Project Title: Identifying Data Integration Quality for Multi – Source

Name:GEETHA PRIYA

CAN_ID: 33738378

Institution Name: Vemana Institute of Technology

Model Development and Evaluation

This project focuses on identifying and improving data integration quality for multi-source systems. The approach emphasizes data cleaning, transformation, and handling biases to enhance model performance across various tasks like anomaly detection and integration quality assessment. Iterative refinement ensures an efficient balance between performance and resource utilization.

Step 1: Advanced Data Cleaning

Effective data integration requires comprehensive cleaning to address missing values, outliers, and class imbalances across multiple data sources.

1.1 Handling Missing Values

K-Nearest Neighbor (KNN) imputation is employed to handle missing values by leveraging neighboring data points to enhance data completeness.

python

```
from sklearn.impute import KNNImputer

data_imputer = KNNImputer(n_neighbors=5)

data_cleaned = pd.DataFrame(data_imputer.fit_transform(data), columns=data.columns)
```

1.2 Outlier Detection

Isolation Forest identifies and removes outliers that may distort integration accuracy.

python

```
from sklearn.ensemble import IsolationForest

iso = IsolationForest(contamination=0.01, random_state=42)

data_cleaned['Anomaly'] = iso.fit_predict(data_cleaned.drop(columns=['target']))

data_cleaned = data_cleaned[data_cleaned['Anomaly'] == 1].drop(columns=['Anomaly'])
```

1.3 Addressing Imbalanced Classes

Synthetic Minority Oversampling Technique (SMOTE) balances class distributions for better data integration results.

python

```
from imblearn.over_sampling import SMOTE

smote = SMOTE(random_state=42, k_neighbors=5, sampling_strategy='minority')

X_resampled, y_resampled = smote.fit_resample(data_cleaned.drop(columns=['target']),
data_cleaned['target'])
```

Step 2: Building and Training Models

2.1 Initial Model with Decision Tree
A baseline Decision Tree model is used to evaluate data integration quality, helping identify important features.

python

```
from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier(max_depth=3)

model.fit(X_train, y_train)
```

2.2 Advanced Model with Random Forest
A Random Forest Classifier optimizes integration assessment while balancing resource usage.

python

```
from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier(random_state=42, n_estimators=30, max_depth=5, n_jobs=-1)

clf.fit(X_train, y_train)
```

Step 3: Leveraging AutoAI

AutoAI automates tasks like model selection and hyperparameter optimization for data integration models.

- **Setup:** Upload the data to IBM Watson Studio AutoAI.
- **Execution:** Run experiments to generate model pipelines.
- **Results:** Evaluate metrics such as accuracy, recall, and ROC AUC.

Step 4: Model Evaluation

Metrics Used:

1. **Accuracy:** Overall correctness of integration results.
2. **Precision & Recall:** Balancing false positives and negatives for better data integration decisions.
3. **ROC AUC:** Comprehensive classification performance assessment.

```
python

from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred))
```

Step 5: Results and Insights

1. **Decision Tree Performance:** Provided quick baselines but limited in handling complex data integration patterns.
2. **Random Forest Performance:** Enhanced accuracy and recall, ideal for multi-source integration tasks.
3. **SVM Model (AutoAI):** Delivered the best performance, emphasizing the utility of automated tools in multi-source integration projects.

Conclusion

The project highlights the significance of advanced data cleaning, balanced model building, and robust evaluation. AutoAI played a key role in optimizing the data integration process, ensuring reliable and fair results. This comprehensive approach provides actionable insights for improving multi-source data integration quality.