Detecting Arbitrage Opportunities in S&P 500 Options Using Unsupervised Deep Learning

Introduction to the Project

- Arbitrage Opportunities and Challenges
- Traditional Methods Are Ineffective
- Proposed Solution: Unsupervised Deep Learning
- Key Features for Anomaly Detection
- Why Unsupervised Learning?

Dataset

- The dataset comes from the **OptionsDX platform**
- Key Data Components
- Implied Volatility
- Market Factors and Granularity
- Real-Time Analysis Potential

Project Objective and Goals

- Design an unsupervised deep learning model to detect pricing anomalies
- Identifying Hidden Patterns
- Minimizing False Positives
- Uncovering Predictive Features
- Capitalize on risk-free profit opportunities, enhancing trading decision-making and strategy.

Project Objective and Goals

- Data Collection and Preprocessing
- Feature Engineering
- Develop an unsupervised deep learning model
- Operating Without Labels
- Model Validation
- Model Evaluation
- Operating Without Labels
- Refine the model to improve performance

Challenges and Questions

- Effectiveness of Unsupervised Learning
- Feature Selection
- Handling High-Frequency Data
- Model Validation Without Labels
- Developing Robust Solutions

Introduction to Options Trading

- **Definition of Options (financial derivatives, call/put rights)**
- **Factors Affecting Premium (strike price, time, volatility).**
- **Options Greeks (price sensitivity metrics).**
- **Uses of Options (hedging, speculation, arbitrage).**
- **Machine Learning Role (hidden patterns, anomaly detection).**

Dataset Features and Their Significance

- Call and Put Options Data (pricing, strike price, discrepancies)
- Options Greeks (delta, gamma, vega, theta sensitivity).
- Implied Volatility (IV) (expected price fluctuations, anomalies)
- Market Factors (S&P 500 levels, time, volume)
- Combined Analysis (patterns, mis-pricing detection, arbitrage)

How Machine Learning Can Help

- **Pattern Recognition** (detect clusters, identify anomalies)
- **Feature Combination** (hidden relationships, Greeks, volatility)
- **Real-Time Analysis** (high-frequency, quick anomaly detection)
- **Anomaly Detection** (autoencoders, clustering algorithms)
- **Adaptability** (scalable, retrains with new data)

Code Overview and Conclusion

- **Essential Libraries** (Pandas, NumPy, Matplotlib for data handling and visualization)
- **Feature Scaling** (MinMaxScaler ensures proportional input contributions)
- **Model Construction** (TensorFlow, Dense layers, Adam optimizer, EarlyStopping)
- **Performance Evaluation** (Mean Squared Error flags anomalies through reconstruction errors)
- **Conclusion** (Machine learning uncovers pricing patterns, detects anomalies, and identifies arbitrage opportunities)

Dataset Loading and Initial Exploration

- **Loading the Dataset** (Pandas, efficient data handling)
- **Data Preview** (df.head() to check structure, features, and consistency)
- **Understanding Key Features** (strike price, expiration date, options Greeks)
- **Identifying Anomalies** (early detection of inconsistencies in the data)
- **Foundation for Machine Learning** (prepares data for cleaning, preprocessing, and model development)

Feature Selection and Data Cleaning

- **Selecting Numerical Features** (filter float 64 and int 64 columns for relevance)
- **Excluding Non-Numerical Data (**remove text and dates to avoid noise)
- **Handling Missing Values (**dropna() ensures clean, complete data**)**
- **Dataset Validation(**head() confirms filtered features and consistency)
- **Machine Learning Readiness** (refined data for pattern detection and anomaly identification)

Data Normalization with Min-Max Scaling

- **Importance of Normalization (**scales features to a consistent range [0, 1])
- **Using MinMaxScaler** (applies fittransform() to scale numerical data**)**
- **Why Scaling Matters** (prevents larger features from dominating neural networks)
- **Improved Model Performance** (balanced inputs enhance pattern detection and anomaly identification)

Autoencoder Model Implementation

- **Input and Latent Dimensions** (24 input features, bottleneck compressed to 4 dimensions)
- **Encoder Structure** (Dense layers: 16 → 8 → 4 neurons with ReLU activation)
- **Decoder Structure** (Mirrors encoder: 4 → 8 → 16 → 24 neurons with sigmoid activation)
- **Model Compilation** (Adam optimizer, learning rate 0.001, Mean Squared Error for loss)

Autoencoder Model Implementation

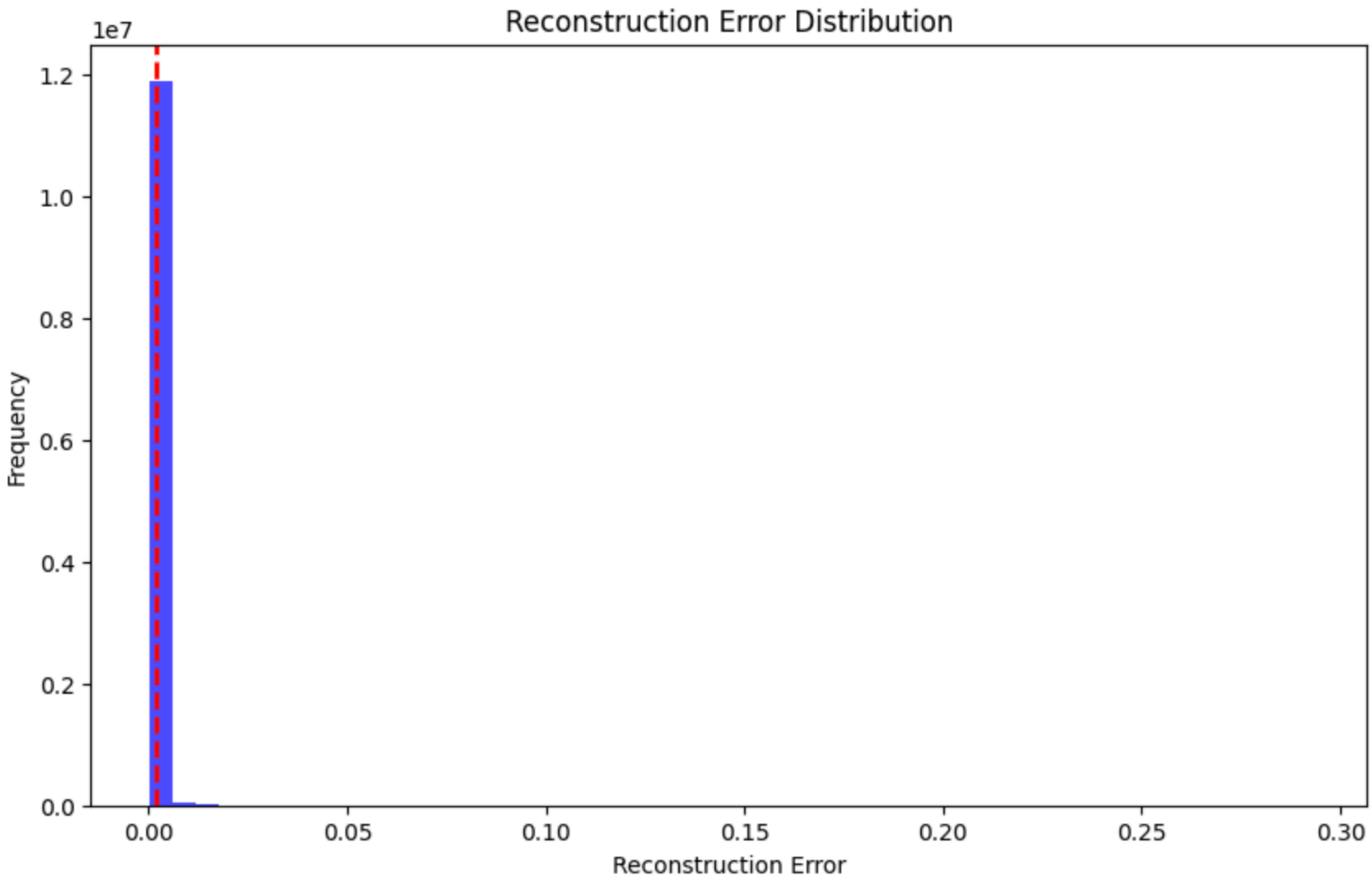
- **Input and Latent Dimensions** (24 input features, bottleneck compressed to 4 dimensions)
- **Encoder Structure** (Dense layers: 16 → 8 → 4 neurons with ReLU activation)
- **Decoder Structure** (Mirrors encoder: 4 → 8 → 16 → 24 neurons with sigmoid activation)
- **Model Compilation** (Adam optimizer, learning rate 0.001, Mean Squared Error for loss)

Train-Test Split

- **Defining Train Size** (80% of total data length using len() and int() for slicing).
- **Training Set Creation** (first 80% of data sliced as train_data)
- **Testing Set Creation** (remaining 20% of data sliced as test_data)
- **Purpose of the Split** (ensures no overlap, model trains on seen data and evaluates on unseen data)

EarlyStopping Callback Implementation

- **Monitoring Validation Loss**
- **Patience Parameter**
- **Restoring Best Weights**
- **Efficient Resource Use** (stops unnecessary epochs, saving time and computational resources)
- **Improved Generalization** (prevents overfitting, ensuring the model generalizes well to new data)



Reconstruction Error Distribution
1e7
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Frequency
0.00
0.05
0.10
0.15
0.20
0.25
0.30
Reconstruction Error

Visualizing Reconstruction Errors

- **Creating the Histogram** (plt.hist( ) visualizes the distribution of reconstruction errors with 50 bins).
- **Error Distribution** (shows how frequently different reconstruction errors occur, indicating model accuracy).
- **Setting the Threshold** (plt.axvline( ) adds a red dashed line to highlight the anomaly threshold).
- **Normal vs Abnormal Errors** (errors below the threshold are “normal”; those above indicate potential anomalies).
- **Plot Interpretation** (most errors close to zero signify accurate reconstruction, while outliers reveal poor reconstructions or anomalies).

Identifying and Displaying Anomalies

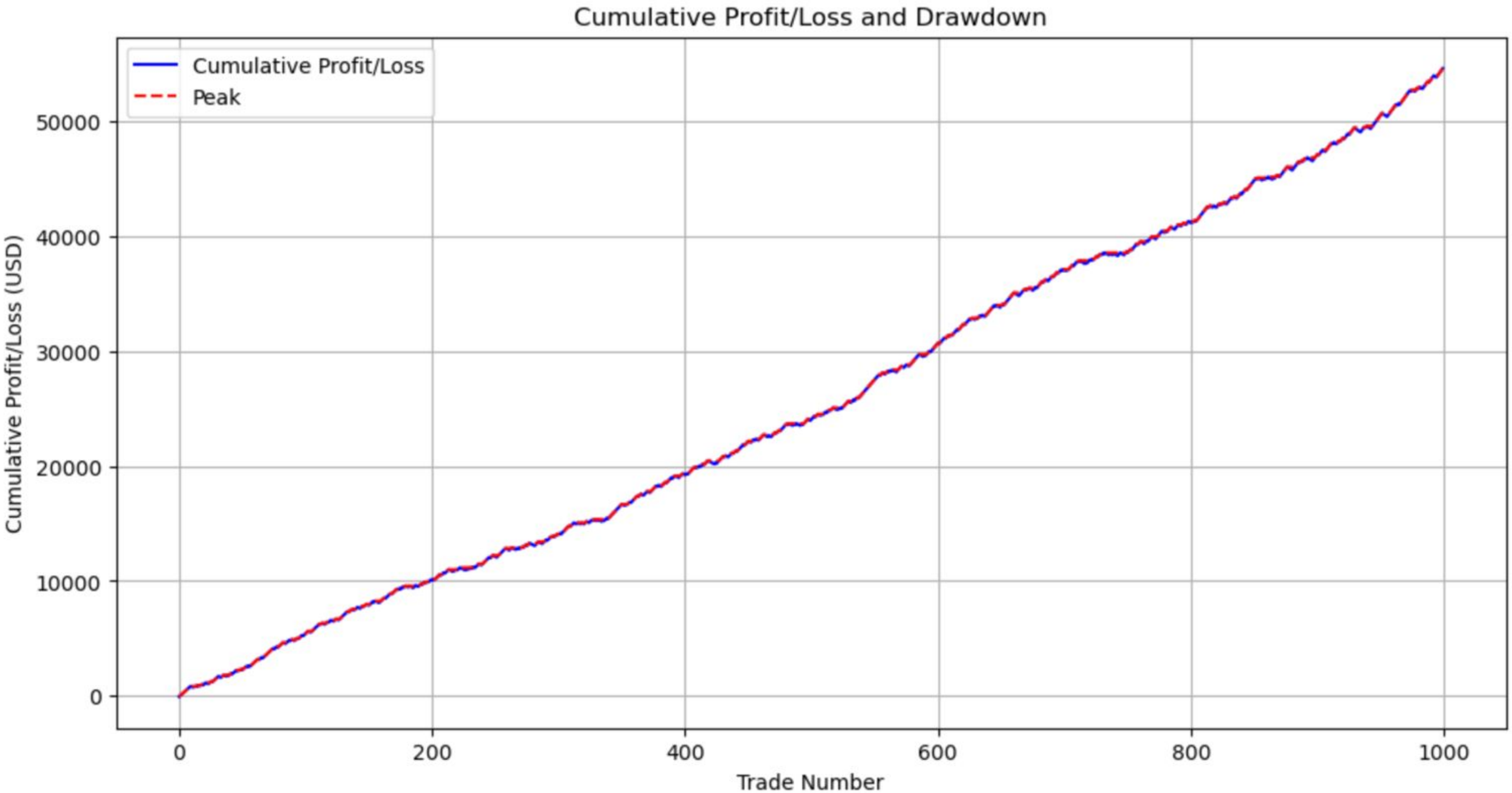
- **Filtering Anomalies** (filter rows where ‘Anomaly’ `== True` based on reconstruction error threshold)
- **Storing Flagged Data** (store anomalies in a new variable, anomalous_data)
- **Displaying Results** (head(`50`) displays the first 50 rows for quick analysis)
- **Purpose of Isolation** (isolates data points the autoencoder struggled to reconstruct accurately)
- **Validation and Insights** (helps validate the anomaly detection process and identify unusual patterns or outliers)

Motivation and Interest

- Personal and Professional Connection
- Shift in Trading Dynamics
- Developing a Practical Solution
- Excitement for Machine Learning
- Contributing to Quantitative Finance

```
Initial Capital: $100000
Final Capital: $154595.40
Total Return: $54595.40
Number of Trades: 1000
Average Return per Trade: $54.60
Volatility (Standard Deviation): $92.15
Sharpe Ratio: 0.59
Maximum Drawdown: $373.21
```

Anomalous dataset risk-returns



```
Initial Capital: $100000
Final Capital: $146690.44
Total Return: $46690.44
Number of Trades: 1000
Average Return per Trade: $46.69
Volatility (Standard Deviation): $91.17
Sharpe Ratio: 0.51
Maximum Drawdown: $496.13
```

Non- Anomalous dataset risk-returns