STU22004 Applied Probability I: Group Assignment

Authors: Keith Lyons - 23375743 Luke Hand - 23374689

29-11-2024

Introduction

In this report, we will address the two questions assigned to us:

1. Simulation of a random temperature process over a continuous time interval $t \in [0, 1]$.
2. Forecasting final Premier League standings for the 2024-2025 season based on probabilistic modelling.

Each question was tackled using methodical and mathematical models, simulations, and analysis with visual aids to support our findings. Python played a huge role in allowing us to perform these simulations not just accurately but efficiently too.

Question 1: Simulation of a Random Temperature Process

Objective

Our objective was to model the temperature $X(t)$, which varies randomly over a continuous time interval t, where $t \in [0, 1]$, and estimate the distributions of:

- P: The proportion of time in which the temperature is positive ($X(t) > 0$).
- T_{max}: The time in which the temperature reaches its maximum.

Assumptions

Before testing our model, the following assumptions were inherited:

- We begin with the assumption that $X(0) = 0$, which means that the temperature at time 0 is 0.
- As we chose a small time incremental representation of Δt, we can make the assumption that the change in temperature from time t to $t + \Delta t$, denoted as $X(t + \Delta t) - X(t)$, follows a normal distribution:

$$X(t + \Delta t) - X(t) \sim \mathcal{N}(0, \Delta t).$$

- We also made the assumption that the process behaves consistently over time, with no prior or ongoing bias toward increasing or decreasing trends.

Our Approach

Mathematical

Process Dynamics The process $X(t)$ begins at $X(0) = 0$ with changes in temperature over small time increments in terms of Δt are modelled as such:

$$X(t + \Delta t) - X(t) \sim \mathcal{N}(0, \Delta t).$$

To simulate $X(t)$, we added independent random increments $Z_i \sim \mathcal{N}(0, \Delta t)$ over $n = \frac{1}{\Delta t}$ steps:

$$X(t) = \sum_{i=1}^{n} Z_i.$$

Target Random Variables

- Proportion of Positive Time (P):

$$P = \frac{\text{Number of steps where } X(t) > 0}{n}.$$

- Time of Maximum Temperature ($T_{\max}$):

$$T_{\max} = t_i, \quad \text{where } X(t_i) = \max_t X(t).$$

Python Implementation

Random Increments We used `numpy.random.normal` to generate the increments for $\mathcal{N}(0, \Delta t)$.

```
increments = np.random.randn(n_steps) * np.sqrt(delta_t)
```

Cumulative Sum We used the built-in function `numpy.cumsum` to calculate the cumulative sum for the simulation of the temperature process $X(t)$.

```
X = np.insert(np.cumsum(increments), 0, 0)
```

Monte Carlo Simulations We ran approximately ten thousand simulations for each Δt using range-based for loops.

Data Storage We used `pandas.DataFrame` to facilitate the simulation results for P and $T_{\max}$, respectively.

```
for delta_t in delta_t_values:
    P_samples, Tmax_samples = simulate_temperature_with_validation(delta_t, num_trials)
    results.append(pd.DataFrame({
        'P': P_samples,
        'Tmax': Tmax_samples,
        'Delta_t': delta_t
    }))
```

Visualization We generated two histograms of the results produced by the simulations of P and $T_{\max}$ using the Python module `plotly.express`.

```
p_plot = px.histogram(
    results,
    x="P",
    color="Delta_t",
    barmode="overlay",
    histnorm="probability",
    title="Distribution of P",
    labels={"P": "Proportion of Time Temperature is Positive", "Delta_t": "Delta t"},
    nbins=50
)
```

Observations

- **Distribution of P:** P managed to approximate to about 0.5 on average but also exhibits variability due to the arbitrary nature of the process. This is reflective of our assumptions. Link to Graph
- **Distribution of $T_{\max}$:** $T_{\max}$ managed to conform to the expectations of our assumptions, exhibiting a near-perfect distribution throughout $[0, 1]$. Link to Graph

Question 2: Forecasting Premier League Standings

Objective

The overall objective of this project was to use Monte Carlo Simulation to generate a prediction of the final standings of Premier League teams for the 2024/25 season. The model we used projected team performance based on historical team statistics. Historical data is based on the first 11/12 games of the 2024/25 season as of 23/11/2024, focusing on two different types of metrics—offensive and defensive. We used a plethora of metrics such as points, xG, shots on target, and possession for offensive statistics, and xG conceded, fouls per match, clean sheets, and general discipline for defensive statistics. We based many of our methods for predicting scores on effective algorithms used for sports betting.

Approach

Data Preparation

- Data was scraped from `fotmob.com` and `premierleague.com`.
- We loaded historical statistics, calculating metrics like points, goal difference, and other performance indicators.
- A file named `simulation.csv` was generated using Python to store the results of all simulations.

Visualization We initially planned to use `matplotlib` in Python for visualizations, but after a brief conversation with a colleague, we decided to use `plotly.express`, as it produced more visually appealing graphs.

Mathematical Approach

Offensive and Defensive Strengths Where offensive and defensive strengths are represented by the terms Soff and Sdef, respectively.

$$S_{\text{off}} = \frac{\text{Goals Scored}}{\text{Games Played}}, \quad S_{\text{def}} = \frac{\text{Goals Conceded}}{\text{Games Played}}.$$

Possession and Passing Accuracy Where Possession (Poss) and passing accuracy (Pacc):

$$P_{\text{adjusted}} = \frac{P_{\text{oss}} \times P_{\text{acc}}}{100}.$$

xG and xG Conceded Where xG represents expected goals. Similarly, xG conceded represents expected goals conceded:

$$\lambda_{\text{home}} = \text{xG}_{\text{home}} \times \frac{S_{\text{off}}^{\text{home}}}{S_{\text{def}}^{\text{away}}}, \quad \lambda_{\text{away}} = \text{xG}_{\text{away}} \times \frac{S_{\text{off}}^{\text{away}}}{S_{\text{def}}^{\text{home}}}.$$

Shots on Target Shots on target were factored into goal probability with (Sshot) representing shots on target:

$$G_{\text{adjusted}} = G \times \frac{S_{\text{shot}}}{\text{xG}}.$$

Discipline Metrics (Yellow and Red Cards) Where Ppenalty represents the probability of a team conceding a penalty.

$$P_{\text{penalty}} = \frac{\text{Yellow Cards}}{\text{Games Played}} + 2 \times \frac{\text{Red Cards}}{\text{Games Played}}.$$

Match Simulation Where the goals scored by home and away teams are represented by the terms Ghome and Gaway, respectively.

$$G_{\text{home}} \sim \text{Poisson}(\lambda_{\text{home}}), \quad G_{\text{away}} \sim \text{Poisson}(\lambda_{\text{away}}).$$

League Statistics Where goal difference is updated after each fixture.

$$\text{Goal Difference} = \text{Goals Scored} - \text{Goals Conceded}.$$

Final Standings Where teams were ranked based on:

$$\text{Rank Criteria: Points, Goal Difference, Goals Scored.}$$

Results and Visualization Insights

Average Points Per Team Across Simulations: Graph Type: Bar Chart. *Insights:* Teams such as Liverpool, Manchester City, and Chelsea consistently dominated the league.

Points Distribution Across Simulations: Graph Type: Box Plot. *Insights:* Variability in team performance was observed across simulations.

League Positions Distribution: Graph Type: Histogram. *Insights:* Liverpool and Manchester City frequently finished in the top 3 positions.

Python Implementation

Data Loading

We used the Pandas library to both load and clean data. Cleaning the data included stripping the white spaces from the columns using the `.strip` function.

```python
import numpy as np
import pandas as pd

team_data = pd.read_csv("premStats24_25.csv")
fixture_data = pd.read_csv("fixtures.csv")
team_data.columns = team_data.columns.str.strip()
fixture_data.columns = fixture_data.columns.str.strip()
```

We had to ensure that all teams in the fixtures were also present in the statistics data.

We actually ran into a problem while doing this — we assumed that the names of the teams from the data we scraped would be the same. However, this was not the case as we had loaded data from two different websites, one using abbreviated versions of some team names. This issue was fixed promptly.

Simulation of Fixture Outcomes

Goals for each team were simulated using the NumPy function `np.random.poisson` with all of the expected goals (λ_{home} and λ_{away}) calculated from offensive and defensive strengths.

```python
home_goals = np.random.poisson(home_lambda * home_discipline_factor)
away_goals = np.random.poisson(away_lambda * away_discipline_factor)
```

Updating League Statistics

Match results were determined using a basic `if-else` statement with points being adjusted accordingly. Points, goals scored, goals conceded, and clean sheets were updated dynamically during the simulation. Clean sheets were updated based on whether or not the opposition scored.

```
if home_goals > away_goals:
    return home_goals, away_goals, "Home Win"
elif away_goals > home_goals:
    return home_goals, away_goals, "Away Win"
else:
    return home_goals, away_goals, "Draw"
```

Aggregation

We ran ten thousand simulations using a range-based `for-loop`, similar to the approach used in the first question. All results were appended to a combined DataFrame using the Pandas library. Average performance across simulations was calculated using the functions `groupby` and `agg` to get the mean.

```
average_standings = combined_results.groupby("Teams").agg('mean').reset_index()
```

Final League Table

Teams were ranked based on points, wins, draws, losses, and finally goals scored, goals conceded, and goal difference. All these statistics are important for the final standings when, inevitably, two teams will share the same amount of points. This is pivotal in determining who wins the league, who is given European status, and who gets relegated. GF == Goals For, GA == Goals Against, and GD = Goal Difference.

```
average_standings = average_standings.sort_values(
    by=["Points", "Wins", "Draws", "Losses", "GF", "GA", "GD"],
    ascending=[False, False, False, True, False, True, False]
).reset_index(drop=True)
```

Saving Results

The final standings were written to a CSV file using Python's built-in file streaming functions.

Strengths and Weaknesses of the Project

Strengths

- We used Poisson modelling, which effectively handled discrete event-based outcomes like goals and discipline.
- Monte Carlo simulations provided a robust framework for uncertainty quantification.
- The use of a plethora of metrics allowed for our model to be as accurate as possible.

Weaknesses

- Our model assumes team strengths are static, ignoring dynamic factors such as injuries and transfers.
- The model does not account for interactions between teams and external influences such as refereeing decisions like VAR.
- We hoped to run 100,000 simulations, but with the sheer number of metrics used, this was not feasible in Python. A change to a language like C++ may improve efficiency in the future.

Results and Visualization Insights

Average Points Per Team Across Simulations

Graph Type: Bar Chart Link to Graph
Insights:

- Teams such as Liverpool, Manchester City, and Chelsea consistently dominated the league, averaging 67, 69, and 63 points respectively across all simulations. This aligns with the general expectations of these clubs at this level.
- Nottingham Forrest unexpectedly emerged as a high-performing team so far in this Premier League season. Across all simulations, they averaged 63 points, highlighting potential variability in the league.
- Teams like Southampton and Crystal Palace had significantly lower averages (28 and 33 points, respectively), often finishing in and around the relegation zone, which is consistent with their historical performances in the top flight.

Points Distribution Across Simulations

Graph Type: Box Plot Link to Graph **Insights:**

- Variability in team performance was observed across simulations.
- Manchester City and Liverpool exhibited narrower distributions, indicating consistent performance.
- Brentford and Nottingham Forest displayed wider distributions, suggesting more variability in their predicted standings.
- This graph highlights the uncertainty in predicting league outcomes for mid-table teams.

League Positions Distribution

Graph Type: Histogram Link to Graph **Insights:**

- Liverpool and Manchester City frequently finished in the top 3 positions.

- Southampton, Crystal Palace, and Everton were often clustered in the bottom five, indicating relegation risk.
- The mid-table positions (7th-15th) showed significant overlap between teams like Brighton, Aston Villa, and West Ham.

Points vs Goal Difference

Graph Type: Scatter Plot Link to Graph **Insights:**

- We observed a strong positive correlation between points and goal difference.
- Teams with higher goal differences, such as Manchester City and Liverpool, were considerably consistent among the top-performing teams.
- Teams like Southampton and Crystal Palace, however, clustered at the bottom with considerably low goal differences and points.

Correlation Heatmap

Graph Type: Heatmap Link to Graph
Insights:

- Points strongly correlated with Goals Scored (0.99) and Goal Difference (0.95), confirming the importance of a team's attacking strength.
- xG showed moderate correlations with points and goal difference, suggesting its predictive value.
- Negative correlations were observed between Goals Conceded and metrics like points and goal difference, underscoring the importance of defensive stability.

Points Evolution for a Specific Team

Graph Type: Line Plot Link to Graph
Team Used: Manchester City (our favorite team)
Insights:

- Manchester City exhibited high variability in points across all simulations, fluctuating between 50 and 85 points.
- These trends highlight the inherent unpredictability of football outcomes, even for generally top-performing teams.

Conclusions

Model Effectiveness

- The Poisson-based Monte Carlo simulation effectively modeled the variability of Premier League outcomes.
- Visualizations provided insights into team performance, variability, and uncertainty.

Key Findings

- Teams like Manchester City and Liverpool consistently ranked high in simulations, validating their historical performance.
- Nottingham Forest's unexpected performance underscores the variability in mid-table teams.
- Relegation risks were consistently observed for Southampton and Crystal Palace.

Implications

- Teams aiming for higher league positions should focus on improving goal difference and xG.
- Defensive metrics like goals conceded were critical for predicting relegation risks.

Limitations

- Dynamic factors like transfers, injuries, and tactical changes were not modeled.
- Team performance was assumed to be constant throughout the season.

1 Final Premier League Standings

Position	Teams	Points	Wins	Draws	Losses	GS	GC	GD
1	Liverpool	70	23	1	14	43	21	21
2	Manchester City	65	21	2	15	44	28	15
3	Chelsea	63	21	0	17	47	29	17
4	Tottenham Hotspur	58	19	1	18	47	30	16
5	Brighton & Hove Albion	58	19	1	18	39	31	8
6	Arsenal	57	19	0	19	37	28	8
7	Brentford	56	18	2	18	44	38	6
8	Nottingham Forest	54	18	0	20	31	29	1
9	Aston Villa	54	18	0	20	34	34	0
10	Fulham	52	17	1	20	31	28	3
11	Bournemouth	50	16	2	20	30	30	0
12	Newcastle United	50	16	2	20	26	30	-4
13	Manchester United	47	15	2	21	25	29	-4
14	Leicester City	45	15	0	23	31	40	-9
15	West Ham United	44	14	2	22	26	35	-9
16	Wolverhampton Wanderers	42	14	0	24	32	42	-11
17	Ipswich Town	39	13	0	25	24	38	-15
18	Everton	36	12	0	26	19	34	-16
19	Crystal Palace	32	10	2	26	16	32	-17
20	Southampton	27	9	0	29	14	38	-25

References

- Premier League, 2024. Premier League Stats: Top rated teams, goals per match, shots on target and other stats. [online] Available at: `https://www.premierleague.com/stats` [Accessed 23 Nov. 2024].
- Said, A., 2020. Predicting Premier League match wins using Bayesian Modelling. [online] Medium. Available at: `https://medium.com` [Accessed 23 Nov. 2024].
- Pinnacle, n.d. Poisson Distribution betting: How to predict soccer results using Poisson Distribution. [online] Pinnacle.com. Available at: `https://www.pinnacle.com` [Accessed 23 Nov. 2024].
- Hu, W., 2021. Monte Carlo Simulation in Python: An Introduction. [online] Medium. Available at: `https://medium.com` [Accessed 23 Nov. 2024].
- GeeksforGeeks, n.d. Plotly Tutorial: A beginner's guide to creating visualizations. [online] Available at: `https://www.geeksforgeeks.org` [Accessed 25 Nov. 2024].
- Bishop, C.M., 2006. Pattern Recognition and Machine Learning. 1st ed. New York: Springer. [Accessed 23 Nov. 2024].
- Ross, S.M., 2014. Introduction to Probability Models. 11th ed. Academic Press.
- Premier League, 2024. Fixture List for 2024/25 Season. [online] Available at: `https://www.premierleague.com/fixtures` [Accessed 23 Nov. 2024].