Anomaly Detection Model Report

Submitted by – Aditya Sharma

aditya.glb15@gmail.com

Description of Design Choices and Performance Evaluation of the Model

1. **Data Preprocessing**

Handling Missing Values

Imputation: Missing values were handled by imputing the mean of the respective columns. This approach is simple and ensures that no data is lost, though it may not capture the underlying data distribution accurately.

Outlier Detection and Treatment

Winsorization: Outliers were treated using Winsorization, limiting extreme values to the 5th and 95th percentiles. This method reduces the impact of outliers on the model without removing data points.

2. Feature Engineering

Lag Features

Lag Feature Creation: A lag feature was created to capture temporal dependencies in the data. This can help the model understand patterns over time.

3. Feature Selection

Correlation Analysis

Selecting Relevant Features: Features with a correlation coefficient greater than 0.1 (in absolute value) with the target variable y were selected. This method ensures that only features with a significant relationship to the target are included in the model.

4. Model Selection

Random Forest Classifier

Choice of Model: A Random Forest Classifier was selected due to its robustness, ability to handle both numerical and categorical features, and its inherent feature importance measurement. Random Forests also tend to perform well with minimal tuning.

5. Model Training and Evaluation

Train-Test Split

Data Splitting: The dataset was split into training and testing sets with an 80-20 ratio using train_test_split with a random state of 42 for reproducibility.

Model Training

Training the Model: The Random Forest model was trained on the training set using default hyperparameters.

Model Predictions

Predictions: Predictions were made on the test set.

Performance Metrics

Accuracy: The model achieved an accuracy score of accuracy_value on the test set.

Discussion of Future Work

Improving Data Quality

Advanced Imputation Methods: Explore more sophisticated imputation techniques such as K-Nearest Neighbors (KNN) imputation or regression imputation to better handle missing values.

Enhanced Outlier Detection: Implement advanced outlier detection methods like Isolation Forest, DBSCAN, or Local Outlier Factor (LOF) to identify and handle outliers more effectively.

Feature Engineering

Temporal Features: Incorporate more temporal features such as rolling means, standard deviations, or exponentially weighted moving averages to capture more complex time-dependent patterns.

Model Improvement

Hyperparameter Tuning: Perform extensive hyperparameter tuning using Grid Search, Random Search, or Bayesian Optimization to find the optimal parameters for the Random Forest model.

Model Ensemble: Combine multiple models (e.g., Gradient Boosting, XGBoost, Neural Networks) to create an ensemble model that leverages the strengths of different algorithms for improved performance.

Source Code

```
import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from scipy.stats.mstats import winsorize

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, classification_report


# Load the dataset

data = pd.read_excel('AnomaData.xlsx')


# Display the first few rows of the dataset

print(data.head())

print(data.isnull().sum())


# Check the column names to ensure they exist

print(data.columns)


# Visualize the distribution of the target variable

sns.countplot(x='y', data=data)

plt.show()


# Explore other variables using histograms, box plots, etc.

sns.histplot(data[correct_column_name])

plt.show()
```

```
data.fillna(data.mean(), inplace=True)

# Outlier detection and treatment
data[correct_column_name] = winsorize(data[correct_column_name], limits=[0.05,
0.05])

data[f'{correct_column_name}_lag1'] = data[correct_column_name].shift(1)

# Feature selection
correlation_matrix = data.corr()
relevant_features = correlation_matrix['y'][abs(correlation_matrix['y']) >
0.1].index.tolist()

X = data[relevant_features]
y = data['y']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = RandomForestClassifier()

# Fit the model
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)

# Model evaluation
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

Classification report

print(classification_report(y_test, y_pred))

Result:

```
                 time  y        x1        x2        x3         x4        x5  \
0 1999-05-01 00:00:00  0  0.376665 -4.596435 -4.095756  13.497687 -0.118830
1 1999-05-01 00:02:00  0  0.475720 -4.542502 -4.018359  16.230659 -0.128733
2 1999-05-01 00:04:00  0  0.363848 -4.681394 -4.353147  14.127997 -0.138636
3 1999-05-01 00:06:00  0  0.301590 -4.758934 -4.023612  13.161566 -0.148142
4 1999-05-01 00:08:00  0  0.265578 -4.749928 -4.333150  15.267340 -0.155314

          x6        x7        x8  ...        x51        x52       x54  \
0 -20.669883  0.000732 -0.061114  ...  29.984624  10.091721 -4.936434
1 -18.758079  0.000732 -0.061114  ...  29.984624  10.095871 -4.937179
2 -17.836632  0.010803 -0.061114  ...  29.984624  10.100265 -4.937924
3 -18.517601  0.002075 -0.061114  ...  29.984624  10.104660 -4.938669
4 -17.505913  0.000732 -0.061114  ...  29.984624  10.109054 -4.939414

         x55        x56       x57       x58       x59       x60  y.1
0 -24.590146  18.515436  3.473400  0.033444  0.953219  0.006076    0
1 -32.413266  22.760065  2.682933  0.033536  1.090502  0.006083    0
2 -34.183774  27.004663  3.537487  0.033629  1.840540  0.006090    0
3 -35.954281  21.672449  3.986095  0.033721  2.554880  0.006097    0
```

```
         x55        x56       x57       x58       x59       x60  y.1
0 -24.590146  18.515436  3.473400  0.033444  0.953219  0.006076    0
1 -32.413266  22.760065  2.682933  0.033536  1.090502  0.006083    0
2 -34.183774  27.004663  3.537487  0.033629  1.840540  0.006090    0
3 -35.954281  21.672449  3.986095  0.033721  2.554880  0.006097    0
4 -37.724789  21.907251  3.601573  0.033777  1.410494  0.006105    0

[5 rows x 62 columns]
time    0
y       0
x1      0
x2      0
x3      0
       ..
x57     0
x58     0
x59     0
x60     0
y.1     0
```

```
Length: 62, dtype: int64
Index(['time', 'y', 'x1', 'x2', 'x3', 'x4', 'x5', 'x6', 'x7', 'x8', 'x9',
       'x10', 'x11', 'x12', 'x13', 'x14', 'x15', 'x16', 'x17', 'x18', 'x19',
       'x20', 'x21', 'x22', 'x23', 'x24', 'x25', 'x26', 'x27', 'x28', 'x29',
       'x30', 'x31', 'x32', 'x33', 'x34', 'x35', 'x36', 'x37', 'x38', 'x39',
       'x40', 'x41', 'x42', 'x43', 'x44', 'x45', 'x46', 'x47', 'x48', 'x49',
       'x50', 'x51', 'x52', 'x54', 'x55', 'x56', 'x57', 'x58', 'x59', 'x60',
       'y.1'],
      dtype='object')
```