Data Exploration and Solution Planning

Phase 2 Solution Architecture

College Name: Maratha Mandal Engineering College

Team Contributions:

- *Mohammed Sayeel Shigganvi* (CAN ID: CAN_33860603): Data cleaning strategies, API design, and additional contributions to model research and selection.
- *Abdussuban Shaboddin Patel* (CAN ID: CAN_34000109): Functional requirements, tools/platform selection, and evaluation metrics.
- *Mohammed Shaibaj Shaikh* (CAN ID: CAN_33990553): Visualization tools, feature engineering, and model evaluation.
- *Tufailahmed M Bargir* (CAN ID: CAN_34002247):): Reporting frameworks, dashboard design, and final documentation.

1. Overview of Bias Detection and Mitigation

Phase 1 documented the progress in understanding bias in AI training datasets, identifying patterns of imbalance, and planning a model design to address these issues. This phase focuses entirely on exploratory data analysis (EDA) and leveraging visualizations to refine our understanding and inform subsequent decisions without yet deploying the models.

Objectives:

- Develop visualizations to identify bias trends and potential imbalances.
- Gain actionable insights through EDA to guide model selection.
- Establish clear hypotheses and criteria for bias mitigation.

---

2. Data Cleaning and Preparation

2.1 Handling Missing Values Missing values were addressed as follows:

- **Numerical Features:** Imputed using the median to mitigate outlier effects.
- **Categorical Features**: Assigned a placeholder value "Unknown" for missing categories, ensuring these entries were preserved for analysis.

Code Example:

```
import pandas as pd
```

```
# Load dataset
data = pd.read_csv("training_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 were identified and addressed to avoid skewed results:

- **Detection:** Visualized using boxplots and identified through Z-score analysis.
- **Treatment**:
    - **Winsorization:** Capped extreme values within the 99th percentile.
    - **Exclusion**: Removed instances with clear evidence of data corruption.

Code Example:

```
import numpy as np
# Cap extreme values
data['Feature'] = np.clip(data['Feature'], data['Feature'].quantile(0.01),
data['Feature'].quantile(0.99))
```

2.3 Resolving Duplicates and Inconsistencies

- Duplicate records were flagged and removed.
- Logical inconsistencies, such as invalid feature values, were corrected.

Code Example:

```
# Removing duplicates
data = data.drop_duplicates()
```

3. Data Visualization

3.1 Tools for Visualization To facilitate effective data analysis, the following Python libraries were employed:

- ***Matplotlib*:** For foundational static plots.
- ***Seaborn*:** For correlation heatmaps and detailed visualizations.
- ***Plotly***: For dynamic and interactive visual dashboards.

3.2 Key Visualizations and Insights

- Correlation Heatmap: Highlighted relationships between features like class imbalances and feature distributions.
- Scatterplots: Pinpointed clusters of anomalies related to biased distributions.
- Boxplots: Visualized feature distributions to confirm patterns of imbalance.

Example Code:

```
import matplotlib.pyplot as plt

import seaborn as sns

# Correlation Heatmap

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

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

plt.title("Correlation Heatmap")

plt.show()
```

---

4. Model Research and Selection Rationale

4.1 Research into Techniques Based on the dataset's characteristics, the following techniques were evaluated:

1. **Decision Trees:** Selected for their ability to handle non-linear relationships and identify key features contributing to biases.

2. **Logistic Regression:** Evaluated for detecting linear relationships among biased variables.
3. **Clustering Algorithms**: Explored for their capability to group similar instances and highlight anomalies.

4.2 Justification for Selected Models

- ***Decision Trees*:** Chosen for their robustness, interpretability, and scalability.
- ***Logistic Regression***: Utilized as a complementary method for linear bias evaluation.

---

5. Data Transformation and Feature Engineering

5.1 Feature Scaling

- **Standardization**: Applied to ensure uniform feature influence.
- **Min-Max Scaling**: Used to normalize skewed features between 0 and 1.

Code Example:

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler

# Standardization

scaler = StandardScaler()

data['Standardized_Feature'] = scaler.fit_transform(data[['Feature']])

# Min-Max Scaling

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

5.2 Encoding Categorical Variables

- One-Hot Encoding preserved interpretability by creating independent binary features.

Code Example:

```
# One-Hot Encoding

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

5.3 Dimensionality Reduction

- PCA: Reduced dimensions while retaining 95% variance.

Code Example:

```
from sklearn.decomposition import PCA

pca = PCA(n_components=5)

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

---

6. Feasibility Assessment

6.1 EDA Results

- **Hypotheses:** Formed based on observed bias patterns and imbalances.
- **Algorithm Testing**: Conducted mock scoring to simulate bias correction techniques.
- **Ethical Alignment**: Ensured solutions adhered to AI fairness standards.

6.2 Metrics for Evaluation

- Fairness Metrics: Statistical Parity Difference, Equal Opportunity.
- Model Metrics: Precision, Recall, and ROC-AUC.

---

7. Conclusion

Phase 2 provides a robust architecture for bias detection and mitigation, prioritizing scalability, usability, and fairness. Lessons learned emphasize the importance of EDA, iterative cleaning, and model selection.

Next Steps:

- Validate hypotheses with statistical tests and domain expert feedback.
- Finalize model training and implement an API for deployment.
- Prepare a deployment framework for real-time bias mitigation.