Data Exploration and Solution Planning

Project Title: Identifying Data Integration Quality for Multi – Source

Analytics Team Members:

1. **Name**: GeethaPriya KA

   CAN_ID: 33738378

Institution Name: Vemana Institute of Technology

Overview of Data Integration Quality Assessment

Phase 1 documents the progress made in understanding data integration quality issues, identifying inconsistencies, and planning for the model design. This phase focuses entirely on **exploratory data analysis (EDA)** and leveraging **visualizations** to assess data quality from multiple sources before applying any integration models.

Objectives:

- Develop visualizations to assess the consistency, completeness, and accuracy of integrated data.
- Identify integration anomalies such as duplicate records, schema mismatches, and missing data patterns.
- Establish quality metrics and benchmarks for evaluating data integration effectiveness.

---

2. Data Cleaning and Preparation

2.1 Handling Missing Values

Using a multi-source dataset, missing values were identified and addressed as follows:

- **Numerical Features:** Imputed using the median to maintain statistical integrity while reducing the effect of outliers.

- **Categorical Features:** Assigned a placeholder value "Unknown" to retain missing categorical entries for further analysis.

Code Example:

```
# Handling missing values

import pandas as pd

# Load the dataset

data = pd.read_csv("multi_source_data.csv")

# Impute numerical columns

numerical_cols = data.select_dtypes(include=['float64', 'int64']).columns

data[numerical_cols] = data[numerical_cols].fillna(data[numerical_cols].median())

# Impute categorical columns

categorical_cols = data.select_dtypes(include=['object']).columns

data[categorical_cols] = data[categorical_cols].fillna('Unknown')
```

2.2 Managing Outliers

Outliers in integrated datasets can result from inconsistent data sources, affecting analysis quality:

- **Detection:** Visualized using boxplots and identified through **Z-score analysis** to highlight extreme values.
- **Treatment:**
  - **Winsorization:** Capped extreme values within the 1st and 99th percentiles to reduce impact.
  - **Exclusion:** Removed entries with clear evidence of integration issues (e.g., negative values in fields that should be positive).

Code Example:

```
# Managing outliers

import numpy as np

# Capping extreme values

data['Value'] = np.clip(data['Value'], data['Value'].quantile(0.01), data['Value'].quantile(0.99))


# Removing corrupted rows

data = data[data['Value'] > 0]
```

2.3 Resolving Duplicates and Inconsistencies

- **Duplicate records** from multiple sources were identified and removed to ensure unique entries.
- **Schema inconsistencies** such as different column formats across sources were standardized.
- **Logical inconsistencies**, such as mismatched timestamps or conflicting values across sources, were resolved.

Code Example:

```
# Removing duplicates

data = data.drop_duplicates()

# Standardizing column formats

data.columns = [col.strip().lower().replace(" ", "_") for col in data.columns]
```

---

3. Data Visualization

3.1 Tools for Visualization

To assess data integration quality, the following Python libraries were utilized:

- **Matplotlib:** For foundational static plots to analyze data distribution and completeness.
- **Seaborn:** For correlation heatmaps and feature relationships across multiple data sources.
- **Plotly:** For interactive exploration of integration inconsistencies and anomalies.

3.2 Key Visualizations and Insights

Data Completeness Heatmap

- Visualized missing data patterns across multiple sources to identify inconsistencies in integration.

Schema Consistency Check

- Boxplots and histograms were used to compare numerical distributions across different data sources, revealing mismatches in format or scale.

Correlation Heatmap

- Showed relationships between attributes from different sources, highlighting inconsistencies in expected correlations.

Duplicate and Anomaly Detection

- Scatterplots were used to detect duplicate records and inconsistencies across sources, such as mismatched timestamps or redundant entries.

Example Code for Visualizations:

```
import matplotlib.pyplot as plt

import seaborn as sns

# Correlation heatmap for multi-source data consistency

plt.figure(figsize=(10, 8))

sns.heatmap(data.corr(), annot=True, cmap='coolwarm')

plt.title("Correlation Heatmap of Integrated Data")

plt.show()

# Boxplot for comparing numerical distributions across sources

plt.figure(figsize=(10, 6))

sns.boxplot(data=data, orient='h')

plt.title("Distribution of Numerical Features Across Sources")

plt.show()
```

---

4. Model Research and Selection Rationale

4.1 Research into Techniques

Based on the characteristics of multi-source integrated data, the following techniques were evaluated for assessing data integration quality:

1. **Anomaly Detection with Isolation Forest:**
    - Chosen for its ability to identify inconsistencies and integration anomalies in high-dimensional datasets.
    - Effective in detecting duplicate, missing, or misaligned records.

2. **Autoencoder Neural Networks:**
    - Tested for learning latent representations of properly integrated data.
    - Struggled with varying quality across multiple sources, requiring extensive tuning.

3. **Rule-Based Data Validation (Constraint Checking):**
    - Applied schema and consistency rules to detect integration errors.
    - Useful for predefined quality metrics but lacked adaptability to unseen errors.

Justification for Isolation Forest:

- **Robustness:** Effectively identified inconsistencies, missing links, and duplicate records.
- **Scalability:** Performed efficiently on large multi-source datasets with varying structures.
- **Interpretability:** Provided anomaly scores to rank potential integration quality issues.

---

5. Data Transformation and Feature Engineering

5.1 Feature Scaling

- **Standardization:** Applied to ensure numerical features from different sources have a common scale.
- **Min-Max Scaling:** Used to normalize attributes with varying ranges between 0 and 1.

Code Example:

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler

# Standardization

scaler = StandardScaler()

data['Value_Standardized'] = scaler.fit_transform(data[['Value']])

# Min-Max Scaling

data['Feature_Scaled'] = MinMaxScaler().fit_transform(data[['Feature']])
```

5.2 Encoding Categorical Variables

- **One-Hot Encoding:** Transformed categorical attributes from multiple sources into interpretable binary features.

Code Example:

```
# One-Hot Encoding

import pandas as pd


encoded_data = pd.get_dummies(data, columns=['SourceSystem', 'CategoryFeature'])
```

5.3 Dimensionality Reduction

- **PCA (Principal Component Analysis):** Applied to reduce redundancy while retaining key information in integrated data.

Code Example:

```
from sklearn.decomposition import PCA

# Applying PCA to reduce dimensions while preserving 95% variance

pca = PCA(n_components=0.95)

data_pca = pca.fit_transform(data.drop(columns=['Target']))
```

---

6. Feasibility Assessment

6.1 EDA Results

- **Hypotheses:** Formed based on observed integration inconsistencies, such as missing values, schema mismatches, and duplicate records.
- **Algorithm Testing:** Conducted mock anomaly scoring using Isolation Forest to simulate integration quality detection.
- **Business Alignment:** Insights aligned with common data integration challenges in multi-source analytics, such as inconsistency in timestamp formats and variations in categorical values.

6.2 Metrics for Future Evaluation

- **Data Consistency Score:** Measures schema conformity and alignment across sources.
- **Completeness Ratio:** Quantifies the percentage of missing or imputed values.
- **Precision and Recall (for Anomaly Detection):** Ensures data integration issues are flagged accurately.

---

7. Conclusion

Phase 1 established a comprehensive foundation for assessing **data integration quality** by analyzing inconsistencies through **EDA and visualization**. Research into modeling techniques highlighted **Isolation Forest** as the optimal choice for anomaly detection in multi-source data.

Lessons Learned

- **EDA Importance:** Early-stage visualizations revealed key integration issues such as missing values and format mismatches.
- **Iterative Cleaning:** Improved data quality, ensuring reliability in downstream analytics.
- **Model Research:** Comparative evaluations clarified the effectiveness of different anomaly detection approaches for integration assessment.