Predicting Troop Betrayal in the War Against the Phrygians

1. Problem Overview

As the leader of the Xernian army, you are tasked with predicting which soldiers might betray your clan and join the Phrygians. The aim is to create a decision-making system that identifies potential traitors based on various risk factors, helping prevent treachery within the ranks.

Objective:

- Develop a system that evaluates soldiers and predicts their likelihood of betrayal.
- Identify key factors that could contribute to betrayal and quantify these into data points.
- Rank each soldier based on their risk level and continuously adapt as new data is gathered.

---

2. Hypothesis and Feature Selection

The system predicts soldier betrayal using historical and behavioral data. We hypothesize that the following factors influence betrayal:

Key Factors (Features):

2. **Greed and Wealth Temptation**

   Feature: Income Percentile (xi_1) – The soldier's financial status relative to peers.

3. **Respect and Leadership**

   Feature: Disciplinary Record (xi_2) – Number of disciplinary actions, normalized for fairness.

4. Proximity to Enemy Territory

   Feature: Distance to Border (xi_3) – Station proximity to the Phrygian border, normalized.

5. **Social Bonds**

   Feature: Number of Comrades (xi_4) – Frequency of interaction with comrades.

6. **Family History of Defection**

   Feature: Family History of Betrayal (xi_5) – Binary variable (1 if a family member defected, 0 otherwise).

7. **Age**

   Feature: Soldier's age – To capture maturity and potential influence on loyalty.

---

3. Mathematical Model for dataset creation

3.1 Risk Score Calculation:

The risk of betrayal for each soldier iii is calculated as a weighted sum of the features:

$$Ri=w1\cdot xi1+w2\cdot xi2+w3\cdot xi3+w4\cdot xi4+w5\cdot xi5$$

Where:

- wj are weights that reflect the importance of each feature xij
- Each xij is a normalized feature value for soldier i.

3.2 Risk Threshold:

To flag soldiers at high risk of betrayal, we define a **threshold T**. Soldiers with **Ri > T** are marked as high risk:

Flag Soldier if **Ri > T**

The threshold T can be determined based on historical data and desired sensitivity.

---

4. Features for Predicting Troop Betrayal

The prediction of troop betrayal relies on carefully selected features that quantify the factors influencing a soldier's likelihood to defect. These features include:

1. **Income Percentile**: This feature represents the soldier's relative financial status within the army. Soldiers with lower income percentiles may be more susceptible to defection due to the temptation of wealth offered by the enemy.
2. **Disciplinary Record**: This feature quantifies the number and severity of infractions in a soldier's record. A poor disciplinary record can indicate dissatisfaction with leadership or the clan, increasing the risk of betrayal.
3. **Proximity to Enemy Territory**: Soldiers stationed closer to the Phrygian border may be at higher risk of defection due to the ease of communication and interaction with the enemy, making geographical location a critical feature.
4. **Social Bonds (Comrades Count)**: The number of strong social connections within the army can act as a deterrent to betrayal. Soldiers with fewer comrades may feel less loyalty to the group and more prone to defection.
5. **Family History of Betrayal:** A binary feature that flags whether a soldier's family has a history of defection. Soldiers with a family history of betrayal may be more inclined to follow the same path.

These features form the foundation of the decision-making system, providing a comprehensive view of the factors that could lead to treachery.

---

4. Optimization of Weights

The weights wj are optimized using historical data on betrayal cases. For example, if yi=1 indicates betrayal and yi=0 indicates loyalty, we minimize the following logistic loss function:

$$L(w) = -i = 1 \sum N[yilog(\sigma(Ri)) + (1 - yi)log(1 - \sigma(Ri))]$$

- $\sigma(R_i) = 1/(1 + e^{-R_i})$ is the logistic function for predicting the probability of betrayal.
- Ri is the risk score for soldier i.

We use gradient descent or other optimization methods to minimize L(w)L(w)L(w) and learn the best weights wj.

---

5. System Design

5.1 Workflow:

1. **Data Collection:**
   Gather data on soldier income, disciplinary records, proximity to enemy borders, social bonds, and family history.
2. **Preprocessing**
   Normalize the data (income percentile, number of comrades, etc.) and convert qualitative data (family history, disciplinary record) into numeric values.
3. **Risk Score Calculation:**
   Use the weighted scoring formula Ri=w1⋅xi1+w2⋅xi2+w3⋅xi3+w4⋅xi4+w5⋅xi5 to compute betrayal risk for each soldier.
4. **Decision-Making:**
   Soldiers with risk score Ri>T are flagged. The system ranks soldiers by Riand outputs a list of the most likely traitors.
5. **Adaptation:**
   As new data on betrayal becomes available, the system retrains using machine learning algorithms (e.g., logistic regression, random forests) to adjust weights and improve accuracy.

5.2 Scalability:

- The system can easily scale as new soldiers and data points are added. Using reinforcement learning, the system dynamically adjusts its predictions based on real outcomes.
- Data is stored in a scalable database such as MongoDB, while real-time risk scores can be presented via Streamlit dashboards.

---

6. Example Calculation

Consider three soldiers with the following feature values:

Feature	Soldier 1	Soldier 2	Soldier 3
Income	0.3	0.7	0.2
Discipline	0.1	0.9	0.5
Proximity	0.6	0.4	0.8
Comrades	0.5	0.2	0.7
Family Hist.	1.0	0.0	1.0
Age	0.9	0.8	0.7

With weights w1=0.2, w2=0.3, w3=0.1, w4=0.25, w5=0.15, the risk scores are calculated as:

$$R1 = 0.2(0.3) + 0.3(0.1) + 0.1(0.6) + 0.25(0.5) + 0.15(1) = 0.56$$

$$R2 = 0.2(0.7) + 0.3(0.9) + 0.1(0.4) + 0.25(0.2) + 0.15(0) = 0.61$$

$$R3 = 0.2(0.2) + 0.3(0.5) + 0.1(0.8) + 0.25(0.7) + 0.15(1) = 0.73$$

In this example, **Soldier 3** has the highest betrayal risk score and should be prioritized for monitoring.

- **Created Datasets (500x) and used the decision making strategies for calculated risk_score then betrayal risk :bool :**

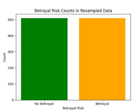
	Soldier_ID	Income_Percentile	Disciplinary_Record	Proximity_to_Border	Comrades_Count	Family_History_of_Betrayal	Age	Betrayal_Risk
0	S1	0.44	0	0.61	11	Yes	52	No
1	S2	0.96	3	0.76	13	No	24	No
2	S3	0.76	4	0.89	8	No	39	Yes
3	S4	0.64	3	0.46	9	No	50	No
4	S5	0.24	5	0.38	12	No	64	No
5	S6	0.24	4	0.77	9	No	50	Yes
6	S7	0.15	4	0.85	12	Yes	63	Yes
7	S8	0.88	2	0.48	5	Yes	51	No
8	S9	0.64	4	0.28	4	No	33	Yes
9	S10	0.74	3	0.95	11	Yes	64	No

```
Soldier_ID  Probability_of_Betrayal
      S522                     0.11
      S738                     0.40
      S741                     0.53
      S661                     0.02
      S412                     0.07
      S679                     0.03
      S627                     0.85
      S514                     0.95
      S860                     0.48
      S137                     0.00
      S812                     0.30
       S77                     0.02
      S637                     0.06
      S974                     0.73
      S939                     0.18
      S900                     0.26
      S281                     0.07
      S884                     0.85
      S762                     0.96
      S320                     0.99
      S550                     0.04
      S175                     0.57
      S372                     0.36
      S528                     0.37
...
      S333                     0.55
      S209                     0.14
      S614                     0.03
       S79                     0.04
```

```
Random Forest Model Evaluation Metrics:
Accuracy: 0.9400
Recall: 0.8696
Precision: 0.9524
F1-Score: 0.9091
Time to Train: 0.1851 seconds
Time to Predict: 0.0150 seconds
Total Time: 0.2113 seconds
```

Random Forest Model without SMOTE (imbalanced dataset) gives best accuracy (94%) obviously because we have used if-else to construct our dataset.

```
Random Forest after SMOTE Evaluation Metrics:
Accuracy: 0.8950
Recall: 0.9130
Precision: 0.8077
F1-Score: 0.8571
```



Random Forest Model with SMOTE(balanced dataset) gives best accuracy (89.5%) obviously because we have used if-else to construct our dataset.

part2.ipynb in github