Hoang Anh Nguyen

Data Preprocessing - Data Cleansing

Because the original dataset was already clean, I first created an unclean version by deliberately introducing missing values, duplicate rows, inconsistencies, and outliers. These issues were injected randomly using the `sample()` function in pandas to simulate realistic data quality problems that often occur in practice.

After creating this unclean dataset, I applied a preprocessing workflow to restore data quality. The process involved three main steps. First, I identified and handled missing values by either filling them with appropriate statistics such as the mean or median for numeric columns or replacing them with a placeholder like `"Unknown"` for categorical fields. Second, I identified and removed duplicate rows, both exact and semantic, to prevent the same match from being counted more than once. Finally, I checked for and corrected inconsistencies in the data, such as formatting issues in categorical fields and unrealistic numerical values (outliers), replacing or standardizing them to ensure the dataset was consistent and credible.

The dataset initially showed several quality issues, including missing values, duplicated entries, inconsistent formatting, and unrealistic numerical values. The first step was to convert numeric-looking text, such as the `attendance` column stored as strings with commas, into proper numeric types. This ensured statistical measures like mean and median could be computed correctly. Missing values were then addressed: numeric columns (for example, goals, shots, and possession) were filled with their median values to preserve central tendencies while limiting the effect of outliers, and categorical columns (such as stadium and date) were filled with the placeholder `"Unknown"` to retain rows without introducing bias.

Duplicate records were removed in two stages. Exact duplicates were dropped first, eliminating rows that were identical across all fields. However, some matches still appeared more than once with only cosmetic differences. To address these, a semantic key built from date, Home Team, Away Team, Goals Home, and Away Goals was used to identify duplicate matches more robustly. This ensured that each match was represented once, preventing inflated statistics.

Formatting inconsistencies in categorical fields were also corrected. Team and stadium names were standardized by trimming whitespace, collapsing multiple spaces, and applying consistent capitalization. This prevented logically identical entries, such as " manchester united " and "Manchester United", from being treated as separate categories. Finally, outliers were detected and handled. Attendance values of zero or greater than 100,000, which fell outside realistic stadium capacities, were replaced with the column median. Similarly, goalcounts greater than ten were considered unrealistic and were also replaced with median values.

After cleaning, the dataset contained no missing values, no duplicate matches, standardized categorical fields, and realistic numerical ranges. The resulting cleaned dataset provides a consistent and trustworthy foundation for statistical analysis and visualization.

```
if "attendance" in df.columns:
    s = df["attendance"].astype(str).str.replace(",", "", regex=False).str.strip()
    df["attendance"] = pd.to_numeric(s, errors="coerce")

for col in obj_cols_initial:
    if col == "attendance":
        continue
    s = df[col].astype(str)
    if looks_numeric_series(s):
        df[col] = pd.to_numeric(s.str.replace(",", "", regex=False).str.strip(),
                                errors="coerce")

# B) Missing values (identify + handle)
na_before = df.isna().sum()
print("\n🔹 Missing values BEFORE:")
print(na_before[na_before > 0].sort_values(ascending=False))

numeric_cols = df.select_dtypes(include="number").columns.tolist()
cat_cols     = df.select_dtypes(include="object").columns.tolist()

# Fill numeric with median (log each)
for col in numeric_cols:
    n_missing = df[col].isna().sum()
    if n_missing > 0:
        med = df[col].median()
        df[col] = df[col].fillna(med)
        print(f"Filled {n_missing} missing values in numeric column '{col}' with median

# Fill categorical with "Unknown" (log each)
for col in cat_cols:
    n_missing = df[col].isna().sum()
    if n_missing > 0:
        df[col] = df[col].fillna("Unknown")
        print(f"Filled {n_missing} missing values in categorical column '{col}' with 'Un

print("🔹 Missing values AFTER (total):", int(df.isna().sum().sum()))

# C) Remove exact duplicate rows
dups_before = int(df.duplicated().sum())
print("\n🔹 Exact duplicate rows BEFORE:", dups_before)
df = df.drop_duplicates().copy()
dups_after = int(df.duplicated().sum())
print("🔹 Exact duplicate rows REMOVED:", dups_before - dups_after)

# D) Formatting fixes for categoricals
for col in cat_cols:
    if col in df.columns:
        s = df[col].astype(str).str.strip()
        s = s.str.replace(r"\s+", " ", regex=True)
        if col not in {"date", "clock", "links"}:
            s = s.str.title()
        df[col] = s

print("\n🔹 Formatting fixes applied: attendance normalized; categorical text standardiz

# E) Outliers (identify + correct)
if "attendance" in df.columns:
    med_att = df["attendance"].median()
    mask_att_zero = df["attendance"] == 0
    mask_att_high = df["attendance"] > 100_000
    n_out_att = int((mask_att_zero | mask_att_high).sum())
    print(f"\n🔹 Attendance outliers found: {n_out_att}")
    if n_out_att > 0:
        df.loc[mask_att_zero | mask_att_high, "attendance"] = med_att
        print(f"Replaced attendance outliers with median ({med_att}).")

for gcol in ["Goals Home", "Away Goals", "home_goals", "away_goals"]:
    if gcol in df.columns and pd.api.types.is_numeric_dtype(df[gcol]):
```