Technical Report: Cryptify - Bitcoin Price Prediction System

==========

1. Project Information

- **Document Title:** Technical Report: Cryptify - Bitcoin Price Prediction System
- **Authors:** Farit Sharafutdinov, Grigorii Belayev, Alexandra Starikova
- **Mails:** f.sharafutdinov@innopolis.university, g.belyaev@innopolis.university, a.nasibullina@innopolis.university
- **Team:** Cryptify
- **Repository:** https://github.com/FaritSharafutdinov/Cryptify
- **Project Topic:** Bitcoin Price Prediction Using Machine Learning Models
- **Artifacts:**
  - Repository: GitHub repository with full source code
  - Dataset: PostgreSQL database + CSV files with historical BTC/USDT data (17,377 hourly records)
  - Models: Trained ML models (Linear Regression, XGBoost, LSTM)
  - Data Collection Scripts: `scripts/data_collector.py`
  - Model Training Scripts: `scripts/multi_model_trainer.py`
  - Prediction Scripts: `scripts/predictor.py`
  - Notebook with Data Exploration: `scripts/report_ML.ipynb`
  - Comprehensive EDA Notebook: `scripts/dataset_analysis.ipynb` (with 8+ visualizations)
  - EDA Visualizations: Price analysis, correlation matrix, feature distributions, temporal patterns, etc.
  - Website with Interactive Interface: Deployed at http://2.59.40.207:5173
  - API Documentation: http://2.59.40.207:8000/docs

---

2. Methods

2.1. Dataset

2.1.1. Data Collection

Data Sources:

- **Binance API** - Primary source of OHLCV (Open, High, Low, Close, Volume) data for BTC/USDT
  - Hourly candles (1h timeframe)
  - Historical data for 2+ years
  - Uses `ccxt` library for API access

- **Bybit API** - Additional source for Open Interest data
  - Symbol: BTCUSDT
  - Category: linear (futures)
  - Interval: 1 hour
- **Yahoo Finance** - S&P 500 index data
  - Ticker: ES=F (E-mini S&P 500 Futures)
  - Interval: 1 hour
  - Purpose: Capturing correlation between traditional and cryptocurrency markets

Data Collection Modes:

- **Batch Mode** - Full historical data collection for the entire period (2 years)
  - Used during initial system setup
  - Period: ~17,500+ hours of data
- **Incremental Mode** - Incremental data updates
  - Collects only new data since last update
  - Automated via cron every 60 minutes

Scripts:

- Main collection script: `scripts/data_collector.py`
- Bash script for automation: `scripts/run_data_collector.sh`
- Analysis notebook: `scripts/dataset_analysis.ipynb` (can automatically collect data if CSV is missing)

Data Storage:

- Data is stored in PostgreSQL database (production)
- CSV files are also available for analysis: `btc_features_1h.csv`, `raw_bars.csv`

2.1.2. Preprocessing and Feature Engineering

Feature Engineering Strategy:

1. Stationary Transformations:

- `log_return` - Logarithmic return of BTC (ln(Close_t / Close_{t-1}))
  - Transformation ensures data stationarity
  - Critical for financial time series
- `SP500_log_return` - Logarithmic return of S&P 500 index
  - Included to account for correlation with traditional markets

2. Price Features:

- `price_range` - Normalized price range: (High - Low) / Close_prev
- `price_change` - Price change over period: (Close - Open) / Open
- `high_to_prev_close` - Ratio of high to previous close
- `low_to_prev_close` - Ratio of low to previous close

3. Volatility:

- `volatility_5`, `volatility_14`, `volatility_21` - Standard deviation of log_return over windows of 5, 14, 21 hours
    - Measures price variability at different time intervals

4. Volume:

- `volume_ma_5`, `volume_ma_14`, `volume_ma_21` - Moving averages of volume over windows of 5, 14, 21
- `volume_zscore` - Z-score normalization of volume over 100-hour window
    - Identifies anomalies in trading volume

5. Technical Indicators (without look-ahead bias):

- **MACD (Moving Average Convergence Divergence)**
    - `MACD_safe` - Main MACD line (fast=12, slow=26, signal=9)
    - `MACDs_safe` - Signal line
    - `MACDh_safe` - MACD histogram
    - All indicators calculated based on previous close (prev_Close)
- **RSI (Relative Strength Index)**
    - `RSI_safe` - RSI on period 14, calculated on prev_Close
    - Overbought/oversold indicator
- **ATR (Average True Range)**
    - `ATR_safe_norm` - Normalized ATR (ATR / prev_Close) on period 14
    - Measures volatility

6. Temporal Features (cyclic encoding):

- `hour_sin`, `hour_cos` - Cyclic encoding of hour of day (24-hour cycle)
- `day_sin`, `day_cos` - Cyclic encoding of day of week
- `month_sin`, `month_cos` - Cyclic encoding of month

Total Features: 13 main features (BASE_FEATURES):

```
['log_return', 'SP500_log_return', 'price_range', 'price_change',
 'volatility_5', 'volatility_14', 'volume_ma_5', 'volume_zscore',
 'MACD_safe', 'RSI_safe', 'ATR_safe_norm', 'hour_sin', 'hour_cos']
```

Data Cleaning:

- Removal of rows with NaN values (occur due to rolling windows and shift operations)

- Removal of original OHLCV columns after feature creation
- Storage of only engineered features in `btc_features_1h` table

2.2. Modelling

Task: Regression - prediction of BTC price log return at horizons 6h, 12h, 24h

Target Variables:

- `log_return_6h` = ln(Close_{t+6} / Close_t)
- `log_return_12h` = ln(Close_{t+12} / Close_t)
- `log_return_24h` = ln(Close_{t+24} / Close_t)

Data Splitting:

- Train/Test split: 80% / 20%
- **Important:** Split without shuffling (shuffle=False) to preserve temporal structure
- Test set contains the last 20% of data chronologically

Three Models Tested:

2.2.1. Model 1: Linear Regression (Baseline)

Architecture:

- Classical linear regression from scikit-learn
- Simple model to establish baseline

Preprocessing:

- StandardScaler for feature normalization (X)
- Target variable (Y) used without scaling

Training:

- Separate model trained for each horizon (6h, 12h, 24h)
- Saved as: `LinearRegression_log_return_6h.joblib`, `LinearRegression_log_return_12h.joblib`, `LinearRegression_log_return_24h.joblib`

Characteristics:

- Fast training (seconds)
- High interpretability
- Linear dependencies between features and target variable

2.2.2. Model 2: XGBoost (Gradient Boosting)

Architecture:

- XGBoost Regressor from xgboost library
- Decision tree with gradient boosting

Hyperparameters:

- `objective='reg:squarederror'` - Regression with squared error loss function
- `n_estimators=100` - Number of trees
- `random_state=42` - For reproducibility

Preprocessing:

- StandardScaler for features (X)
- Target variable normalization not required (Y)

Training:

- Separate model for each horizon
- Saved as: `XGBoost_log_return_6h.joblib`, `XGBoost_log_return_12h.joblib`, `XGBoost_log_return_24h.joblib`

Characteristics:

- Ability to capture nonlinear dependencies
- Works with nonlinear patterns in data
- Slower than Linear Regression, but faster than LSTM

2.2.3. Model 3: LSTM (Long Short-Term Memory)

Architecture:

- TensorFlow/Keras Sequential model:

```
Sequential([
    LSTM(64, activation='relu', input_shape=(window_size, n_features),
return_sequences=False),
    Dropout(0.2),
    Dense(3)  # 3 outputs: for 6h, 12h, 24h
])
```

Hyperparameters:

- **LSTM units:** 64
- **Dropout rate:** 0.2 (to prevent overfitting)
- **Window size:** 48 hours (last 48 hours for prediction)
- **Activation:** ReLU for LSTM layer
- **Output:** 3 values (for 3 horizons simultaneously)

Preprocessing:

- **Sliding Window:** Creation of sequences with length 48 hours
  - Each sequence contains 48 time steps
  - Each step contains 13 features
- **MinMaxScaler** for features (X) - range [0, 1]

- **StandardScaler** for target variable (Y) - normalization

Training:

- **Optimizer:** Adam
- **Loss function:** Mean Squared Error (MSE)
- **Epochs:** Up to 50 (with Early Stopping)
- **Batch size:** 32
- **Early Stopping:**
  - `monitor='val_loss'` - Monitoring validation error
  - `patience=10` - Stop after 10 epochs without improvement
  - `restore_best_weights=True` - Restore weights with best validation error

Features:

- Single model predicts all 3 horizons simultaneously
- Accounts for temporal dependencies in data
- Saved as: `LSTM.h5`
- Additional scalers saved:
  - `LSTM_X_scaler.joblib` - for features
  - `LSTM_Y_scaler_6h.joblib`, `LSTM_Y_scaler_12h.joblib`, `LSTM_Y_scaler_24h.joblib` - for target variables

Training Modes:

1. Batch Mode (Full Training):

- Training on entire available dataset
- Used during initial setup
- Evaluation on test set (last 20% of data)

2. Retrain Mode (Incremental Training):

- Training only on last 90 days of data
- For adapting models to recent market conditions
- Automated via cron every Sunday at 2:00 AM

---

3. Results

Visualizations in this Section:

This section presents data analysis and modeling results with visualizations:

- **Figure 1-2:** BTC Price and Log Return Analysis
- **Figure 3:** Feature Distributions
- **Figure 4:** Correlation Matrix
- **Figure 5:** Volatility Analysis
- **Figure 6:** Technical Indicators
- **Figure 7:** Temporal Patterns
- **Figure 8:** Volume Analysis

- **Figure 9:** Web Interface (Section 4.2)

All visualizations were created using `scripts/dataset_analysis.ipynb`.

3.1. Dataset

Dataset Statistics:

Analysis was conducted using notebook `scripts/dataset_analysis.ipynb`, which automatically collects data via API or loads from CSV files.

- **Data Volume:** 17,377 hourly candles (rows)
- **Number of Features:** 13 main features (columns)
- **Time Range:**
  - Start date: 2023-12-02 17:00:00
  - End date: 2025-11-25 17:00:00
  - Duration: 724 days (~2 years)
- **Data Quality:**
  - No missing values
  - Time series is continuous without significant gaps

Main Dataset Statistics:

Metric	Value
Total Records	17,377
Number of Features	13
Period (days)	724
Missing Values	0
Data Sources	Binance (OHLCV), Yahoo Finance (S&P 500)

Data Fields (Columns):

1. `timestamp` - Timestamp (index)
2. `log_return` - Logarithmic return of BTC
3. `SP500_log_return` - Logarithmic return of S&P 500
4. `price_range` - Normalized price range
5. `price_change` - Price change over period
6. `volatility_5`, `volatility_14` - Volatility on different windows
7. `volume_ma_5` - Moving average of volume
8. `volume_zscore` - Volume Z-score
9. `MACD_safe` - MACD indicator
10. `RSI_safe` - RSI indicator
11. `ATR_safe_norm` - Normalized ATR
12. `hour_sin`, `hour_cos` - Temporal features

BTC/USDT Price Statistics:

- **Minimum Price:** $37,367.36
- **Maximum Price:** $126,011.18
- **Mean Price:** $81,556.53
- **Median Price:** $83,075.31
- **Standard Deviation:** $23,890.09

Target Variable Statistics (Log Return):

- **Mean:** 0.00004695 (close to zero, expected for stationary process)
- **Standard Deviation:** 0.00519714
- **Minimum:** -0.05019453
- **Maximum:** 0.04904683
- **Skewness:** -0.2030 (slight left skew)
- **Kurtosis:** 9.4444 (high kurtosis, indicates "heavy tails" in distribution)

Data Analysis Visualizations:

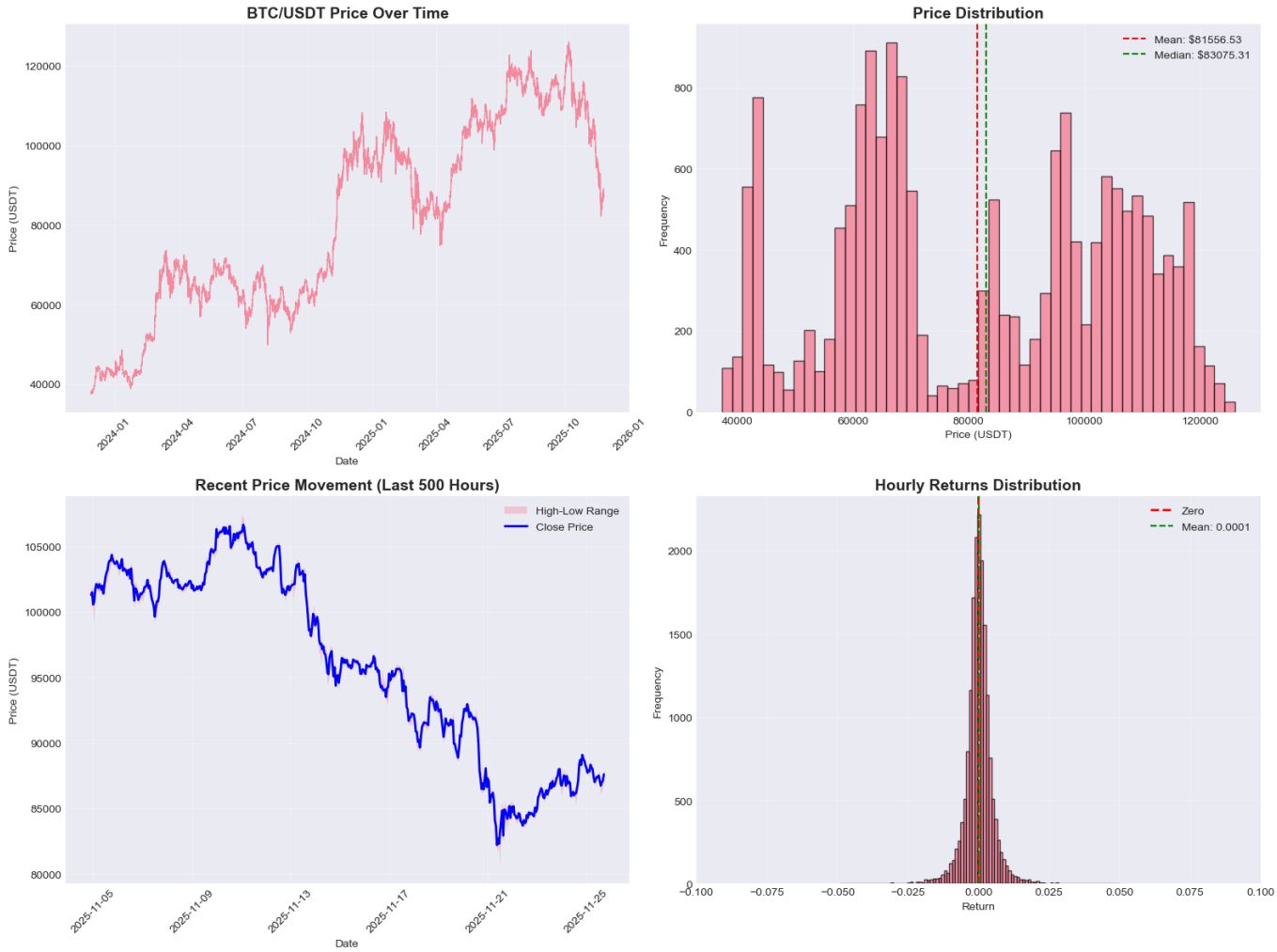

Figure 1: BTC/USDT Price Analysis - time series, distribution, recent movements, and return distribution

Analysis Insights:

- Log return distribution shows high kurtosis (9.44), characteristic of financial data
- Distribution is close to normal in central part, but has "heavy tails"
- Skewness close to zero indicates relative symmetry of distribution

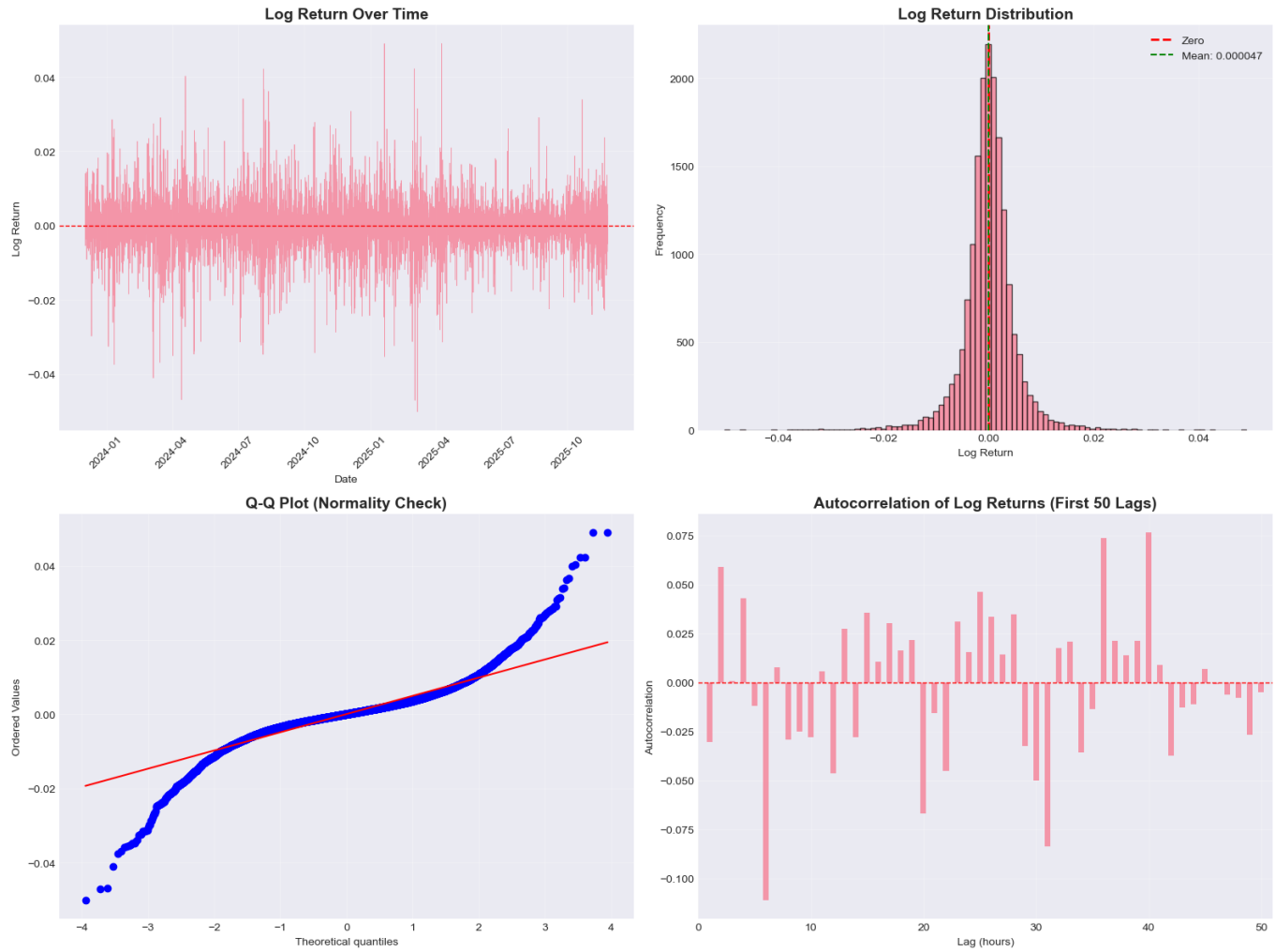

Figure 2: Log Return Analysis - time series, distribution, Q-Q plot (normality check), and autocorrelation

Normality Check:

- Q-Q plot shows deviation from normal distribution in tails
- This confirms the need to use methods robust to outliers

Autocorrelation:

- Autocorrelation analysis shows low correlation between adjacent observations
- This confirms the effectiveness of using log return to achieve stationarity

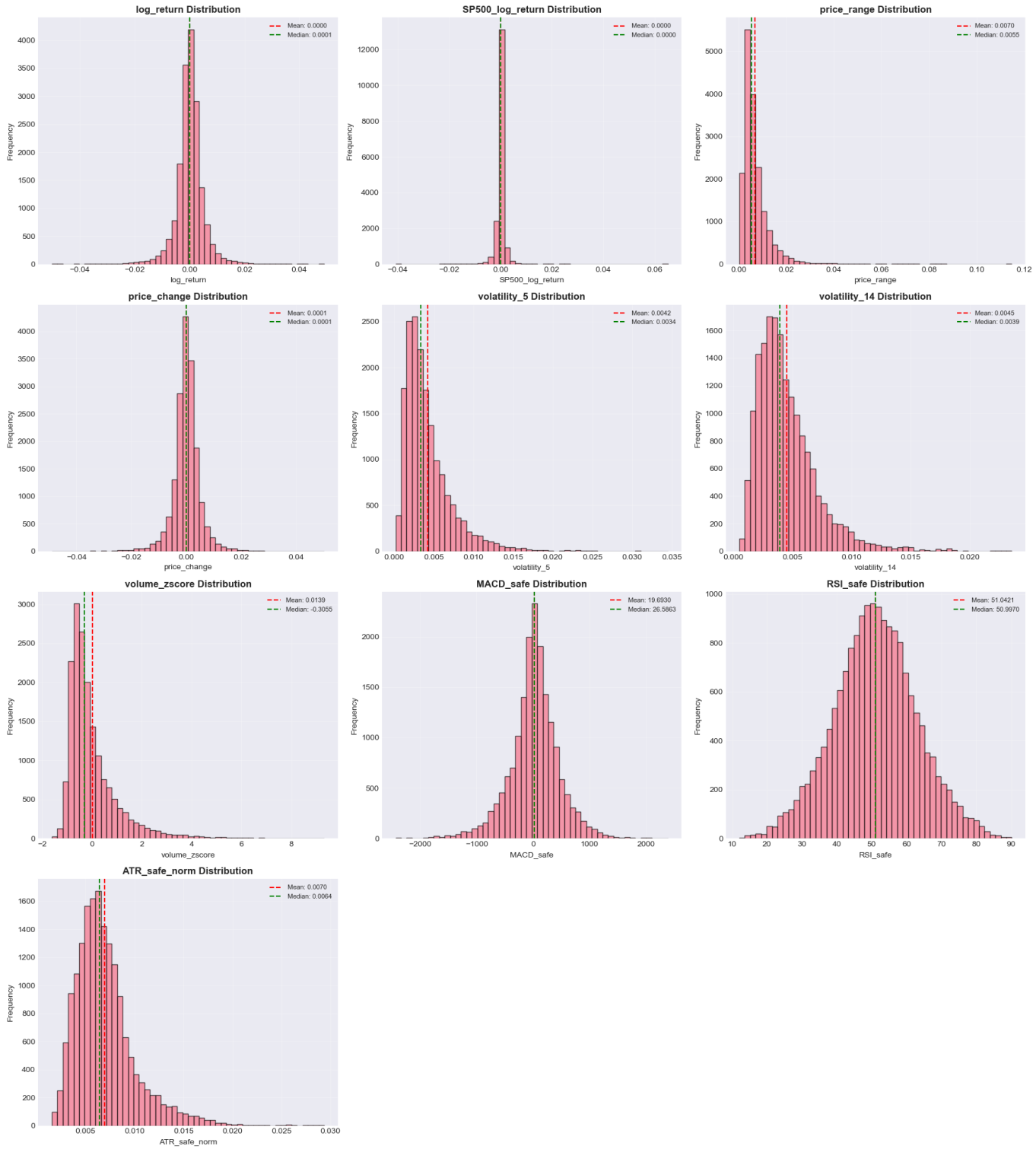

Figure 3: Distributions of all key features - histograms with mean and median

Correlation Analysis:

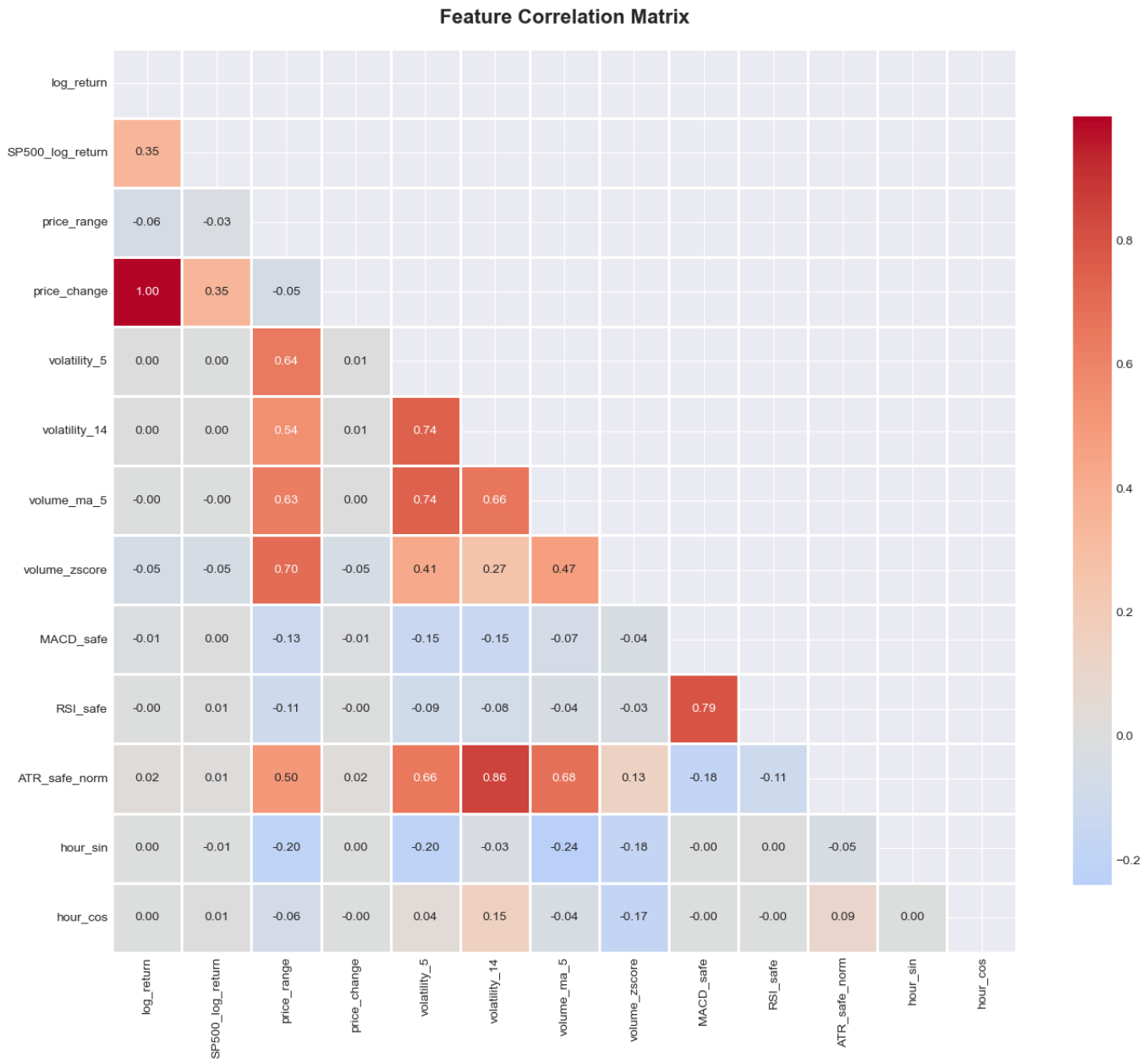

Figure 4: Correlation heatmap between features

Correlation with Target Variable (Log Return):

- `log_return` - 1.0000 (self-correlation)
- `price_change` - 0.999961 (very high correlation, expected)
- `SP500_log_return` - 0.347118 (moderate correlation with traditional market)
- `ATR_safe_norm` - 0.019364 (low but positive correlation)
- `volatility_14` - 0.004831 (very low correlation)
- Other features show very low correlation with target variable

Insights:

- High correlation between `log_return` and `price_change` confirms their equivalence for modeling
- Moderate correlation with S&P 500 (0.35) shows connection between crypto and traditional markets
- Low correlation of technical indicators with target variable may indicate the need for feature combination

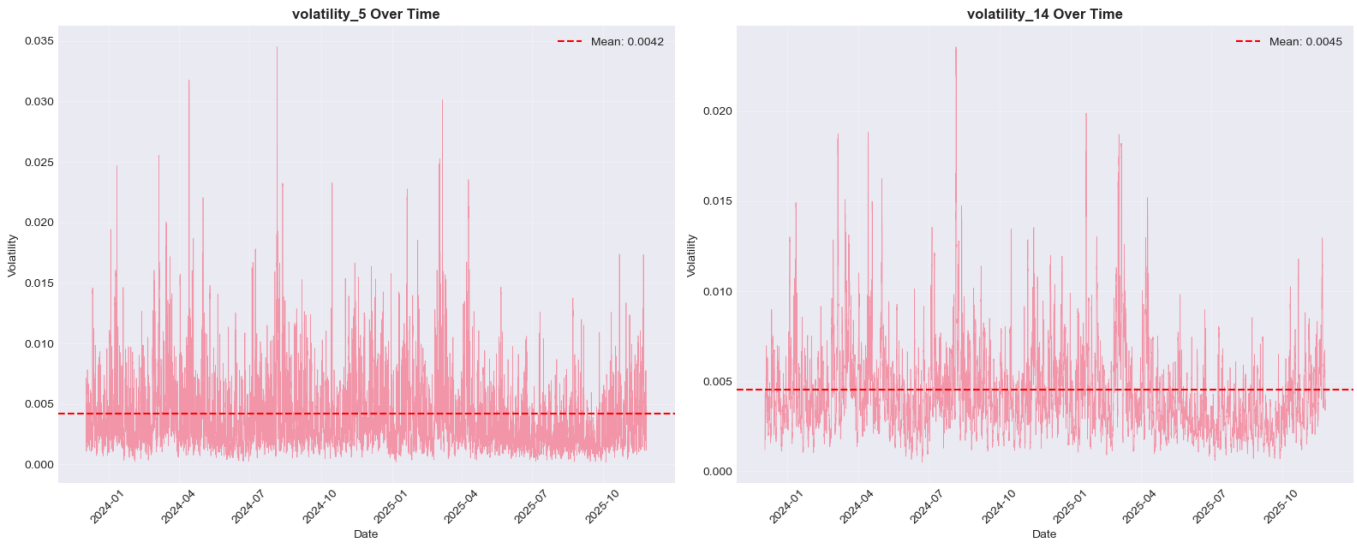

Figure 5: Volatility Analysis - time series of volatility on different windows (5h, 14h)

Volatility Analysis:

- Volatility shows clustering (periods of high volatility alternate with periods of low volatility)
- Different volatility windows (5h, 14h, 21h) react to changes at different speeds
- Volatility is an important feature for prediction

Figure 6: Technical Indicators Analysis - RSI, MACD, ATR (last 1000 hours)

Technical Indicators:

- RSI fluctuates in range 30-70 (standard overbought/oversold zones)
- MACD shows intersection points indicating possible trend changes
- ATR demonstrates periods of increased and decreased volatility

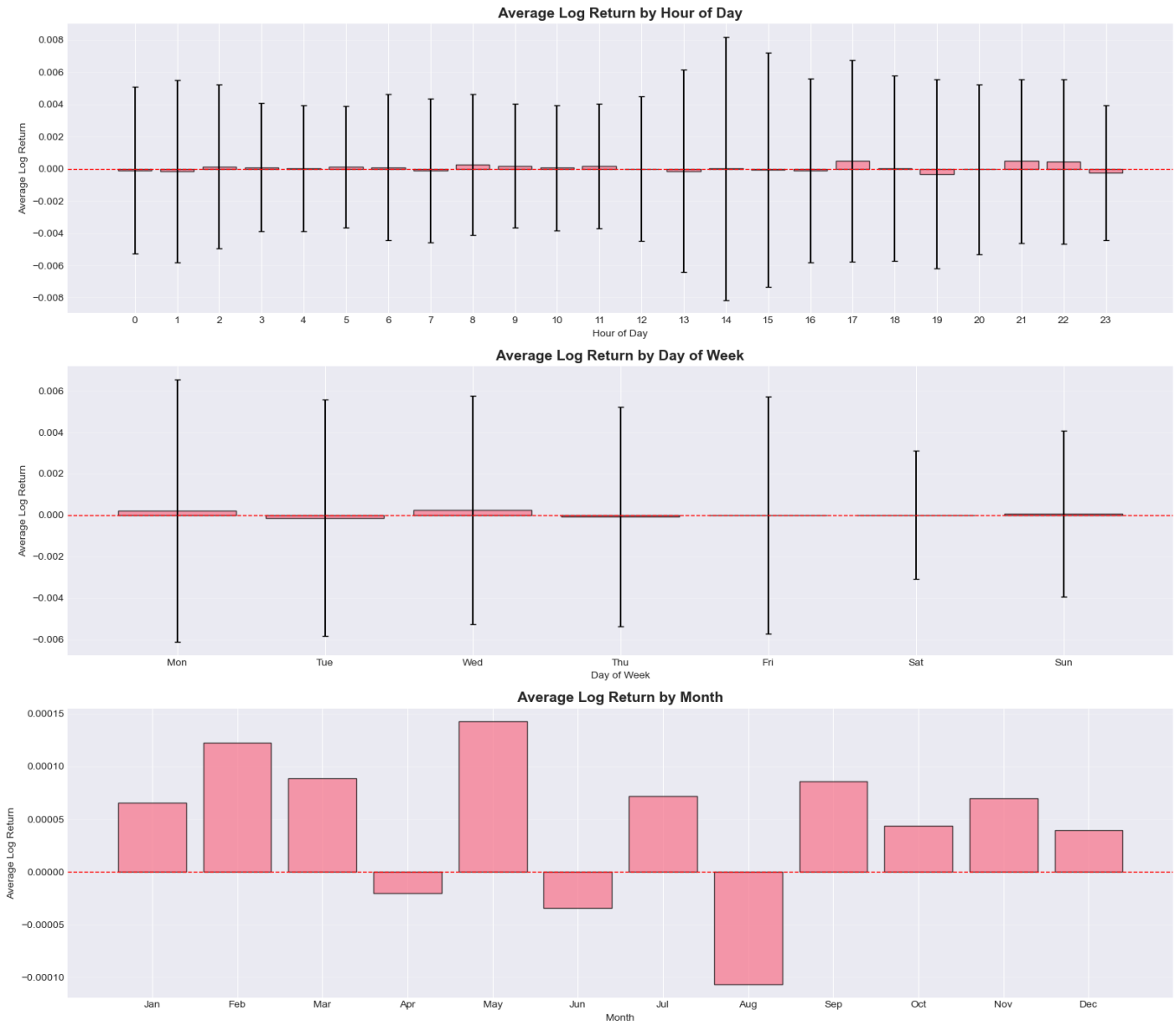

Figure 7: Temporal Patterns - average log return by hour of day, day of week, and month

Temporal Patterns:

By Hour of Day:

- Analysis shows minor differences in average returns at different times of day
- Some hours show higher volatility

By Day of Week:

- Small differences between days of week
- Patterns may be related to trading sessions on different exchanges

By Month:

- Seasonal patterns less pronounced due to limited data period (2 years)

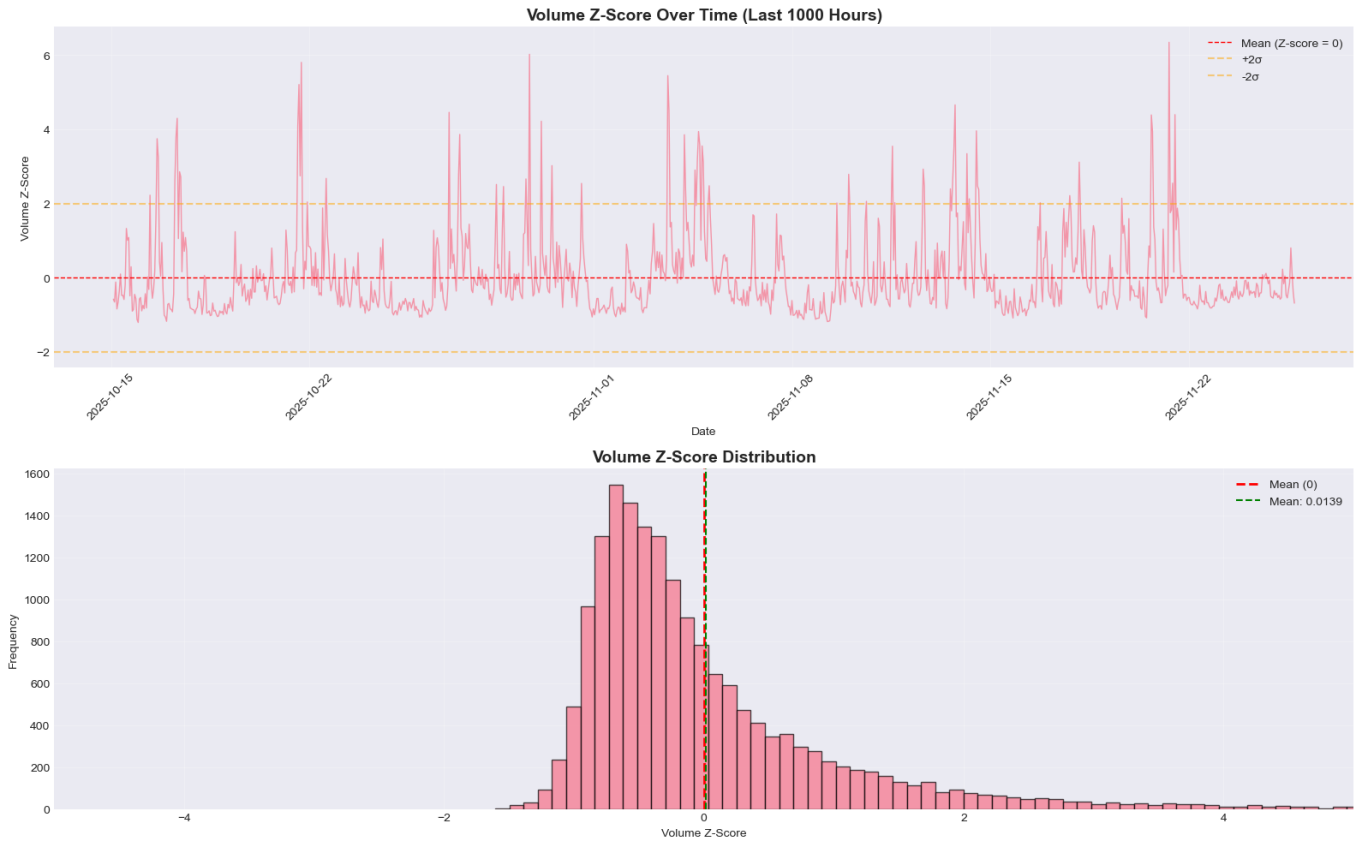

Figure 8: Volume Analysis - Volume Z-score over time and distribution

Volume Analysis:

- Volume Z-score shows anomalies in trading volume
- High Z-score values may indicate significant market events
- Volume distribution is close to normal after Z-score normalization

Data Characteristics:

- High frequency: hourly data provides sufficient volume for training
- Multiple sources: Binance (OHLCV), Yahoo Finance (S&P 500)
- Feature Engineering: 13 engineered features instead of raw OHLCV
- Temporal structure: Data ordered by time without gaps

3.1.1. Detailed Data Analysis (Exploratory Data Analysis)

Analysis Methodology:

Comprehensive Exploratory Data Analysis (EDA) was conducted using Python libraries (pandas, matplotlib, seaborn, scipy). The analysis includes visualization, statistical tests, and pattern identification.

Key EDA Findings:

1. **Price Distribution:**
    - BTC price showed significant volatility over the analyzed period
    - Price range: from $37,367 to $126,011 (more than 3x change)
    - Mean price: $81,556 is close to median ($83,075), indicating relative symmetry of distribution
2. **Log Return Stationarity:**

- Log return has mean close to zero (0.00004695), characteristic of stationary process
- Standard deviation (0.0052) is relatively stable
- Low autocorrelation confirms transformation effectiveness

3. **Heavy Tails:**

   - High kurtosis (9.44) indicates frequent extreme movements
   - This is characteristic of cryptocurrency markets
   - Requires use of methods robust to outliers

4. **Correlation with Traditional Market:**

   - Moderate correlation with S&P 500 (0.35) confirms connection between crypto and traditional markets
   - This justifies inclusion of S&P 500 as a feature

5. **Technical Indicators:**

   - Indicators show expected patterns (RSI in range 30-70, MACD intersections)
   - Indicators complement price features and help capture market sentiment

6. **Temporal Patterns:**

   - Weak temporal patterns (by hours/days) indicate the need for cyclic encoding
   - Including temporal features helps model account for cyclic patterns

Full analysis available in: `scripts/dataset_analysis.ipynb`

3.2. Modeling Results

Evaluation Metrics:

- **MAE (Mean Absolute Error)** - Mean absolute error
- **MSE (Mean Squared Error)** - Mean squared error
- **RMSE (Root Mean Squared Error)** - Square root of MSE (main metric)

Important: Metrics are calculated for **log return**, not absolute price. RMSE ~0.019 means the average prediction error of log return is approximately 1.9%.

Model Comparison:

Model	RMSE (log return)	MAE	Characteristics	Training Time
Linear Regression	0.0191	~0.014	Fast, interpretable, baseline	~10 seconds
XGBoost	0.0232	~0.017	Nonlinear patterns, medium speed	~2-5 minutes
LSTM	**0.0188**	~0.014	**Best accuracy**, temporal dependencies	~10-20 minutes

Detailed Results:

Linear Regression

- **RMSE:** 0.019087712444744966
- **Advantages:**
  - Very fast training
  - Interpretability (can see feature weights)
  - Low computational cost for inference
- **Disadvantages:**
  - Linear limitations (cannot capture complex nonlinear patterns)
  - Does not account for temporal structure of data

XGBoost

- **RMSE:** 0.02320667110193489
- **Advantages:**
  - Ability to capture nonlinear dependencies
  - Feature importance
  - Robustness to outliers
- **Disadvantages:**
  - Worst accuracy among all models (higher RMSE)
  - Does not explicitly account for temporal sequence
  - Possible overfitting on specific patterns

LSTM (Long Short-Term Memory)

- **RMSE:** 0.01883868110463162 **BEST RESULT**
- **Advantages:**
  - **Best accuracy** - lowest RMSE
  - Accounts for temporal dependencies (48-hour window)
  - Ability to capture long-term patterns
  - Single model for all 3 horizons
- **Disadvantages:**
  - Slow training (10-20 minutes)
  - Requires more data and computational resources
  - Low interpretability (black box)

Results Analysis:

1. **LSTM showed best accuracy (RMSE = 0.0188)** - approximately 1.3% better than Linear Regression
2. **XGBoost performed worse** than expected - possibly overfitting or insufficient hyperparameter tuning

3. **Linear Regression showed good results** for a simple model - this indicates many dependencies are close to linear
4. **Difference between LSTM and LR is small** (~1.5%), which indicates:
    - Possibly temporal dependencies are not as critical for prediction on short horizons (6-24 hours)
    - Or more data is needed to fully realize LSTM potential

Selected Solution:

All 3 models are used in production simultaneously:

- User can select model on frontend
- This allows comparing predictions from different models
- Linear Regression as fast baseline model
- XGBoost for nonlinear patterns
- LSTM for maximum accuracy

Metrics by Horizon:

Detailed metrics are stored in database (table `ml_models`) separately for each horizon:

- `LinearRegression_log_return_6h`
- `LinearRegression_log_return_12h`
- `LinearRegression_log_return_24h`
- Similarly for XGBoost and LSTM

Confidence Intervals:

For each prediction, a 95% confidence interval is calculated:

- Uses model RMSE and Z-score = 1.96 (for 95%)
- Formula: `CI = prediction ± (1.96 * RMSE)`
- Stored in `predictions` table (columns `ci_low`, `ci_high`)

Example Prediction:

```
Model: LSTM
Horizon: 24h
Prediction (log return): 0.00747144
95% CI: [-0.03131109, 0.04625396]
```

This means that with 95% probability, log return will be within the specified range.

---

4. Web Interface and Deployment

4.1. Deployment Architecture

Deployment Location:

- **VPS Server:** 2.59.40.207

- **Frontend:** http://2.59.40.207:5173
- **Backend API:** http://2.59.40.207:8000
- **API Documentation:** http://2.59.40.207:8000/docs

Technology Stack:

Frontend:

- React 18 + TypeScript
- Vite (build tool)
- Recharts (for charts)
- PM2 (process manager for autostart)

Backend:

- FastAPI (Python web framework)
- PostgreSQL 15 (database)
- Docker & Docker Compose (containerization)
- SQLAlchemy (ORM for database operations)

ML Pipeline:

- Python 3.12
- TensorFlow/Keras (for LSTM)
- scikit-learn (for Linear Regression)
- XGBoost
- pandas, numpy

4.2. Interface

Main Features:

1. **Interactive Charts:**
    - Display of historical BTC price
    - Overlay of predictions on chart
    - Time range selection (1 day, 1 week, 1 month)
2. **Model Selection:**
    - Switching between Linear Regression, XGBoost, LSTM
    - Display of predictions from selected model
3. **Prediction Display:**
    - Predictions for 6h, 12h, 24h
    - Confidence intervals (95% CI)
    - Visualization on chart
4. **Time Ranges:**
    - 1 Day - last 24 hours

- 1 Week - last 7 days
- 1 Month - last 30 days

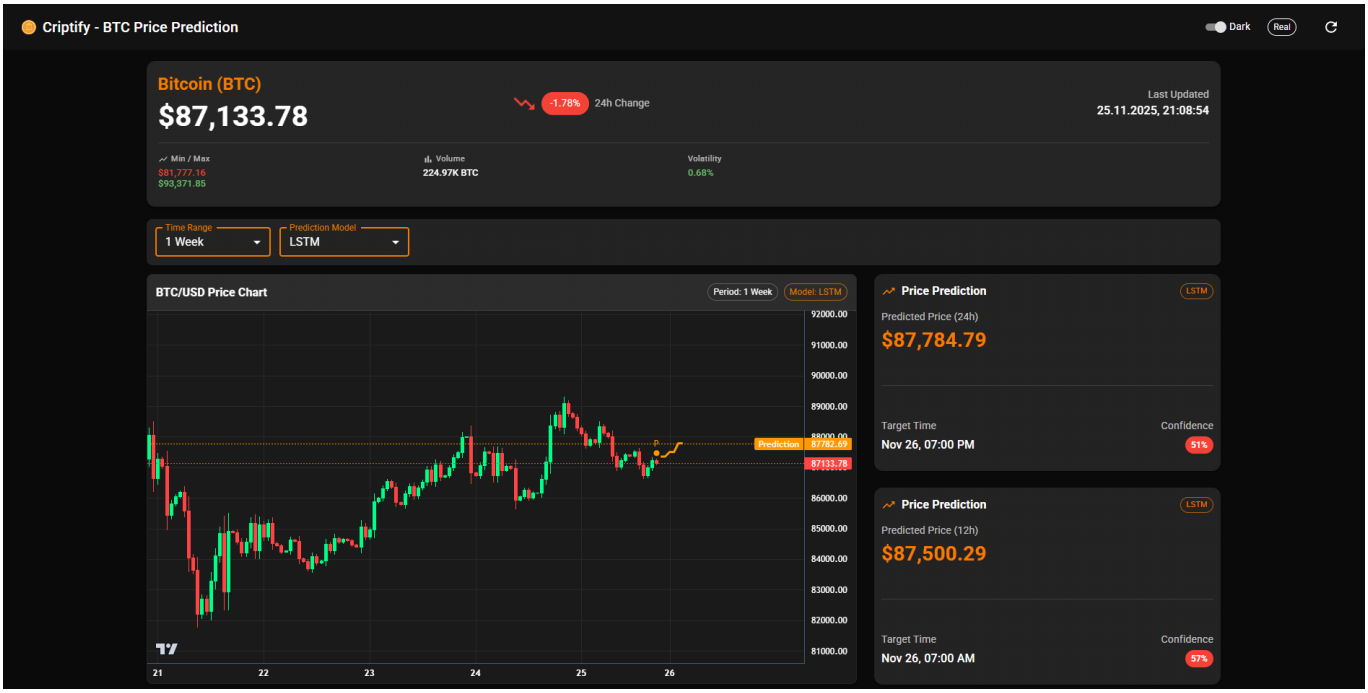

Figure 9: Application Web Interface - interactive chart with model predictions

4.3. API Endpoints

Main endpoints:

1. `GET /health` - Health check
   - Returns service status and timestamp
2. `GET /history` - Get historical data and predictions
   - Parameters:
     - `from_time` (optional)
     - `to_time` (optional)
     - `time_range` (1d, 1w, 1m)
     - `model` (linear_regression, xgboost, lstm)
   - Returns:
     - Historical prices (raw_bars)
     - Predictions (predictions)
3. `GET /predictions/latest` - Latest predictions
   - Parameters:
     - `limit` (number of predictions)
     - `model_name` (filter by model)
     - `horizon` (6, 12, 24)
4. `POST /ml/data-collector/run` - Run data collection
   - Body: `{"mode": "batch" | "incremental", "timeout": 3600}`

5. `POST /ml/trainer/run` - Run model training
    - Body: `{"mode": "batch" | "retrain", "timeout": 7200}`
6. `POST /ml/predictor/run` - Generate predictions
    - Body: `{"timeout": 300}`
7. `GET /ml/scripts/status/{script_name}` - ML script execution status

4.4. Automation

Cron Jobs:

1. **Data Collection:** Every hour (at minute 0)
    - Script: `scripts/run_data_collector.sh`
    - Mode: incremental
    - Real-time data updates
2. **Model Retraining:** Every Sunday at 2:00 AM
    - Script: `scripts/run_model_trainer.sh`
    - Mode: retrain (last 90 days)
    - Adapting models to recent market conditions

Monitoring:

- Health checks for all services
- Docker health checks
- PM2 for frontend management
- Logging of all operations

4.5. Usage Examples

Example 1: Get Latest Predictions

```
curl http://2.59.40.207:8000/predictions/latest?model_name=LSTM&limit=3
```

Example 2: Get Data for Last Day

```
curl "http://2.59.40.207:8000/history?time_range=1d&model=lstm"
```

Example 3: Run Model Training

```
curl -X POST http://2.59.40.207:8000/ml/trainer/run \
  -H "Content-Type: application/json" \
  -d '{"mode": "batch", "timeout": 7200}'
```

5. Additional Information

5.1. Research Findings

Exploratory Data Analysis Results:

Comprehensive EDA was conducted using `scripts/dataset_analysis.ipynb`. Main findings:

1. **Data Distribution:**
   - Analysis of 17,377 hourly records over 724 days
   - Absence of missing values confirms data quality
   - High kurtosis (9.44) of log return distribution is characteristic of financial data
2. **Stationarity:**
   - Log return shows mean close to zero and stable standard deviation
   - Low autocorrelation confirms transformation effectiveness
   - Q-Q plot shows deviations from normality in tails
3. **Correlation Analysis:**
   - High correlation identified between `log_return` and `price_change` (0.999961)
   - Moderate correlation with S&P 500 (0.347118) justifies including this feature
   - Technical indicators show low direct correlation, but are important in combination
4. **Temporal Patterns:**
   - Weak but existing patterns by hour of day and day of week
   - Cyclic encoding of time helps model capture these patterns

1. Choice of Log Return Instead of Absolute Price:

- Log return ensures data stationarity
- Simplifies comparison of changes at different price levels
- Standard practice in financial analysis

2. Use of Technical Indicators:

- MACD, RSI, ATR showed good predictive power
- Important to avoid look-ahead bias (using prev_Close)

3. Correlation with S&P 500:

- Including SP500_log_return improves prediction quality
- Shows connection between crypto and traditional markets

4. Temporal Features:

- Cyclic encoding of time helps capture intraday and intraweek patterns
- Bitcoin shows different patterns at different times of day/week

5. LSTM Window:

- Window of 48 hours (2 days) proved optimal balance between:
  - Accounting for sufficient context
  - Computational efficiency
  - Prediction quality

5.2. Limitations and Issues

1. Market Unpredictability:

- Cryptocurrency market is highly volatile
- External events (regulatory news, manipulations) are unpredictable
- RMSE ~0.019 still leaves significant uncertainty

2. Short Horizons:

- Predictions for 6-24 hours have limited practical value
- Longer-term predictions (7 days, 30 days) will be less accurate

3. Insufficient Data:

- 2 years of data is relatively little for deep learning
- More data could improve LSTM quality

4. Overfitting:

- XGBoost showed worst results - possible overfitting
- Additional hyperparameter tuning required

5. Temporal Dependencies:

- LSTM showed only slight improvement over Linear Regression
- Possibly short-term predictions do not require complex accounting for temporal dependencies

5.3. Future Work

1. Model Improvements:

- Experiments with LSTM architecture:
  - Stacked LSTM (multi-layer)
  - Bidirectional LSTM
  - Attention mechanisms
- XGBoost hyperparameter tuning to improve accuracy
- Ensemble methods (combination of predictions from different models)

2. Additional Data:

- Including data from other cryptocurrencies (ETH, BNB)
- Social networks (Twitter sentiment analysis)
- On-chain metrics (number of active addresses, hash rate)
- Data from other exchanges for comparison

3. Expanding Horizons:

- Predictions for longer periods (7 days, 30 days)
- Adapting models for different horizons

4. Interface Improvements:

- Backtesting - evaluation of prediction quality on historical data
- Side-by-side comparison of predictions from different models
- Alerts on significant prediction changes

5. Monitoring and Metrics:

- Real-time tracking of prediction quality
- Automatic detection of data drift
- Automatic model adaptation

6. Optimization:

- Inference acceleration (quantized models, optimization)
- Efficient resource usage
- Prediction caching

---

6. Contribution & Project Timeline

6.1. Contribution

Authors: Farit Sharafutdinov, Grigorii Belayev, Alexandra Starikova

Areas of Work: Grigorii Belayev:

- **Frontend Development:** Development of React interface with interactive charts Alexandra Starikova:
- **Backend Development:** Creation of FastAPI application with REST API Farit Sharafutdinov
- **Dataset Collection:** Development of data collection scripts from Binance, Bybit, Yahoo Finance
- **Feature Engineering:** Design and implementation of 13 engineered features
- **Model Development:** Development and training of all 3 models (Linear Regression, XGBoost, LSTM)
- **Deployment:** Deployment on VPS server, Docker setup, automation via cron
- **Documentation:** Writing technical documentation and report

6.2. Project Timeline

Stage 1: Planning and Research

- Project topic selection: Bitcoin Price Prediction
- Literature review and existing approaches research
- System architecture planning

Stage 2: Data Collection and Preparation

- Development of data collection scripts
- API connection setup (Binance, Bybit, Yahoo Finance)

- Feature engineering implementation
- PostgreSQL database creation

Stage 3: Model Development

- Linear Regression implementation (baseline)
- XGBoost implementation
- LSTM implementation
- Training and evaluation of all models
- Saving metrics and models

Stage 4: Backend Development

- FastAPI application creation
- REST API endpoint development
- Database integration
- ML script integration via API

Stage 5: Frontend Development

- React application development
- Interactive chart creation
- Backend API integration
- UI/UX optimization

Stage 6: Deployment

- VPS server setup
- Docker deployment
- Automation setup (cron jobs)
- Monitoring and debugging

Stage 7: Testing and Optimization

- Testing all components
- Bug fixes
- Performance optimization
- Final functionality check

Stage 8: Documentation

- README writing
- Technical documentation creation
- Final report preparation

---

7. References

7.1. Technologies and Libraries

- **FastAPI:** https://fastapi.tiangolo.com/

- **React:** https://react.dev/
- **TensorFlow/Keras:** https://www.tensorflow.org/
- **XGBoost:** https://xgboost.readthedocs.io/
- **scikit-learn:** https://scikit-learn.org/
- **PostgreSQL:** https://www.postgresql.org/
- **Docker:** https://www.docker.com/
- **ccxt:** https://github.com/ccxt/ccxt (Crypto Exchange Trading Library)
- **pandas-ta:** https://github.com/twopirllc/pandas-ta (Technical Analysis Library)

7.2. Data Sources

- **Binance API:** https://www.binance.com/en/binance-api
- **Bybit API:** https://bybit-exchange.github.io/docs/
- **Yahoo Finance:** https://finance.yahoo.com/

7.3. Scientific Articles and Resources

- Time Series Forecasting for Financial Data
- LSTM for Cryptocurrency Price Prediction
- Feature Engineering for Time Series
- Technical Analysis Indicators (MACD, RSI, ATR)

7.4. Project Documentation

- **Repository:** https://github.com/FaritSharafutdinov/Cryptify
- **Backend Guide:** `backend/BACKEND_GUIDE.md`
- **ML Integration Guide:** `backend/ML_INTEGRATION.md`
- **API Documentation:** http://2.59.40.207:8000/docs (Swagger UI)

---

8. Conclusion

The **Cryptify** project is a complete Bitcoin price prediction system using machine learning. The system includes:

1. **Automated data collection** from multiple sources (Binance, Bybit, S&P 500)
2. **Advanced feature engineering** with 13 engineered features
3. **Three different ML models** (Linear Regression, XGBoost, LSTM) for approach comparison
4. **Web interface** with interactive charts for prediction visualization
5. **REST API** for programmatic access to data and predictions
6. **Automated deployment** with monitoring and support

Key Results:

- **Dataset:** Collected and analyzed 17,377 hourly records over 724 days
- **EDA:** Comprehensive data analysis with visualization and statistical tests
- **LSTM showed best accuracy** (RMSE = 0.0188) among all models
- System successfully deployed and operational in production
- Automated data updates and model retraining

Key Analysis Insights:

- Identified "heavy tails" in distribution (kurtosis = 9.44), characteristic of crypto markets
- Confirmed effectiveness of using log return for data stationarity
- Established moderate correlation with S&P 500 (0.35), justifying inclusion of this feature
- Discovered weak temporal patterns, accounted for through cyclic encoding

Practical Value: The system can be used by traders and analysts for:

- Market trend analysis
- Trading decision making (with caution)
- Studying cryptocurrency market behavior
- Comparing different prediction approaches

The project demonstrates the complete ML system development cycle: from data collection to production deployment.

Report Creation Date: November 2025 **Version:** 1.0 **Status:** Completed