Predicting Troop Betrayal in the War Against the Phrygians

A Comprehensive Decision-Making System

Submitted by:
Neural Nomads

September 24, 2024

Contents

Executive Summary

Betrayal within military ranks poses a significant threat to the integrity and success of the Xernian forces in their impending war against the Phrygians. Historical patterns indicate that the Phrygians' promises of wealth and power have successfully enticed Xernian soldiers to defect. To counter this internal threat, a predictive decision-making system is proposed to identify soldiers at high risk of betrayal. This system leverages data-driven insights by analyzing various factors influencing a soldier's likelihood to defect. The following report details the comprehensive approach to designing, developing, and implementing this predictive system, ensuring strategic management of forces and prevention of treachery.

Chapter 1

Introduction

1.1 Background

In the context of warfare, internal cohesion and loyalty are as critical as external military strength. The Xernian army has faced challenges with soldiers defecting to the Phrygian side, attracted by incentives such as wealth and power. Preventing such betrayals is essential to maintaining operational effectiveness and achieving victory.

1.2 Objective

The primary objective of this report is to outline a systematic approach to predict potential betrayals within the Xernian army. By identifying key factors that contribute to a soldier's decision to defect, the proposed system aims to flag high-risk individuals, enabling preemptive measures to maintain unit integrity.

1.3 Scope

This report covers the identification of relevant predictive factors, data collection methodologies, feature engineering, system workflow, decision-making algorithms, scalability considerations, technical stack, and integration strategies. While code submission is not required, supplementary materials including code and datasets are provided to enhance the system's development and implementation.

Chapter 2

Identifying Key Factors

Understanding the underlying reasons that drive soldiers to betray their clan is pivotal. Through historical analysis and hypothesis formulation, several factors have been identified:

2.1 Factors Influencing Betrayal

1. **Greed**
    - *Description*: The desire for wealth beyond what the clan provides.
    - *Indicators*: Salary satisfaction, bonus expectations.
2. **Temptation**
    - *Description*: Exposure to Phrygian promises of power and prestige.
    - *Indicators*: Phrygian offer exposure, attraction to Phrygian ideals.
3. **Respect**
    - *Description*: Level of respect and recognition within the clan.
    - *Indicators*: Peer recognition count, supervisor approval rating.
4. **Morale**
    - *Description*: Overall satisfaction and motivation of the soldier.
    - *Indicators*: Morale index, engagement level.
5. **Personal Ambition**
    - *Description*: Individual career aspirations that may conflict with clan loyalty.
    - *Indicators*: Career progression score, educational aspirations.
6. **Financial Status**
    - *Description*: Economic hardships making external offers more appealing.
    - *Indicators*: Debt levels, savings.
7. **Family Ties**

- *Description*: Obligations or connections influencing loyalty.
- *Indicators*: Family dependents, family support score.

8. **Peer Influence**
   - *Description*: Influence from fellow soldiers who may defect or consider defection.
   - *Indicators*: Number of defectors in unit, peer pressure score.
9. **Combat Stress**
   - *Description*: Psychological stress from ongoing conflicts affecting decision-making.
   - *Indicators*: Stress level, frequency of PTSD symptoms.
10. **Leadership Trust**
    - *Description*: Trust in clan leadership and its decisions.
    - *Indicators*: Trust in leadership score, frequency of complaints against leadership.

2.2 Quantifying the Factors

Each identified factor is translated into measurable data points:

2.2.1 Greed

- **Salary Satisfaction Score** (1-10)
- **Bonus Expectations** (USD)

2.2.2 Temptation

- **Phrygian Offer Exposure** (number of interactions)
- **Attraction to Phrygian Ideals** (1-10)

2.2.3 Respect

- **Peer Recognition Count** (number of commendations)
- **Supervisor Approval Rating** (1-10)

2.2.4 Morale

- **Morale Index** (1-10)
- **Engagement Level** (% participation in clan activities)

2.2.5 Personal Ambition

- **Career Progression Score** (1-10)
- **Educational Aspirations** (number of courses or trainings pursued)

2.2.6 Financial Status

- **Debt Levels** (USD)
- **Savings** (USD)

2.2.7 Family Ties

- **Family Dependents** (number of dependents)
- **Family Support Score** (1-10)

2.2.8 Peer Influence

- **Number of Defectors in Unit**
- **Peer Pressure Score** (1-10)

2.2.9 Combat Stress

- **Stress Level** (1-10)
- **Frequency of PTSD Symptoms** (count)

2.2.10 Leadership Trust

- **Trust in Leadership Score** (1-10)
- **Frequency of Complaints Against Leadership** (count)

Chapter 3

Data Collection and Feature Engineering

3.1 Data Collection Methods

To accurately predict betrayal, comprehensive data collection is essential. The following methods will be employed:

1. **Surveys and Questionnaires**
    - Regular assessments to gauge morale, trust, personal ambition, and attraction to Phrygian ideals.
2. **Financial Audits**
    - Monitoring of debt levels and savings to assess financial stability.
3. **Performance Reviews**
    - Documentation of supervisor approval ratings and peer recognition counts.
4. **Psychological Assessments**
    - Evaluations to measure stress levels and identify PTSD symptoms.
5. **Social Network Analysis**
    - Examination of peer interactions to detect influence patterns and potential defection clusters.
6. **Interaction Logs**
    - Recording interactions with Phrygian agents or exposure to their propaganda.

3.2 Feature Engineering

Transforming raw data into actionable features involves:

3.2.1 Normalization

Scaling survey-based scores (1-10) to a consistent range (0-1) to ensure uniformity across different metrics.

3.2.2 Composite Indicators

Combining related features, such as creating a **Financial Health Index** by integrating debt and savings figures.

3.2.3 Temporal Features

Tracking changes over time to identify trends, such as declining morale or increasing stress levels.

3.2.4 Interaction Features

Capturing the interplay between different factors, for example, high stress coupled with low morale.

3.3 Dummy Dataset

An extended dummy dataset comprising 100 soldiers has been generated to simulate real-world scenarios. This dataset includes all the aforementioned features along with a target variable indicating betrayal (**1**) or no betrayal (**0**).

Refer to the Appendix for a preview of the dataset and details on its generation.

Chapter 4

System Workflow

The proposed decision-making system follows a structured workflow to ensure accurate prediction and effective management:

4.1 Workflow Stages

1. **Data Ingestion**
    - Collection of data from various sources such as surveys, audits, and assessments.
    - Ensuring data integrity and consistency through validation checks.
2. **Data Preprocessing**
    - Cleaning the data by handling missing values and outliers.
    - Normalizing and scaling features to prepare for analysis.
3. **Feature Selection**
    - Identifying the most predictive features using statistical methods like correlation analysis and feature importance scores derived from models.
4. **Model Training**
    - Utilizing machine learning algorithms trained on historical data to recognize patterns associated with betrayal.
5. **Risk Scoring**
    - Assigning a betrayal risk score to each soldier based on model predictions.
6. **Ranking and Flagging**
    - Ranking soldiers by their risk scores.
    - Flagging high-risk individuals for further monitoring or intervention.
7. **Actionable Insights**

- Providing recommendations for preventive measures, such as counseling or financial support, based on identified risk factors.

8. **Feedback Loop**

- Continuously integrating new data to refine and improve the model, ensuring adaptability to changing circumstances.

Chapter 5

Decision-Making Algorithm

5.1 Model Selection

Given the binary nature of betrayal prediction, classification algorithms are suitable. The following models are considered:

5.1.1 Logistic Regression

- *Pros*: Simplicity and interpretability.
- *Cons*: Limited in capturing complex relationships.

5.1.2 Random Forest

- *Pros*: Handles non-linear relationships and interactions effectively.
- *Cons*: Can be computationally intensive with large datasets.

5.1.3 Gradient Boosting Machines (e.g., XGBoost)

- *Pros*: High accuracy and robustness in handling complex patterns.
- *Cons*: Requires careful tuning of hyperparameters.

5.1.4 Support Vector Machines (SVM)

- *Pros*: Effective in high-dimensional spaces.
- *Cons*: Less interpretable and can be slow with large datasets.

5.1.5 Neural Networks

- *Pros*: Capable of modeling highly complex relationships.
- *Cons*: Requires substantial data and computational resources.

5.2 Model Training and Validation

5.2.1 Dataset Splitting

- **Training Set**: 80%
- **Test Set**: 20%

5.2.2 Evaluation Metrics

- **Accuracy**: Overall correctness of predictions.
- **Precision**: Correctly identified betrayals out of all flagged.
- **Recall (Sensitivity)**: Correctly identified betrayals out of actual betrayals.
- **F1-Score**: Balance between precision and recall.
- **ROC-AUC**: Measure of the model's ability to distinguish between classes.

5.3 Example Implementation: Random Forest

While code submission is optional, providing a Random Forest implementation can offer deeper insights. Below is a conceptual overview:

1. **Import Necessary Libraries**
2. **Load and Preprocess the Dataset**
3. **Split the Data into Training, Validation, and Test Sets**
4. **Initialize and Train the Random Forest Model**
5. **Perform Hyperparameter Tuning Using Cross-Validation**
6. **Evaluate the Model Using Relevant Metrics**
7. **Interpret Feature Importance and Derive Actionable Insights**

Refer to the Appendix for the complete Python script and dataset.

Chapter 6

Scalability and Adaptability

6.1 Continuous Learning

6.1.1 Incremental Training

Updating the model periodically with new data to maintain accuracy without retraining from scratch.

6.1.2 Online Learning Algorithms

Implementing models that can learn in real-time as data streams in, ensuring responsiveness to emerging patterns.

6.2 Data Pipeline

6.2.1 Automated Data Ingestion

Utilizing ETL (Extract, Transform, Load) processes to seamlessly handle data from multiple sources.

6.2.2 Real-Time Analytics

Employing streaming platforms like Apache Kafka for real-time data processing and immediate risk assessment.

6.3 Modular Architecture

6.3.1 Microservices

Decomposing the system into smaller, manageable services (e.g., data processing, model serving) to facilitate scaling and maintenance.

6.3.2 Containerization

Using Docker to containerize applications, enhancing deployment flexibility and scalability across different environments.

6.4 Monitoring and Maintenance

6.4.1 Performance Monitoring

Tracking model performance metrics over time to detect and address any degradation in accuracy.

6.4.2 Alert Systems

Setting up alerts for unusual patterns or significant drops in prediction performance.

6.4.3 Regular Updates

Scheduling periodic reviews and updates to incorporate new insights, data, and advancements in modeling techniques.

Chapter 7

Technical Stack

7.1 Programming Languages

- **Python**: Primary language for data analysis, machine learning, and system integration.

7.2 Libraries and Frameworks

7.2.1 Data Processing

- **pandas**: Data manipulation and analysis.
- **numpy**: Numerical operations and array management.

7.2.2 Machine Learning

- **scikit-learn**: Traditional machine learning models.
- **XGBoost / LightGBM**: Gradient boosting frameworks for enhanced performance.
- **TensorFlow / PyTorch**: Neural network frameworks for deep learning applications.

7.2.3 Data Visualization

- **matplotlib** and **seaborn**: Exploratory data analysis and static visualizations.
- **Plotly** or **Dash**: Interactive dashboards for real-time insights.

Chapter 8

Integration Strategy

8.1 System Components

1. **Data Layer**
    - **Databases**: Secure storage of collected data, ensuring data integrity and accessibility.
    - **Data Pipeline**: Automated ETL processes to handle data ingestion, transformation, and loading.
2. **Application Layer**
    - **Backend Services**: APIs developed using Flask or FastAPI to facilitate data access and model predictions.
    - **Machine Learning Models**: Hosted models capable of real-time prediction based on incoming data.
3. **Presentation Layer**
    - **Dashboards**: User-friendly interfaces displaying risk scores, trends, and actionable insights.
    - **Alerts and Notifications**: Real-time alerts for high-risk soldiers, enabling timely interventions.

8.2 Integration Steps

1. **Set Up Data Infrastructure**
    - Implement and configure databases and data pipelines to manage data flow effectively.
2. **Develop Backend Services**
    - Build and deploy APIs that interface with the machine learning models for prediction requests.
3. **Deploy Machine Learning Models**

- Containerize models using Docker and deploy them on cloud platforms for scalability and reliability.

4. **Create User Interfaces**
    - Develop interactive dashboards using web frameworks like Dash or React to visualize insights and risk assessments.
5. **Implement Security Measures**
    - Ensure data privacy through encryption, access controls, and secure authentication mechanisms.
6. **Test the System**
    - Conduct comprehensive testing, including unit tests, integration tests, and user acceptance tests to ensure system reliability and accuracy.
7. **Launch and Monitor**
    - Deploy the system into the production environment and establish continuous monitoring to maintain performance and security.

Chapter 9

Conclusion

Betrayal within military ranks can severely compromise the success of the Xernian forces. By systematically identifying and quantifying the factors influencing betrayal, and leveraging advanced machine learning techniques, the proposed decision-making system offers a robust framework to predict and prevent internal threats. The system's design emphasizes clarity, effectiveness, scalability, and adaptability, ensuring it evolves alongside the dynamic battlefield environment. Implementing this system will enhance strategic management of forces, safeguarding the clan's integrity and bolstering the war effort against the Phrygians.

Appendix A

Appendix

A.1 Sample Dummy Dataset Preview

Below is a preview of the first **20 soldiers** in the extended dummy dataset:

Soldier_ID	Salary_Satisfaction_Score	Bonus_Expectations	Phrygian_Offer_Exposure
1	7	1037	0
2	4	4380	1
3	4	6565	1
4	8	8207	0
5	7	5937	1
6	7	2458	2
7	9	1904	2
8	5	8176	3
9	6	8929	2
10	9	7692	2
11	6	4830	1
12	3	6000	4
13	5	7200	2
14	4	8500	1
15	9	2200	1
16	5	6100	3
17	6	4600	2
18	3	9200	6
19	8	3100	1
20	2	10200	7

Note: The full dataset contains **100 soldiers**. Refer to the attached dataset file for complete information.

A.2 Dataset Generation

The extended dummy dataset was generated using a Python script that simulates realistic soldier profiles based on the identified factors. Key aspects of the dataset generation include:

A.2.1 Randomization with Constraints

- Utilized random distributions (e.g., Poisson, Uniform) to simulate variability in features.
- Applied constraints to ensure data realism (e.g., clipping values within logical ranges).

A.2.2 Betrayal Assignment Logic

- Calculated a betrayal probability score based on factors such as Phrygian offer exposure, attraction to Phrygian ideals, peer pressure, stress level, and inversely on trust in leadership.
- Normalized the betrayal probability to a range between 0 and 1.
- Assigned betrayal (**1**) if a randomly generated number fell below the betrayal probability; otherwise, no betrayal (**0**).

A.3 Python Script Overview

The Python script (‘generate$_e$$xtended_d$$ummy_d$$ataset.py$‘)$encompassesthefollowingsteps$:

1. **Import Libraries**
   - ‘pandas‘ and ‘numpy‘ for data manipulation and numerical operations.
2. **Set Random Seed**
   - Ensures reproducibility of the dataset.
3. **Generate Features**
   - Created each feature based on defined distributions and constraints.
4. **Calculate Betrayal Probability**
   - Integrated multiple factors to derive a probability score indicative of betrayal risk.
5. **Assign Betrayal Labels**
   - Determined betrayal status based on the calculated probabilities.
6. **Create and Export DataFrame**
   - Compiled all features into a pandas DataFrame.
   - Exported the dataset to a CSV file for further analysis and model training.

A.4 Python Script

Filename: ‘generate$_e$$xtended_d$$ummy_d$$ataset.py$‘

```
import pandas as pd
import numpy as np

np.random.seed(42)

num_soldiers = 100  # You can increase this number as needed

soldier_ids = np.arange(1, num_soldiers + 1)

salary_satisfaction = np.random.randint(1, 11, size=num_soldiers)

bonus_expectations = np.random.randint(0, 10001, size=num_soldiers)

phrygian_offer_exposure = np.random.poisson(lam=2, size=num_soldiers)
phrygian_offer_exposure = np.clip(phrygian_offer_exposure, 0, 10)

attraction_phrygian = np.random.randint(1, 11, size=num_soldiers)

peer_recognition = np.random.poisson(lam=3, size=num_soldiers)
peer_recognition = np.clip(peer_recognition, 0, 10)

supervisor_approval = np.random.randint(1, 11, size=num_soldiers)

morale_index = np.random.randint(1, 11, size=num_soldiers)

engagement_level = np.random.randint(0, 101, size=num_soldiers)

career_progression = np.random.randint(1, 11, size=num_soldiers)

educational_aspirations = np.random.poisson(lam=1, size=num_soldiers)
educational_aspirations = np.clip(educational_aspirations, 0, 5)

debt_levels = np.random.randint(0, 20001, size=num_soldiers)

savings = np.random.randint(0, 30001, size=num_soldiers)

family_dependents = np.random.poisson(lam=1.5, size=num_soldiers)
family_dependents = np.clip(family_dependents, 0, 5)

family_support = np.random.randint(1, 11, size=num_soldiers)

number_defectors = np.random.poisson(lam=1, size=num_soldiers)
number_defectors = np.clip(number_defectors, 0, 5)

peer_pressure = np.random.randint(1, 11, size=num_soldiers)
```

```
stress_level = np.random.randint(1, 11, size=num_soldiers)

frequency_ptsd = np.random.poisson(lam=1, size=num_soldiers)
frequency_ptsd = np.clip(frequency_ptsd, 0, 10)

trust_leadership = np.random.randint(1, 11, size=num_soldiers)

frequency_complaints = np.random.poisson(lam=0.5, size=num_soldiers)
frequency_complaints = np.clip(frequency_complaints, 0, 5)

betrayal_prob = (
    (phrygian_offer_exposure * 0.2) +
    (attraction_phrygian * 0.3) +
    (peer_pressure * 0.2) +
    (stress_level * 0.2) -
    (trust_leadership * 0.3)
)

betrayal_prob = (betrayal_prob - betrayal_prob.min()) / (betrayal_prob.max() - betray

betrayal = np.where(np.random.rand(num_soldiers) < betrayal_prob, 1, 0)

data = {
    'Soldier_ID': soldier_ids,
    'Salary_Satisfaction_Score': salary_satisfaction,
    'Bonus_Expectations': bonus_expectations,
    'Phrygian_Offer_Exposure': phrygian_offer_exposure,
    'Attraction_to_Phrygian_Ideals': attraction_phrygian,
    'Peer_Recognition_Count': peer_recognition,
    'Supervisor_Approval_Rating': supervisor_approval,
    'Morale_Index': morale_index,
    'Engagement_Level': engagement_level,
    'Career_Progression_Score': career_progression,
    'Educational_Aspirations': educational_aspirations,
    'Debt_Levels': debt_levels,
    'Savings': savings,
    'Family_Dependents': family_dependents,
    'Family_Support_Score': family_support,
    'Number_of_Defectors_in_Unit': number_defectors,
    'Peer_Pressure_Score': peer_pressure,
    'Stress_Level': stress_level,
    'Frequency_of_PTSD_Symptoms': frequency_ptsd,
    'Trust_in_Leadership_Score': trust_leadership,
    'Frequency_of_Complaints_Against_Leadership': frequency_complaints,
    'Betrayal': betrayal
}
```

```
df = pd.DataFrame(data)

df.to_csv('extended_dummy_troop_betrayal_dataset.csv', index=False)

print("Extended dummy dataset generated and saved as 'extended_dummy_troop_betrayal_d
```

A.5 Running the Script

1. **Save the Script**
   - Save the above script to a file named `generate_extended_dummy_dataset.py`.
2. **Install Dependencies**
   - Ensure you have Python installed.
   - Install required libraries using pip:

     ```
     pip install pandas numpy
     ```

3. **Execute the Script**
   - Run the script via the command line:

     ```
     python generate_extended_dummy_dataset.py
     ```

   - This will generate a CSV file named `extended_dummy_troop_betrayal_dataset.csv` containing 100 soldiers with all necessary features.

A.6 Dataset Usage

The generated dataset serves as a foundational tool for developing and testing the betrayal prediction system. It can be utilized for:

- **Model Training and Evaluation**
  - Training machine learning models to predict betrayal.
  - Evaluating model performance using various metrics.
- **Data Analysis**
  - Exploring feature distributions and relationships.
  - Identifying key predictors of betrayal.
- **System Development**
  - Integrating the dataset into the decision-making system for real-time predictions.

A.7 Additional Resources

- **README Instructions**
  - A README file accompanies the dataset and script, providing detailed instructions on usage, dependencies, and troubleshooting.
- **Code Repository**
  - For bonus points, a GitHub repository is provided containing the script, dataset, and README:

    https://github.com/ansh200516/solution-for-problem-2.git

Appendix B

References

1. **Data Science Concepts**
    - Feature Engineering: Transforming raw data into meaningful features for machine learning.
    - Model Evaluation Metrics: Understanding accuracy, precision, recall, F1-score, and ROC-AUC.
2. **Machine Learning Techniques**
    - Classification Algorithms: Logistic Regression, Random Forest, Gradient Boosting, SVM, Neural Networks.
    - Hyperparameter Tuning: Techniques such as Grid Search and Random Search for optimizing model performance.
3. **System Design Principles**
    - Scalability: Designing systems that can handle increasing amounts of data and complexity.
    - Modularity: Building systems with interchangeable and independent components for ease of maintenance and scalability.