Documentation for Blockchain-Based Liquidity Risk Prediction Model

Model Overview

This model is developed to predict the likelihood of liquidation events for loans taken on decentralized finance (DeFi) platforms. Liquidation occurs when the value of the collateral deposited for a loan drops relative to the borrowed amount, triggering a liquidation threshold. The model aims to predict such events, providing risk management insights for DeFi lending protocols. It uses various cryptocurrency-related data, including loan and asset prices, to simulate and estimate liquidation events.

Model Architecture

The core model architecture is based on the **XGBoost Classifier**. XGBoost is a highly efficient, decision-tree-based ensemble machine learning algorithm that implements gradient boosting. It is designed to handle structured/tabular data and excels in both classification and regression tasks. For this classification problem, it predicts whether a liquidation event occurs (binary outcome).

- **Key Model Parameters**:
  - **n_estimators**: 1000 (number of boosting rounds)
  - **learning_rate**: 0.2 (the step size at each iteration while moving toward a minimum of the loss function)
  - **max_depth**: 3 (the maximum depth of the trees)
  - **colsample_bytree**: 1 (the fraction of features to consider during each split)
  - **gamma**: 1 (the minimum loss reduction required to make a further partition on a leaf node)
  - **random_state**: 42 (to ensure reproducibility and consistency in results)

XGBoost constructs weak learners (decision trees) iteratively, improving their performance by correcting errors made in the previous rounds. This boosting approach increases the model's overall accuracy, especially in imbalanced classification problems, where one class (liquidation events) is underrepresented compared to the non-liquidation class.

Training Process

1. **Data Preprocessing**:
   - **Time-based Feature Extraction**: Features related to loan start dates, such as the year, month, and day, are extracted from the raw dataset.
   - **Standardization**: Numerical features were standardized using **StandardScaler** to ensure that they all share a common scale, which benefits models like XGBoost that are sensitive to feature scaling.
2. **Handling Class Imbalance**:
   - Given that liquidation events are rare in comparison to non-liquidations, the dataset is imbalanced. To address this, we use **RandomOverSampler** to oversample the minority class (liquidation events) in the training data. This ensures that the model learns to identify liquidation risks effectively.

3. **Model Training**:
    - The oversampled, preprocessed, and standardized dataset is used to train the XGBoost model. It iteratively improves its performance through 1000 boosting rounds, optimizing the log loss (detailed below).
    - **Boosting**: Each tree added by XGBoost focuses on correcting the errors made by the previous trees, leading to stronger performance as the boosting iterations continue.
4. **Prediction**:
    - Once trained, the model predicts the probability of a loan being liquidated. This probability is output as part of the test dataset evaluation, which provides insights into the likelihood of liquidation for new loans.

Performance Metric – Log Loss

The model's performance is evaluated using the **log loss** metric, which is commonly used in binary classification problems. Log loss measures how uncertain the model's predicted probabilities are, penalizing incorrect predictions more heavily.

The formula for log loss is:

$$\text{Log Loss} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(p_i) + (1 - y_i)\log(1 - p_i)]$$

Where:

- N is the number of samples.
- y_i is the actual binary outcome (1 for liquidation, 0 for no liquidation).
- p_i is the predicted probability of a sample belonging to the liquidation class.

A lower log loss value indicates a better-performing model. It accounts for how well the predicted probabilities match the actual labels, providing insight into both classification accuracy and prediction confidence.

Model Application

This model specifically predicts liquidation risks for loans on DeFi platforms using assets like WETH, WBTC, and USDC as collateral and borrowed amounts. By analyzing the Health Factor and price changes in these assets over time, the model provides a probability estimate of whether a loan will be liquidated. This information can be used by lenders or borrowers to mitigate risks and optimize asset management strategies.

Model Setup

1. **Clone the Repository:**

   Run git clone https://github.com/SirioFinance/MLModel.git and then change directory to MLModel.

2. **Create and Activate a Virtual Environment:**

   Use python -m venv venv to create a virtual environment. Activate it with

   source venv/bin/activate (on Windows, use venv\Scripts\activate).

3. **Install Required Packages:**

   Install the packages using pip install -r requirements.txt.

4. **Ensure the Script Path:**

   Make sure your script liquidity_model.py is located at /Users/[*username*]/MLModel/Code/liquidity_model.py or update the path as needed.

Model Usage

1. **Training the Model**

   To train the model and save it, use the following command:

   python /Users/[*username*]/MLModel/Code/liquidity_model.py /path/to/dataset.csv

   Replace /path/to/dataset.csv with the path to your dataset.

2. **Making Predictions**

   To make predictions on new data, use the --predict argument:

   python /Users/[*username*]/MLModel/Code/liquidity_model.py /path/to/dataset.csv --predict /path/to/new_data.csv

   Replace /path/to/new_data.csv with the path to the new data file.

Alternatively, run model using Docker:

1. **Build the Docker Image**

   To set up the Docker image for the liquidity risk model, navigate to the project directory (where the Dockerfile is located) and run the following command:

   docker build -t liquidity-model .

2. **Run the Docker Container with the Training Dataset**

   Once the image is built, run the container while passing in the dataset for model training. Make sure to map the dataset from your local machine to the Docker container:

   docker run -it --name liquidity-model-container -v /path/to/your/dataset.csv:/app/data/dataset.csv liquidity-model python liquidity_model.py /app/data/dataset.csv

3. **Making Predictions on New Data**

   To predict with new data using the trained model, run the following command:

   docker run -it --name liquidity-model-container -v /path/to/your/new_data.csv:/app/data/new_data.csv liquidity-model python liquidity_model.py /app/data/dataset.csv --predict /app/data/newdata.csv

   Replace /path/to/new_data.csv with the path to the new data file.

Conclusion

The **XGBoost Classifier** employed for this task provides an effective tool for predicting liquidation events in DeFi lending. With its robust architecture, detailed training process, and the use of log loss for performance measurement, the model is well-equipped to assist in managing liquidity risks, ultimately contributing to more efficient and safer lending practices on blockchain platforms.