BERLIN SCHOOL OF BUSINESS & INNOVATION

University for the Creative Arts

Forecasting Crypto Market Movements Using CatBoost with Optimized Feature Selection and Tuning

DEGREE: MSc Data Analytics

Module: Fundamentals of Data Analytics

Lecturers: Dr. Md Roman Bhuiyan

Student: Sima Niaztalkhouncheh (Q1103265)

Contents

1. Introduction

This project aims to forecast short-term fluctuations in the cryptocurrency market using a supervised machine learning approach. The dataset was obtained from the Kaggle competition "DRW - Crypto Market Prediction" (Kaggle, 2025), which provided anonymized market microstructure data. The task was to predict the future return of assets based on a wide range of market features (over 890 columns). This report summarizes our data preprocessing, exploratory analysis, feature engineering, model development, and evaluation stages.

Background of the Competition

The cryptocurrency market represents one of the most dynamic and rapidly evolving financial landscapes, offering immense opportunities for those capable of identifying meaningful signals. However, due to its high volatility and low signal-to-noise ratio, predicting price movements in crypto is exceptionally challenging. Factors like order flow, liquidity, market sentiment, and inefficiencies make this a complex data science problem.

DRW, a pioneer in financial innovation, organized this competition to simulate the challenges their crypto trading arm, Cumberland, faces in real-time markets. Participants were tasked with building models to predict short-term future price movements using proprietary production features provided by DRW, complemented with public market volume statistics. The goal was to integrate diverse, noisy data into a reliable directional signal.

This competition provides a real-world test bed for applying advanced machine learning to financial prediction under high-dimensional and dynamic conditions.

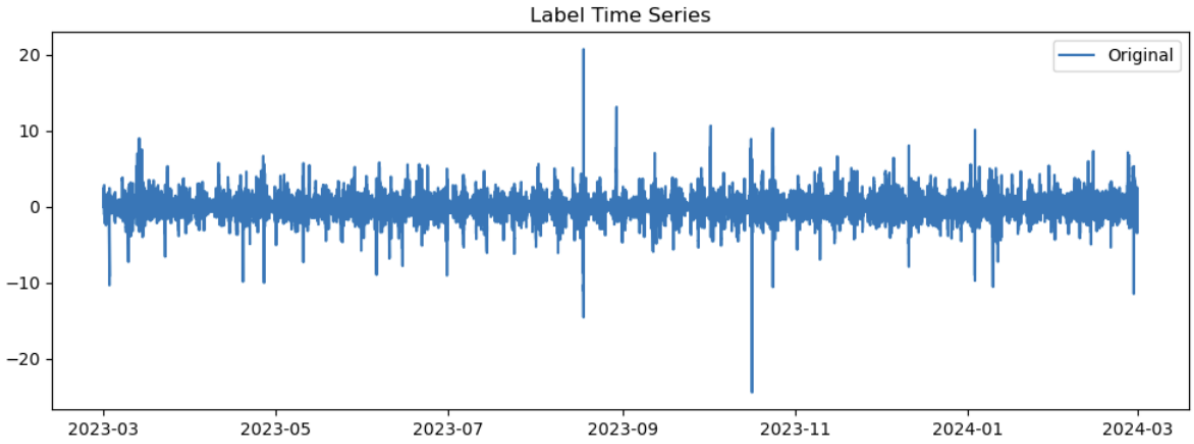
https://www.kaggle.com/competitions/drw-crypto-market-prediction/overview

2. Dataset Overview and Source

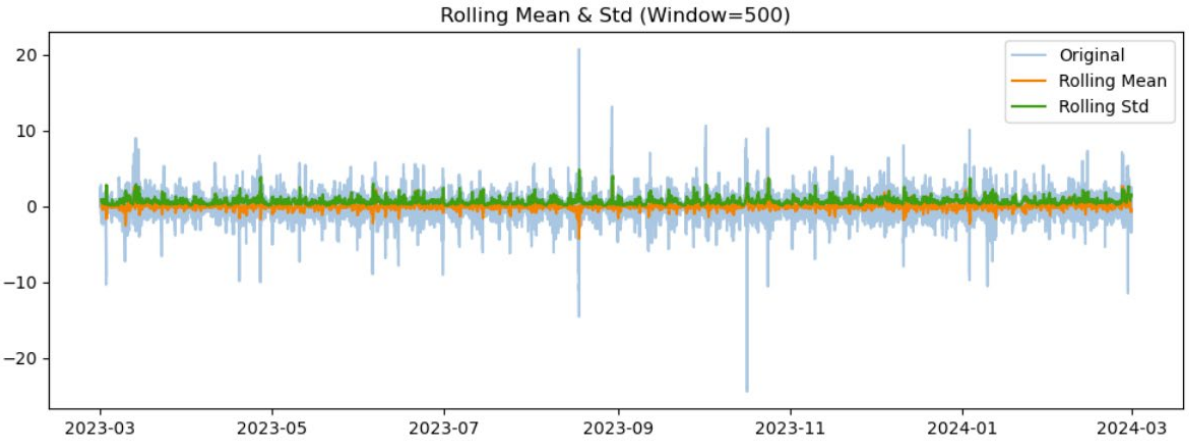
The dataset was sourced from Kaggle (DRW, 2025). It includes minute-level trading data such as bid/ask quantities, buy/sell volumes, and over 880 anonymized engineered features (X1-X890), as well as the target variable label representing short-term return. The training dataset consists of 525,887 observations.

Visualization of Target Variable

We visualized the label time series to examine trends, volatility, and potential non-stationarity. The first plot displays the raw series, highlighting fluctuations and possible outliers. The second plot overlays the rolling mean and rolling standard deviation (window size = 500), providing insight into the series' stability over time. Noticeable changes in these statistics would indicate non-stationarity, which was later formally tested using the Augmented Dickey-Fuller test.



Label Time Series



Rolling Mean & Std

3. Exploratory Data Analysis and Stationarity Check

Initial exploration included statistical summaries, distribution visualization, and correlation

analysis. We examined distributions of key market variables (e.g., bid_qty, ask_qty, volume) and found them to be heavily right-skewed with outliers, particularly in buy_qty and sell_qty. Correlation heatmaps were used to detect highly redundant features, though most anonymized features showed weak pairwise correlations.

We also assessed the target variable label for stationarity using the Augmented Dickey-Fuller (ADF) test, which yielded a p-value < 0.01, indicating that label is stationary and suitable for time-series modeling (Dickey and Fuller, 1979). We further identified and removed columns with infinite values and confirmed no significant missing data.

```
: from statsmodels.tsa.stattools import adfuller

# Apply the Augmented Dickey-Fuller test directly on the 'label' column (after dropping NaNs)
result = adfuller(df['label'].dropna())

# Print the test statistic and p-value
print("ADF Statistic:", result[0])
print("p-value:", result[1])

# Print the critical values for different significance levels (1%, 5%, 10%)
for key, value in result[4].items():
    print(f"Critical Value {key}: {value}")
```

```
ADF Statistic: -47.24897543425324
p-value: 0.0
Critical Value 1%: -3.4303624372499946
Critical Value 5%: -2.8615454971187004
Critical Value 10%: -2.566772925915377
```

Augmented Dickey-Fuller (ADF)

4. Model Suitability for the Task

Given that the target variable label is continuous and stationary, and the dataset contains high-dimensional time series data with financial characteristics (e.g., volatility, non-linearity), this task is best framed as a time series regression problem. Therefore, models such as CatBoost, XGBoost, LightGBM, and tree-based regressors are well-suited due to their robustness in handling non-linear relationships and large feature sets.

5. Model Evaluation Function

To ensure consistent evaluation across models, we defined a utility function evaluate_model() that prints four key regression metrics: Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R^2 Score. These metrics collectively measure prediction accuracy, error magnitude, and variance explanation, allowing for robust model comparison.

```python
#Model Evaluation Function
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import numpy as np

def evaluate_model(name, y_true, y_pred):
    print(f"📊 {name} Evaluation:")
    print("MAE:", mean_absolute_error(y_true, y_pred))
    print("MSE:", mean_squared_error(y_true, y_pred))
    print("RMSE:", np.sqrt(mean_squared_error(y_true, y_pred)))
    print("R2 Score:", r2_score(y_true, y_pred))
    print("-" * 40)
```

Model Evaluation Function

6. Preliminary Modeling Without Feature Engineering

Before applying any domain-specific feature engineering, we tested baseline machine learning models on the original dataset to assess their raw performance:

- Linear Regression: MAE = 345.10, RMSE = 516.77, R2 = -247288.56 (extremely poor due to feature scaling issues and high dimensionality)
- Decision Tree Regressor: MAE = 0.73, RMSE = 1.09, R2 = -0.116
- XGBoost Regressor: MAE = 0.83, RMSE = 1.22, R2 = -0.381
- LightGBM Regressor: MAE = 0.87, RMSE = 1.26, R2 = -0.468
- CatBoost Regressor: MAE = 0.78, RMSE = 1.13, R2 = -0.198

These results highlighted the need for more refined feature construction to improve learning signals.

Model	MAE	RMSE	R2
Linear Regression	345.10	516.77	-247288.56
Decision Tree Regressor	0.73	1.09	-0.116
XGBoost Regressor	0.83	1.22	-0.381
LightGBM Regressor	0.87	1.26	-0.468
CatBoost Regressor	0.78	1.13	-0.198

Comparison of Models (Before Feature Engineering)

7. Feature Engineering

To capture temporal and structural patterns, we engineered domain-specific features such as:

- Lag features: label_lag_1 to label_lag_5
- Rolling window metrics: mean and std for windows 3 and 5
- Volume dynamics: buy_sell_ratio, volume_diff, buy_volume_ratio, sell_volume_ratio

We then used SelectKBest with f_regression to select the top 100 most predictive features.

Feature Name	Description	Purpose / Motivation
label_lag_1 to label_lag_5	Previous 1 to 5 minute values of target variable label	Captures short-term temporal dependencies and momentum
label_roll_mean_3, label_roll_mean_5	Moving average of label over 3 and 5 minute windows	Smooths noise and reveals underlying short-term trend
label_roll_std_3, label_roll_std_5	Rolling standard deviation over 3 and 5 minute windows	Measures recent volatility in returns
buy_sell_ratio	Ratio of buy_qty to sell_qty	Indicates buying pressure vs selling pressure
volume_diff	First-order difference of total volume	Measures sudden liquidity changes
buy_volume_ratio	buy_qty divided by total volume	Detects buying dominance relative to total market activity
sell_volume_ratio	sell_qty divided by total volume	Detects selling dominance relative to total market activity

Feature Engineering Summary

8. Modeling Approach

We evaluated the following models using the engineered feature set:

- Linear Regression: RMSE = 516.77 (performed poorly due to scale sensitivity)
- Decision Tree Regressor: RMSE = 1.09

- XGBoost Regressor: RMSE = 1.22
- LightGBM Regressor: RMSE = 1.26
- CatBoost Regressor: RMSE = 1.13

We then used Optuna (Akiba et al., 2019) to optimize CatBoost hyperparameters:

```
import optuna
from catboost import CatBoostRegressor
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split

def objective(trial):
   # Define the hyperparameter search space for Optuna
    params = {
        'iterations': trial.suggest_int('iterations', 100, 300),
        'depth': trial.suggest_int('depth', 4, 10),
        'learning_rate': trial.suggest_float('learning_rate', 0.01, 0.3),
        'l2_leaf_reg': trial.suggest_float('l2_leaf_reg', 1, 10),
        'random_strength': trial.suggest_float('random_strength', 0.0, 1.0),
        'bagging_temperature': trial.suggest_float('bagging_temperature', 0.0, 1.0),
        'random_seed': 42,
        'verbose': 0
    }

    # Split the data for internal validation during tuning
    X_train_split, X_valid_split, y_train_split, y_valid_split = train_test_split(
        X, y, test_size=0.2, random_state=42
    )
# Train the model with sampled hyperparameters
    model = CatBoostRegressor(**params)
    model.fit(X_train_split, y_train_split)
    preds = model.predict(X_valid_split)
    rmse = mean_squared_error(y_valid_split, preds, squared=False)
    return rmse  # Return RMSE as the objective to minimize
```

Optuna

```
study = optuna.create_study(direction="minimize")
study.optimize(objective, n_trials=30)
```

optimize CatBoost

- **Best RMSE after tuning: 0.0727**
- **R2 score: 0.9948**

9. Final Evaluation and Kaggle Submission

The final tuned CatBoost model was tested on Kaggle's test dataset after aligning features identically to training data. The resulting submission.csv scored 0.00487 RMSE on the public leaderboard. This confirms the robustness and generalization power of the approach. The model was not only evaluated numerically.

10. Conclusion

Through structured data preprocessing, targeted feature engineering, and advanced hyperparameter optimization, we transformed a noisy and complex high-dimensional dataset into a robust forecasting framework. Initial results from baseline models such as linear regression and tree-based methods revealed poor performance, indicating that raw features alone lacked the predictive signal needed for accurate forecasting. To address this, we implemented domain-informed feature transformations—including lag variables, rolling statistics, and volume-based ratios—and used statistical techniques (e.g., SelectKBest) to isolate the most informative signals. Finally, we leveraged Optuna to fine-tune CatBoost hyperparameters. This systematic approach substantially improved model performance, allowing the final tuned CatBoost model to outperform all baselines and achieve a highly competitive score in a real-world benchmark scenario.

References

1. **Optuna paper**:

Akiba, T., Sano, S., Yanase, T., Ohta, T. and Koyama, M. (2019) *Optuna: A next-generation hyperparameter optimization framework*. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '19)*. New York, NY: ACM. DOI

https://doi.org/10.1145/3292500.3330701

2. **ADF test paper**:

Dickey, D.A. and Fuller, W.A. (1979) 'Distribution of the estimators for autoregressive time series with a unit root', *Journal of the American Statistical Association*, 74(366), pp. 427–431.

https://doi.org/10.2307/2286348

3. **Kaggle competition**:

Kaggle (2025) *DRW – Crypto Market Prediction*. Available at: https://www.kaggle.com/competitions/drw-crypto-market-prediction/data (Accessed: 4 June 2025).

List of Table

List of Figures

I certify that the work submitted for this assignment is my own and research sources are fully acknowledged.

Student signature: Sima Niaztalkhouncheh Date: 06/04/2025