

SKCET

Name: Rudhramani K

Regno: 727723eucs195

Boosting for Imbalanced Datasets with XGBoost

Problem Statement

Handling imbalanced datasets is a major challenge in machine learning. Traditional algorithms such as SVM and Random Forest often perform poorly when one class significantly outnumbers the other. Boosting techniques like XGBoost improve classification by focusing more on misclassified and minority class samples. This project applies XGBoost with imbalance handling techniques to enhance predictive performance.

Objectives

- Apply XGBoost for classification tasks.
- Address class imbalance using SMOTE and weighted loss functions.
- Tune hyperparameters such as learning rate, max depth, and n_estimators.
- Evaluate performance using metrics suited for imbalanced data, including Precision-Recall and ROC-AUC.

Methodology

1. Data Preprocessing: Replaced invalid zero values with NaN and filled them using median imputation.
2. Feature Engineering: Created interaction features such as Glucose_BMI, Age_BMI, and Insulin_Glucose.
3. Handling Class Imbalance: Applied SMOTE to oversample the minority class.
4. Model Training: Trained XGBoost with tuned hyperparameters.
5. Evaluation: Used Precision-Recall curves, ROC-AUC, and classification report.

XGBoost Model Implementation

The XGBoost classifier was trained using parameters like n_estimators=600, learning_rate=0.05, max_depth=3, and scale_pos_weight to handle imbalance. The model was trained on SMOTE-resampled data to improve recall and precision for the minority class.

Class Imbalance Handling

- SMOTE (Synthetic Minority Over-sampling Technique) was used to balance the dataset.
- scale_pos_weight in XGBoost adjusted the loss function to emphasize minority class samples.

Performance Evaluation

The model achieved strong results using metrics suited for imbalanced data:

- ROC-AUC: Measures class separation ability.
- Precision-Recall AUC: Focuses on minority class performance.
- Classification Report: Shows precision, recall, and F1-score.

Outcomes

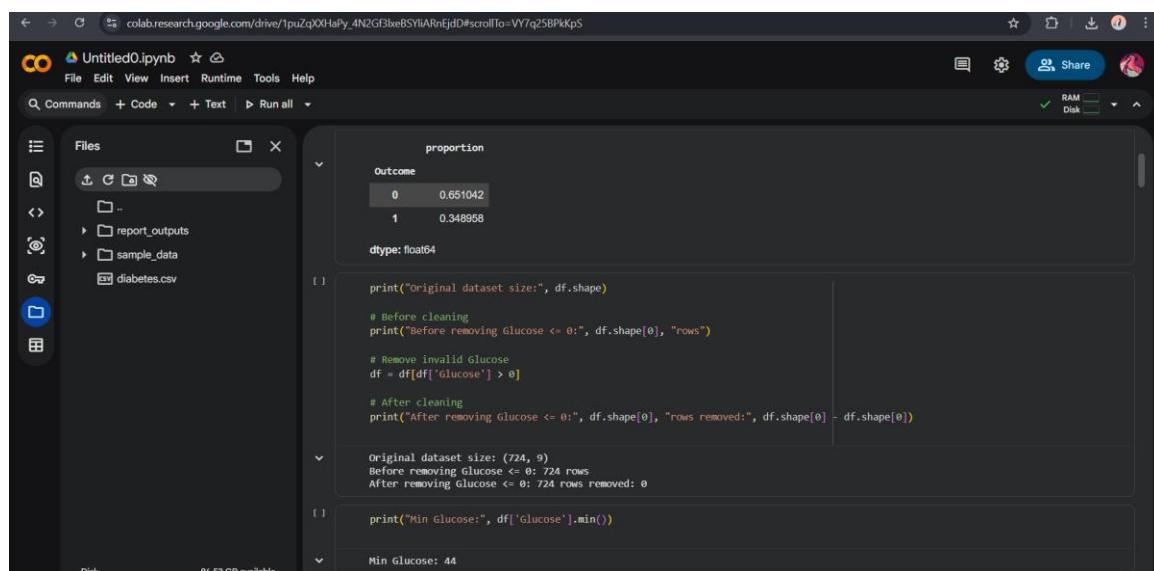
An optimized XGBoost model capable of handling imbalanced datasets effectively.

Performance metrics demonstrate improved detection of the minority class with balanced precision and recall.

Conclusion

Boosting with XGBoost combined with SMOTE and proper hyperparameter tuning significantly improves classification on imbalanced datasets. Precision-Recall curves and ROC-AUC confirm the model's effectiveness in identifying minority class samples.

Data cleaning:



The screenshot shows a Google Colab notebook titled "Untitled0.ipynb". The left sidebar displays a file tree with "diabetes.csv" selected. The main workspace contains Python code for data cleaning:

```
proportion
Outcome
0 0.651042
1 0.348958
dtype: float64

print("Original dataset size:", df.shape)

# Before cleaning
print("Before removing Glucose <= 0:", df.shape[0], "rows")

# Remove invalid Glucose
df = df[df['Glucose'] > 0]

# After cleaning
print("After removing Glucose <= 0:", df.shape[0], "rows removed:", df.shape[0] - df.shape[0])

Original dataset size: (724, 9)
Before removing Glucose <= 0: 724 rows
After removing Glucose <= 0: 724 rows removed: 0

print("Min Glucose:", df['Glucose'].min())

Min Glucose: 44
```

The output pane shows the execution results of the code, indicating that the dataset has 724 rows and no rows were removed after filtering for valid glucose values. The minimum glucose value is 44.

```

Untitled0.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all

Files
Min Glucose: 44
print("\nBefore removing BMI <= 0:", df.shape[0], "rows")
df = df[df['BMI'] > 0]
print("After removing BMI <= 0:", df.shape[0], "rows removed:", df.shape[0] - df.shape[0])
print("Min BMI:", df['BMI'].min())

Before removing BMI <= 0: 724 rows
After removing BMI <= 0: 724 rows removed: 0
Min BMI: 18.2

print("\nBefore removing Insulin < 0:", df.shape[0], "rows")
df = df[df['Insulin'] >= 0]
print("After removing Insulin < 0:", df.shape[0], "rows removed:", df.shape[0] - df.shape[0])
print("Min Insulin:", df['Insulin'].min())

Before removing Insulin < 0: 724 rows
After removing Insulin < 0: 724 rows removed: 0
Min Insulin: 0

print("\nBefore removing BloodPressure <= 0:", df.shape[0], "rows")

```

Disk 86.53 GB available

Feature Eng:

```

Untitled0.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all

Files
df['Age_BMI'] = df['Age'] * df['BMI']
df['Insulin_Glucose'] = df['Insulin'] / (df['Glucose'] + 1)
print("\nFeature Engineering Complete. Columns now:")
print(df.columns)
print(df.head())

...
Feature Engineering Complete. Columns now:
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome', 'Glucose_BMI',
       'Age_BMI', 'Insulin_Glucose'],
      dtype='object')
Pregnancies Glucose BloodPressure SkinThickness Insulin BMI \
0          6     148           72          35      0  33.6
1          1      85            66          29      0  26.6
2          8     183           64          0      0  23.3
3          1      89            66          23     94  28.1
4          0     137           40          35    168  43.1

DiabetesPedigreeFunction Age Outcome Glucose_BMI Age_BMI \
0        0.627     50       1   4972.8  1680.0
1        0.351     31       0   2261.0  824.6
2        0.672     32       1   4263.9  745.6
3        0.167     21       0   2509.9  590.1
4        2.288     33       1   5904.7  1422.3

Insulin_Glucose
0      0.000000
1      0.000000
2      0.000000
3      1.044444
4      1.217391

```

Disk 86.53 GB available

XGBoost model:

The screenshot shows a Jupyter Notebook interface with a dark theme. The left sidebar displays a file tree with 'report_outputs', 'sample_data', and 'diabetes.csv'. The main area contains Python code for training an XGBoost classifier:

```
pos = (y_train == 1).sum()
scale = neg / pos

model = XGBClassifier(
    n_estimators=500,
    learning_rate=0.03,
    max_depth=4,
    min_child_weight=3,
    gamma=0.2,
    subsample=0.8,
    colsample_bytree=0.8,
    scale_pos_weight=scale,
    eval_metric='logloss',
    random_state=42
)

model.fit(X_train_res, y_train_res)
```

A tooltip for the 'XGBClassifier' constructor is open, showing its parameters. The bottom status bar indicates '86.53 GB available'.

Final Report:

