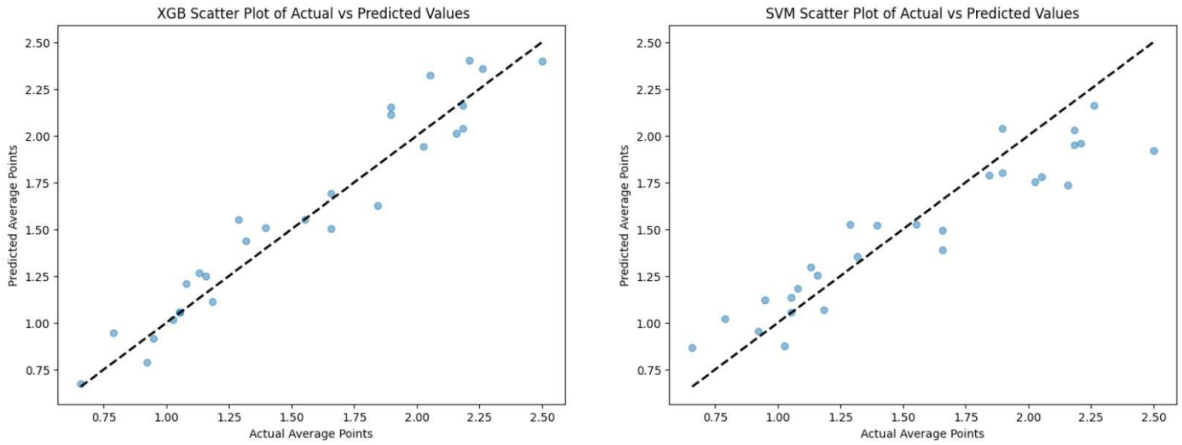


Charts and tables related to prediction models and variables used in them were presented. In the next section, the comparison of the values obtained as a result of the prediction models is presented.

5.4 Comparison of actual values and predicted values

One of the appropriate criteria to check the performance of a predictive model is to compare the actual values and the predicted values of the target variable. Figure 9 shows this comparison in SVM and XGBoost models. The scatter plots of the comparison for models Random Forest and Neural Networks are presented in Figure 10.

Figure 9 Scatter plot of Actual vs Predicted Values in SVM and XGBoost



Model NN is one of the most inaccurate models of this research.

Figure 10 Scatter plot of Actual vs Predicted Values in Random Forest and Neural Networks

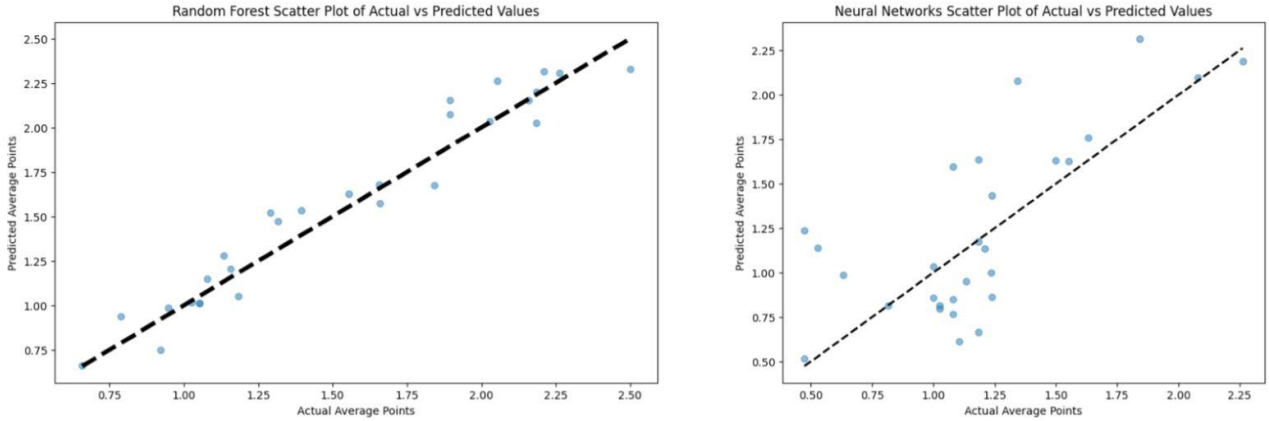


Figure 11 which is on the next page is the scatter plot of comparison between predicted values and actual values of average points in GBM and Decision Tree.

Figure 11 Scatter plot of Actual vs Predicted Values in GBM and Decision Tree

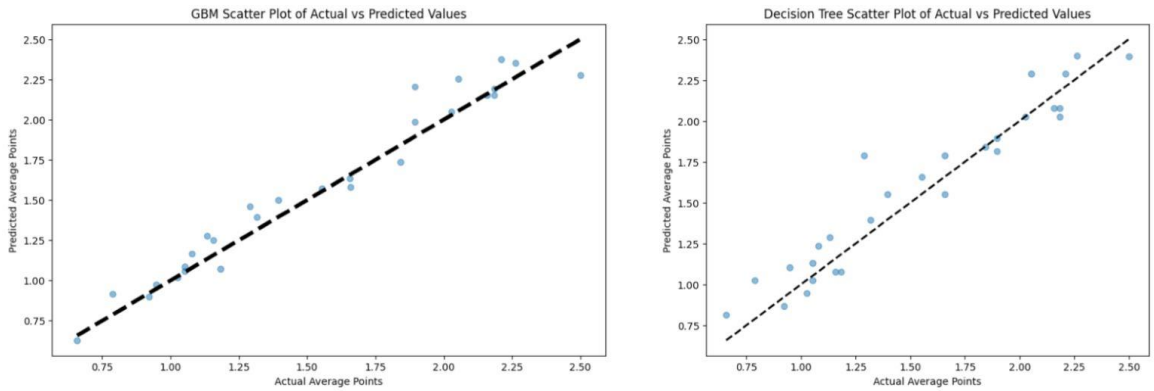
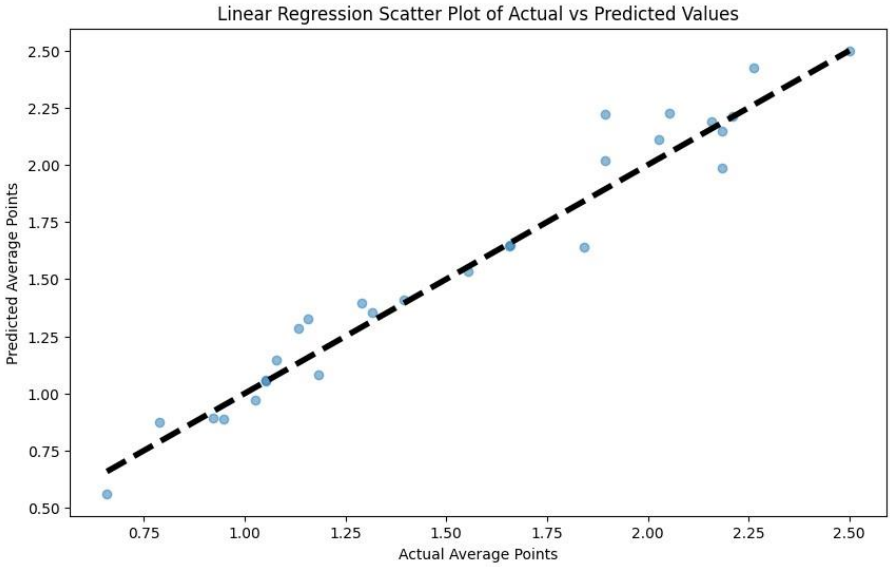


Figure 12 shows scatter plot of comparison between predicted values and actual values of average points in LR.

Figure 12 Scatter plot of Actual vs Predicted Values in LR



It seems that the Neural Networks diagram has an inappropriate dispersion, but in the rest of the diagrams, the values are close, especially in models LR, GBM and Random Forest.

5.5 Learning curves

A learning curve is a curve that shows the changes in training score and cross-validation score for different MSE and different training examples. The ideal state of these curves is that the MSE decreases with the increase of training examples. Figure 13 shows learning curves for SVM and XGBoost.

Figure 13 Learning curves for SVM and XGBoost

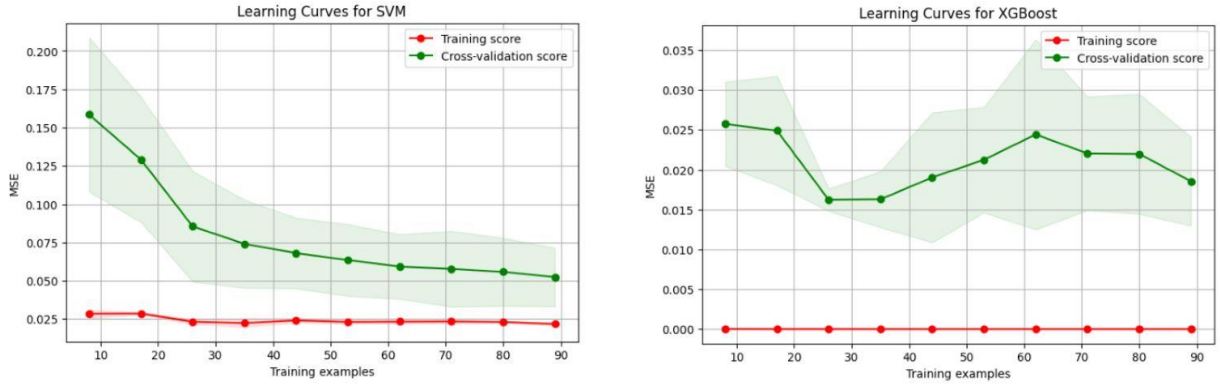
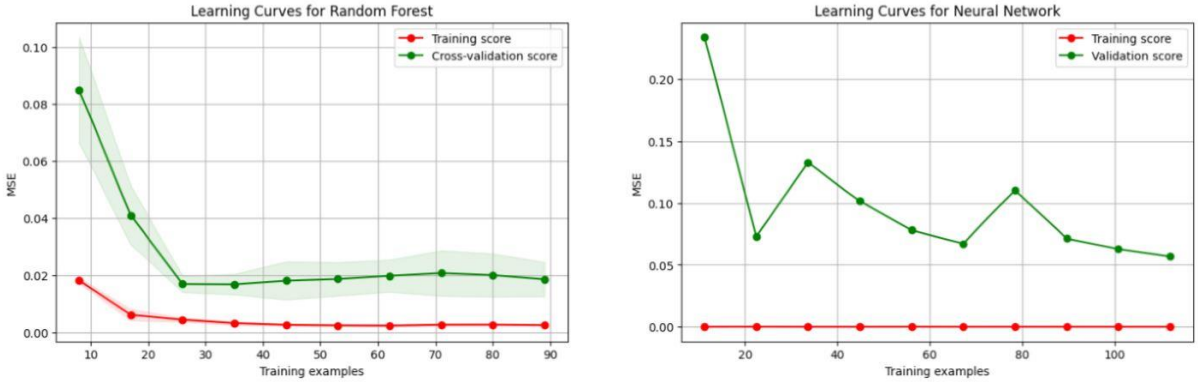


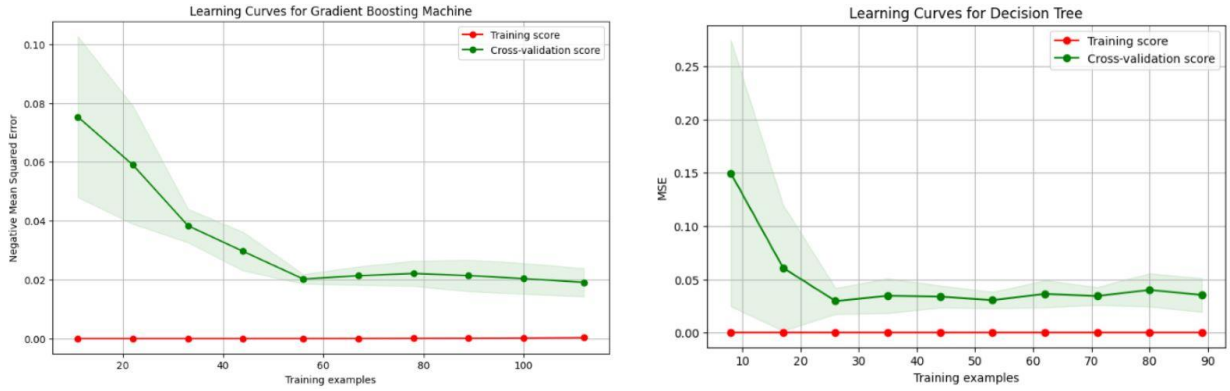
Figure 14 show learning curves for Random Forest and Neural Networks. It is worth mentioning that model NN is the worst model

Figure 14 Learning curves for Random Forest and Neural Networks



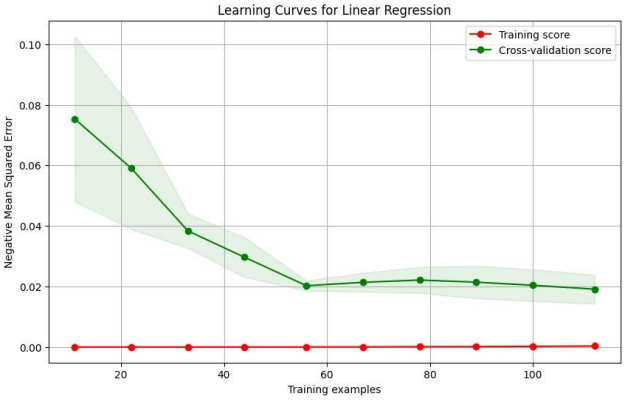
Model GBM is one of the best models of this thesis. Figure 15 shows learning curves for GBM and DT.

Figure 15 Learning curves for GBM and Decision Tree



Model DT like RF performed almost well in this comparison. Figure 16 shows learning curves for LR.

Figure 16 Learning curves for Linear Regression



6 Results

In this section, the results of the analytical comparison of the models are examined and displayed. 14 variables which are called features were implemented in 7 predictive models. Among the features, average goals per game had the most effect on the target variable which is average points per game. After that, average shots on target was more important than other features. Average pass accuracy (%) in most of the models is in the third place in the case of importance of features.

In Table 11, the predictive models are sorted according to the amount of Mean Absolute Error from low to high. Linear Regression, GBM and Random Forest have the least MAE and MSE (Table 12), and also have the most R-squared (Table 13) among the 7 models. Models LR, GBM, RF, XGB and DT have acceptable MAE values, but the values of MAE for SVM and NN models are not suitable. Related to MSE, models GBM, RF, XGB and DT have appropriate values. But the values of SVM, LR and NN models are not satisfactory. R-Squared value of NN is very low and R-Squared value of SVM is almost good but the value of R-squared for other 6 models is in a high range. Overall, Linear Regression and GBM are the best model from the point of view of the values related to the measure of errors (MAE and MSE) and accuracy (R-Squared) on test sets. According to the tables, Neural Networks model is the worst model.

Table 11 Sorting the models in test sets by MAE

Rank	Model	Mean Absolute Error
1	Linear Regression	0.0843
2	GBM	0.0859
3	Random Forest	0.1026
4	XGBoost	0.1147
5	Decision Tree	0.1195
6	SVM	0.1713
7	Neural Networks	0.3429

The table of the sort of the models based on MAE is almost similar to the sort of models based on MSE, only they have a series of minor differences.

Table 12 Sorting the models in test sets by MSE

Rank	Model	Mean Squared Error
1	GBM	0.0130
2	Linear Regression	0. 0132
3	Random Forest	0.0161
4	XGBoost	0.0197
5	Decision Tree	0.0232
6	SVM	0.0442
7	Neural Networks	0.1597

In this thesis, in addition to the error criterion, the accuracy criterion has also been examined for different models.

Table 13 Sorting the models in test sets by R-Squared

Rank	Model	R-Squared
1	GBM	0.9521
2	Linear Regression	0.9513
3	Random Forest	0.9406
4	XGBoost	0.9272
5	Decision Tree	0.9143
6	SVM	0.8367
7	Neural Networks	0.4097

According to Table 14 , Models LR, GBM and RF are the best model in view point of MAE. According to Table 15, models LR, RF and GBM are the best model in case of MSE, however all 7 models have MSE values less than 0.1. According to Table 16, models LR, RF, GBM and XGB have the appropriate values of R-squared. The R-Squared value of model NN is low. So it can be retrieved that the model Linear Regression is the best model from the point of view of the values related to the measure of cross-validate errors (MAE and MSE) and accuracy (R-squared). After LR, the model RF and GBM are the best models in this case.

Table 14 Sorting the models in cross-validate by MAE

Rank	Model	Mean Absolute Error
1	Linear Regression	0.0859
2	GBM	0.1009
3	Random Forest	0.1024
4	XGBoost	0.1057
5	Decision Tree	0.1409
6	SVM	0.1713
7	Neural Networks	0.2451

In terms of validity, Model LR is the best model in terms of errors.

Table 15 Sorting the models in cross-validate by MSE

Rank	Model	Mean Squared Error
1	Linear Regression	0.0122
2	Random Forest	0.0171
3	GBM	0.0171
4	XGBoost	0.0191
5	Decision Tree	0.0326
6	SVM	0.0465
7	Neural Networks	0.0914

Model X is the best in terms of accuracy and validity. As it had fewer errors, it is also highly accurate.

Table 16 Sorting the models in cross-validate by R-squared

Rank	Model	R-squared
1	Linear Regression	0.9437
2	Random Forest	0.9228
3	GBM	0.9225
4	XGBoost	0.9149
5	Decision Tree	0.8514
6	SVM	0.7907
7	Neural Networks	0.5847

All predictive models forecast that football club Inter will champion in the current Serie A football league (2023-24). In fact, the Inter football team with the average points 2.618 per game is currently in the first place in the league. At this time 34 out of 38 league matches have been played. According the Table 17, the predictive model Linear Regression has the best prediction of average points of team Inter in current league. XGB and DT are in the second and third place and have predicted the average points of the Inter team relatively accurately. The rest of the models are not very close to reality in this case.

Table 17 Sorting the predictive models based on average points predictions for the team Inter

Rank	Model	Average Points
1	Linear Regression	2.560
2	XGBoost	2.410
3	Decision Tree	2.400
4	GBM	2.351
5	Random Forest	2.322
6	SVM	1.936
7	Neural Networks	2.807

All models are almost the same in team ranking and differ slightly from reality. As it can be seen in Table 18, among the predictive models, GBM is closer to current standings of Serie A football league.

In comparison the predictive values in models against actual values, the models LR, GBM, RF and XGB are the more accurate models. It seems the model GBM is the best model in this case. Figure 17 shows this comparison as a line chart for predictive model GBM.

Figure 17 Line Chart of Actual vs Predicted Values in GBM

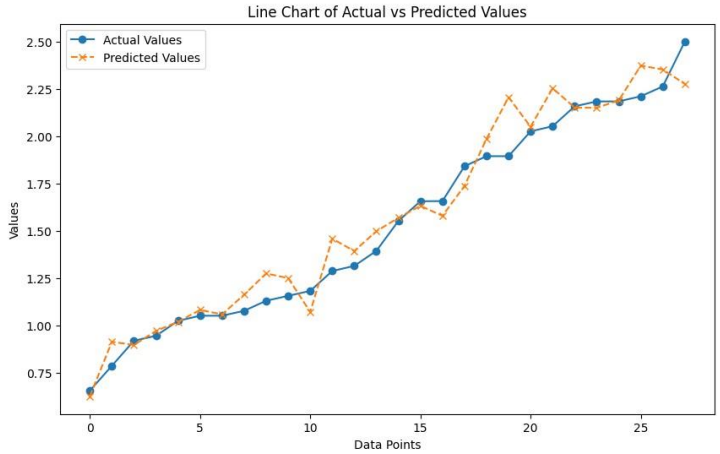


Table 18 Comparison of current standings of Serie A against predictive standings by GBM

GBM Predictive Model	Real Standings (34 matches)
Inter	Inter
Juventus	Milan
Milan	Juventus
Bologna	Bologna
Napoli	Roma
Roma	Atalanta
Atalanta	Lazio
Fiorentina	Fiorentina
Lazio	Napoli
Torino	Torino
Genoa	Monza
Monza	Genoa
Lecce	Lecce
Frosinone	Caligari
Sassuolo	Verona
Udinese	Frosinone
Caligari	Empoli
Verona	Udinese
Empoli	Sassuolo
Salernitana	Salernitana

It can be concluded that the models Linear Regression, GBM, Random Forest, XGB and Decision Tree are suitable models for predicting performance of football matches and among them, the GBM and Linear Regression models are better than all the models. Figure 18 shows that in the model GBM has almost a symmetric bell-shaped histogram which shows that the variance is almost normally distributed.

Figure 18 Histogram and Q-Q plot of residuals in GBM

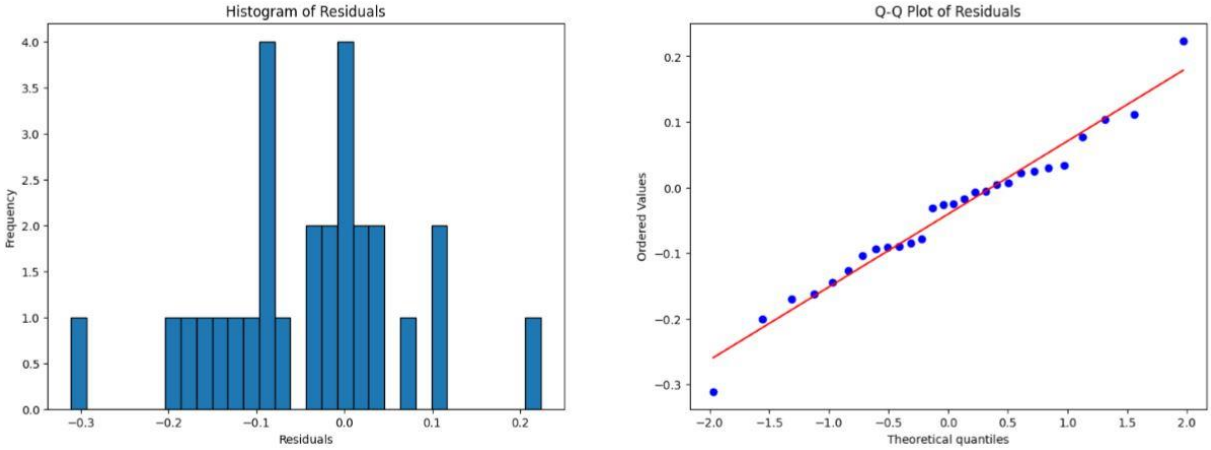
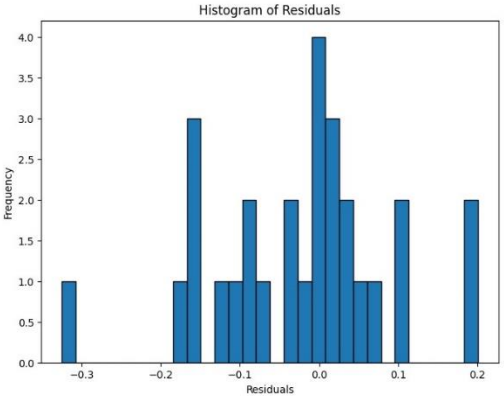


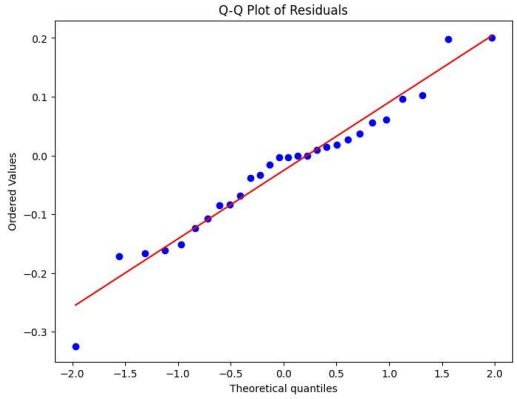
Figure 19 shows the Q-Q plots of the model Linear Regression. The plot shows what exact model LR is.

Figure 19 Histogram of residuals for LR model



Q-Q plot of residuals in LR is presented in Figure 20. This plot shows the closeness of the circles to the blue line.

Figure 20 Q-Q plot of residuals in Linear Regression model.



7 Summary

Football is the most popular sport in the world. For this reason, a lot of wealth flows in it, and its beneficiaries are football clubs, betting companies and individuals. So, it is very important for them to know the outcomes and stats of future football matches. In this thesis, It has been tried to provide a suitable and accurate predictive model which predict the performance of football teams, their ranking and the champion in the current Serie A football league using neural networks and machine learning methods.

It can be learned from the process of this thesis and the results that the use of predictive models requires abundant and accurate historical data. It means that the greater the variety of variables and their number, the more appropriate a model can be made. Also, it is important and vital to choose the method of creating the prediction model and to choose the appropriate parameters for those models. After selecting the models and designing them, the results of these models have been compared and the best model that provides the most accurate results has been introduced. In brief, it can be said that model Linear Regression and GBM provided the most accurate predictions. Also, variable average goals per game had the greatest impact on the target variable. All prediction models correctly guessed the champion of the 2023-24 season. Inter team has won the league at the end of the 34th week. The total number of Serie A football league games is 38. The ranking of the football teams is also reasonably predicted by some models, especially the GBM model.

Definitely, the prediction models presented in this thesis have several flaws, and efforts should be made to improve the models. This can be done by trying newer machine learning models, increasing the number and variety of historical data, etc. But compared to the coefficients considered for the champion of the season as well as the ranking provided by the betting websites, it can be concluded that the prediction models of this thesis are also suitable models to be used in betting websites. Also, football club owners and team coaches can use these prediction models to improve the performance of their teams.

References

Rahman, M. A. (2020). A deep learning framework for football match prediction. (ss. 2(2), 165). SN Applied Sciences. doi:https://doi.org/10.1007/s42452-019-1821-5

2023-2024 Serie A Stats. fbref.com. (2024, April 1). Retrieved from https://fbref.com/en/comps/11/Serie-A-Stats

Corner Stats Italy Serie A 2023/24. betaminic.com. (2024, April 1). Retrieved from https://www.betaminic.com/statistics/corner-stats-serie-a/

Football team statistics—Serie A 18/19, Italy. (n.d.). Theplayer.Com. (2024, April 1). Retrieved from https://theplayer.com/statistics/football/italy/serie-a/18-19/teams

Gomes, J.;Portela, F.;& Santos, M. F. (2015). Decision Support System for predicting Football. In Computers-19th International Conference on Circuits, Systems, Communications and Computers-Intelligent Systems and Applications Special Sessions., (ss. Series (Vol. 32, pp. 348-353).).

Herbinet, C. (2018). Predicting football results using machine learning techniques. MEng thesis, Imperial College London.

Italy Italy Serie A. totalcorner.com. (2024, April 1). Retrieved from https://www.totalcorner.com/league/corner_stats/12

Italy Serie A League Table BET365. (1. April 2024). Noudettu osoitteesta https://www.bet365.com/#/AC/B1/C1/D1002/E92269709/G40/H^1/K^4/

Machine Learning for Evolution Strategies. (2016). In O. Kramer. Springer Cham. doi:https://doi.org/10.1007/978-3-319-33383-0

Serie A Team Statistics. whoscored.com. (2024, April 1). Retrieved from https://www.whoscored.com/Regions/108/Tournaments/5/Seasons/7468/Stages/16548/TeamStatistics/Italy-Serie-A-2018-2019

Sjöberg, F. (2023). Football Match Prediction Using Machine Learning. Åbo Akademi University. Retrieved from https://urn.fi/URN:NBN:fi-fe2023053050645

Sports betting market size, growth, trends, forecast 2021 to 2030. (2024, April 29). Retrieved from https://www.visionresearchreports.com/sports-betting-market/38553

Stubinger, J.;Mangold, B.;& Knoll, J. (2019). Machine Learning in Football Betting: Prediction of Match Results Based on Player Characteristics. Applied Sciences, ss. 10(1), 46 . doi:https://doi.org/10.3390/app10010046

Tiwari , E.;Sardar , P.;& Jain , S. (2020). Football match result prediction using neural networks and deep learning. 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO) (ss. 229-231). IEEE. doi:https://doi.org/10.1109/ICRITO48877.2020.9197811

Unlocking the potential of machine learning: past, present, and future insights. ileaf Solutions. (2024, April 29). Retrieved from https://www.ileafsolutions.com/machine-learning-from-past-to-future-what-you-need-to-know

Zaveri, N.;Shah, U.;Tiwari, S.;Shinde, P.;& Teli, L. K. (2018). Prediction of Football Match Score and Decision Making Process. International Journal on Recent and Innovation Trends in Computing and Communication, (ss. 6(2), 162-165).

Appendix 1: Material management plan

In adherence to HAMK's thesis guidelines for obtaining, processing, storing, and disposing of data., the author has, throughout this research, collected data from several sources. This data is analyzed for the thesis. The data is stored on drive D of the author's computer and is regularly backed up on a usb flash. The data is kept at station D for at least one year after the completion of the thesis