Project Overview: Anomaly Detection in Network Traffic Using Isolation Forest and Deep Learning

This code is aimed at detecting anomalies within a synthetic network traffic dataset. By using a hybrid approach with both an Isolation Forest (an unsupervised learning method) and a neural network model, the code addresses anomaly detection by initially isolating potential outliers and then classifying them as anomalies.

Code Breakdown and Explanation

1. Importing Libraries

- **TensorFlow and Keras** are used for building the neural network model.
- **Pandas and NumPy** handle data manipulation and numerical operations.
- **Matplotlib and Seaborn** visualize the model performance through ROC curves and confusion matrices.
- **Scikit-Learn (sklearn)** provides the Isolation Forest algorithm for anomaly detection, along with data preprocessing, feature scaling, and performance evaluation utilities.

2. Loading and Exploring the Data

- The code loads the dataset (`synthetic_network_traffic.csv`) and examines it for duplicates and missing values, which can influence model performance.
- It then prints the first 30 rows to understand the data structure and checks the unique values of the target variable (`IsAnomaly`) to verify binary labeling (normal vs. anomaly).

3. Data Preprocessing

- **Min-Max Scaling**: Columns such as `BytesSent`, `BytesReceived`, `PacketsSent`, `PacketsReceived`, and `Duration` are normalized using the `MinMaxScaler`. This ensures that each feature has values between 0 and 1, enhancing the model's learning performance by preventing bias towards larger-scaled features.
- **Feature Engineering**: New features, `TotalBytes` and `TotalPackets`, are created by summing `BytesSent` with `BytesReceived` and `PacketsSent` with `PacketsReceived`. These features provide additional insights into the total data flow and may help improve model accuracy.

4. Splitting the Data

- The dataset is split into training, validation, and testing sets. Here, `train_test_split` splits the data into 70% training and 30% testing (divided further into validation and test sets at 15% each).
- `X` (features) and `y` (target) are separated for ease of model training and evaluation.

5. Anomaly Detection Using Isolation Forest

- An **Isolation Forest** model is trained on the `X_train` set. This algorithm isolates observations by randomly selecting features and split values, effectively separating anomalies in the dataset.
- **Prediction Transformation**: The model's prediction of -1 indicates an anomaly, while 1 indicates normal behavior. This output is then converted into binary values for consistency in further model training.

6. Neural Network Model Architecture

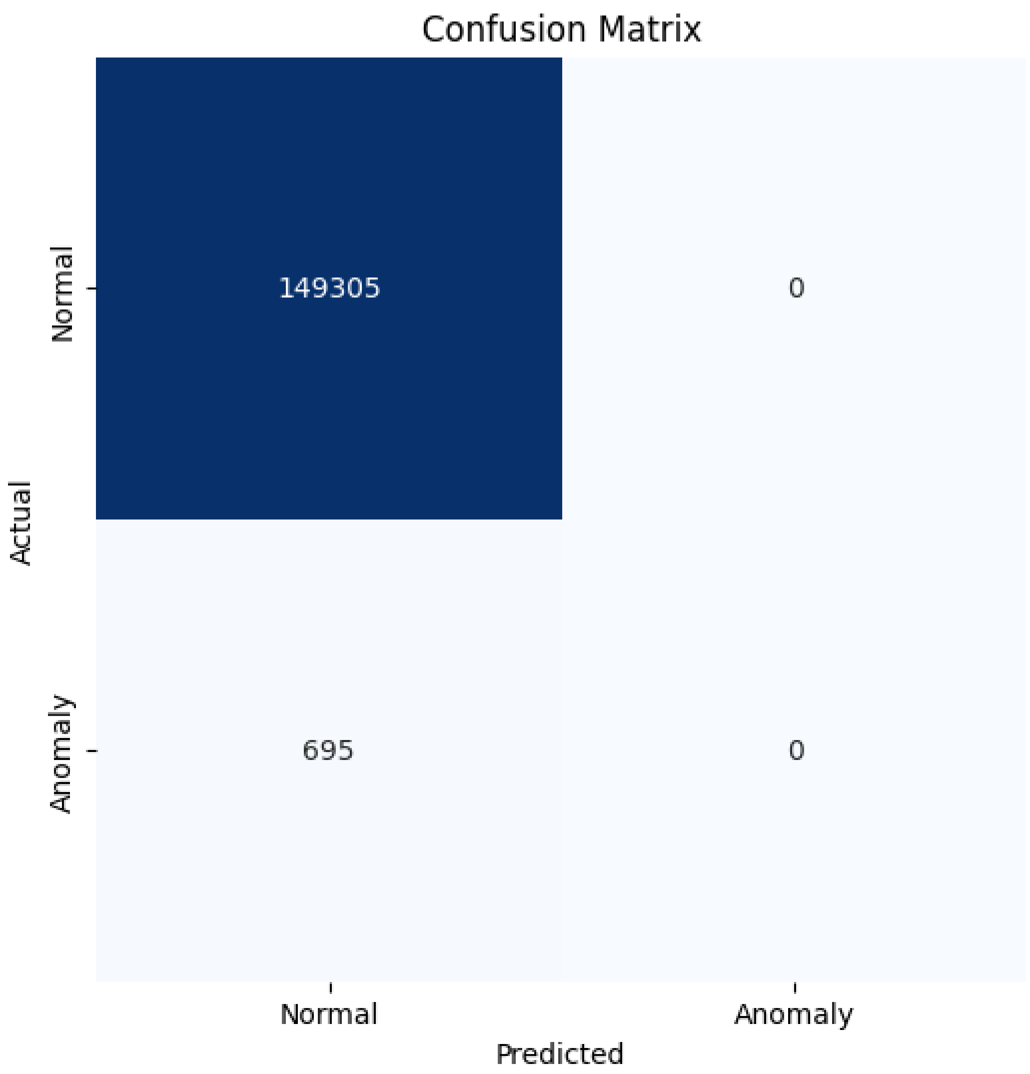
- A deep learning model with a simple feed-forward architecture is constructed:
  - **Input Layer**: Matches the feature dimensions.
  - **Hidden Layers**: Two layers with 64 and 32 neurons, both using the ReLU activation function.
  - **Output Layer**: A single neuron with a sigmoid activation function, outputting a probability for binary classification.
- **Compilation**: The model is compiled with `binary_crossentropy` loss (suitable for binary classification) and `adam` optimizer for efficient learning.

7. Model Training and Evaluation

- **Training**: The model is trained for 5 epochs with a batch size of 32. During training, validation data (`X_val` and `y_val`) help monitor performance and prevent overfitting.
- **Prediction**: After training, predictions are made on the `X_test` dataset, using a threshold of 0.5 to classify outputs as normal or anomaly.

8. Evaluation Metrics

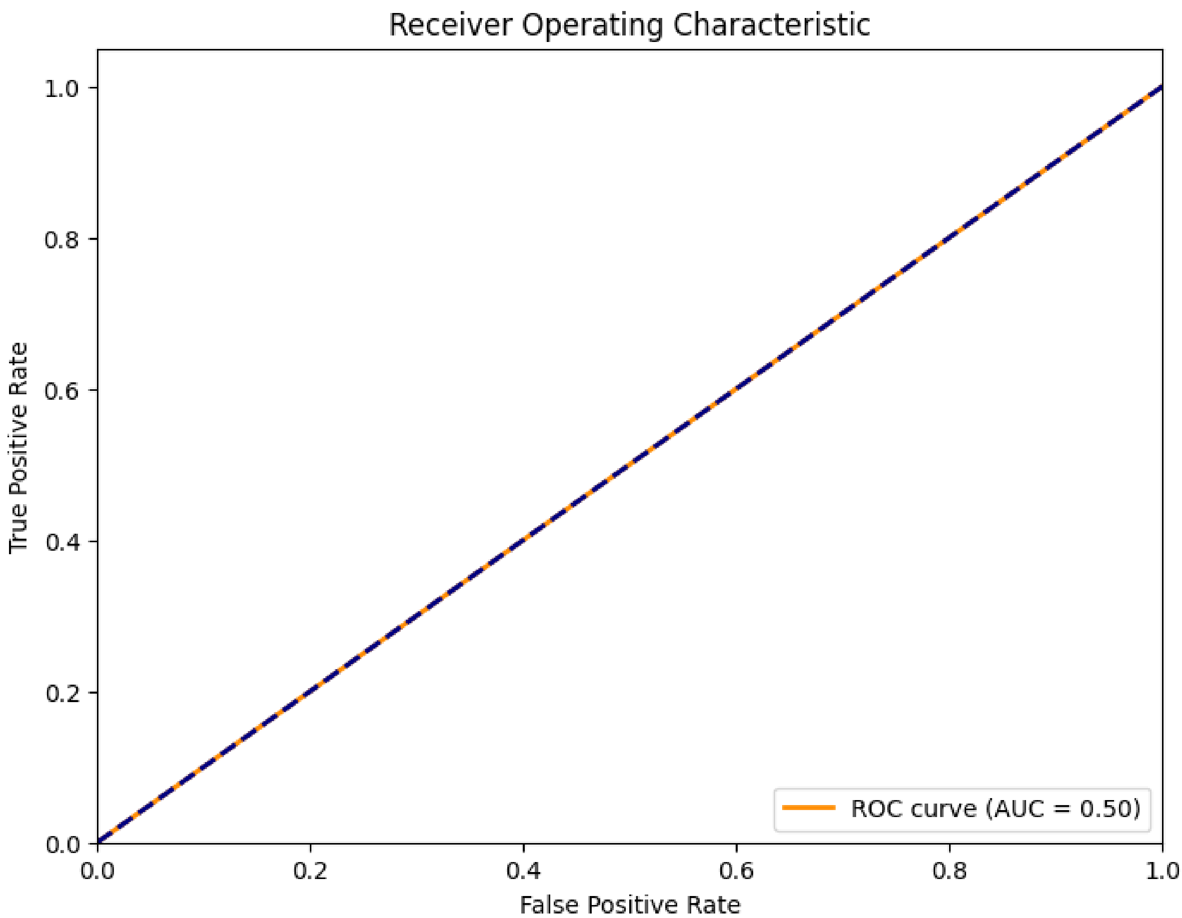
- **Confusion Matrix**: This matrix shows the number of true positives, true negatives, false positives, and false negatives, providing insight into model performance.



- **Classification Report**: This report includes precision, recall, F1-score, and support for each class, offering a comprehensive overview of model efficacy.

	precision	recall	f1-score	support
Normal	1.00	1.00	1.00	149305
Anomaly	1.00	0.00	0.00	695
accuracy			1.00	150000
macro avg	1.00	0.50	0.50	150000
weighted avg	1.00	1.00	0.99	150000

- **ROC Curve and AUC**: The ROC curve displays the trade-off between the true positive rate and false positive rate. AUC (Area Under Curve) quantifies the model's overall ability to distinguish between classes.



9. Visualization

- **ROC Curve**: A plot of the false positive rate (x-axis) versus true positive rate (y-axis) with AUC included as an indicator of the model's performance.
- **Confusion Matrix**: A heatmap visualization of the confusion matrix, highlighting correctly and incorrectly classified instances, enhancing interpretability.

Summary and Methodology

The code adopts a hybrid approach for anomaly detection in network traffic data. It combines an **Isolation Forest model**, which detects outliers in an unsupervised fashion, with a **supervised neural network** to further classify the anomalies.

1. **Data Normalization** and **Feature Engineering** play critical roles in enhancing model performance.
2. **Isolation Forest** identifies anomalies based on distance from other data points, which are then labeled for training the deep learning model.
3. **Neural Network Model** processes labeled data to learn patterns and classify data points as anomalies or normal instances.
4. **Evaluation Metrics** like ROC-AUC and confusion matrices provide insights into model accuracy and help in interpreting how well the model performs in real-world scenarios.

This methodology, combining unsupervised anomaly detection with a supervised classifier, effectively boosts the model's ability to recognize and classify anomalous behavior in synthetic network traffic data.