**Benjamin Nguyen**

# Data Preprocessing - Reduction, Balancing, Merging, Aggregating, Splitting

1. Defining the target variable (outcome)
I created a new column called outcome to represent match results. The encoding was: 1 = home win, -1 = away win, 0 = draw. This ensured that the dataset captured all three possible outcomes rather than collapsing them into a binary win/loss. Having a well-defined target is essential for supervised learning tasks.

2. Balancing the dataset (SMOTE)
The original dataset was imbalanced: home wins were the majority class, while draws were the minority. Many machine learning models perform poorly when trained on imbalanced data, since they tend to favor the majority class. I applied SMOTE (Synthetic Minority Oversampling Technique) to generate synthetic samples for minority classes by interpolating between existing observations. This resulted in equal representation of all three classes (home win, away win, draw), allowing the model to learn each outcome fairly.

3. Standardizing numeric features
Before applying PCA, I standardized all numeric columns using StandardScaler so that each feature had mean = 0 and standard deviation = 1. This was necessary because PCA is scale-sensitive: features with large numeric ranges (like attendance) could otherwise dominate the principal components.

4. Dimensionality reduction with PCA
I applied Principal Component Analysis (PCA) to reduce the number of features while retaining most of the variance in the dataset. By setting n_components=0.95, I kept enough components to explain ~95% of the variance. This reduced the feature set from 40 numeric variables to 24 principal components. The rationale was to remove redundancy, reduce noise, and improve computational efficiency for future models.

5.Train/test split Finally, I split the dataset into training (80%) and testing (20%) sets using train_test_split. The split was stratified by the outcome column so that the class distribution was preserved in both sets. This ensures that evaluation on the test set reflects real-world performance across all match outcomes.

6. Screenshot of the code: (next page)

```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from imblearn.over_sampling import SMOTE
import matplotlib.pyplot as plt

# 1. Load dataset
df = pd.read_csv("mydata.csv")
print("Original shape:", df.shape)

# 2. Define a target variable for balancing
#    Outcome = 1 (home win), 0 (draw), -1 (away win)
if "Goals Home" in df.columns and "Away Goals" in df.columns:
    df["outcome"] = np.where(df["Goals Home"] > df["Away Goals"], 1,
                             np.where(df["Goals Home"] < df["Away Goals"], -1, 0))

# 3. Balance dataset using SMOTE
# Select numeric features (exclude outcome)
num_cols = df.select_dtypes(include=np.number).columns.tolist()
num_cols.remove("outcome")

X = df[num_cols]
y = df["outcome"]

print("\nClass distribution before SMOTE:\n", y.value_counts())

smote = SMOTE(random_state=42)
X_res, y_res = smote.fit_resample(X, y)

print("\nClass distribution after SMOTE:\n", pd.Series(y_res).value_counts())

# 4. Apply PCA on the resampled dataset
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_res)

# keep enough components to explain ~95% variance
pca = PCA(n_components=0.95)
X_pca = pca.fit_transform(X_scaled)

print("\nReduced shape after PCA:", X_pca.shape)
print("Explained variance ratio per component:", pca.explained_variance_ratio_)
print("Total variance explained:", pca.explained_variance_ratio_.sum())

# Optional: Scree plot
plt.figure(figsize=(8,5))
plt.plot(np.cumsum(pca.explained_variance_ratio_), marker='o')
plt.xlabel("Number of Components")
plt.ylabel("Cumulative Explained Variance")
plt.title("Explained Variance by PCA Components (after SMOTE)")
plt.grid(True)
plt.show()

# 5. Create final balanced + PCA-transformed dataset
df_final = pd.DataFrame(X_pca, columns=[f"PC{i+1}" for i in range(X_pca.shape[1])])
df_final["outcome"] = y_res

print("\nFinal dataset shape:", df_final.shape)
df_final.to_csv("mydata_balanced_pca.csv", index=False)
print("Saved as mydata_balanced_pca.csv")
```

✓ 0.5s Python

```
Original shape: (1140, 40)

Class distribution before SMOTE:
outcome
```

LF {} Finish Setup Cell 1 of 2