IntelliTradeAI - Data Sources and Data Location Documentation

Document Purpose: This document provides comprehensive information about all data sources used in the IntelliTradeAI system, including API endpoints, data formats, access methods, and relevant documentation links for review.

Table of Contents

1. Primary Data Sources
2. Data Storage Locations
3. API Configuration Details
4. Data Refresh Rates
5. Data Quality Considerations

---

1. Primary Data Sources

1.1 Yahoo Finance API

Purpose: Historical OHLCV (Open, High, Low, Close, Volume) data for stocks and cryptocurrencies

Official Documentation: - Main Site: https://finance.yahoo.com/ - Python Library (yfinance): https://pypi.org/project/yfinance/ - GitHub Repository: https://github.com/ranaroussi/yfinance

Data Provided: - Historical price data (up to 10+ years) - Daily, weekly, monthly intervals - Adjusted close prices (dividend/split adjusted) - Trading volume - Basic company information

Symbol Format: | Asset Type | Symbol Format | Example | |------------|---------------|---------| | US Stocks | Ticker symbol | AAPL, MSFT, GOOGL | | Cryptocurrencies | SYMBOL-USD | BTC-USD, ETH-USD | | ETFs |

Ticker symbol | SPY, QQQ, VOO | | Indices | ^SYMBOL | ^GSPC (S&P 500), ^DJI (Dow) |

Access Method:

```
import yfinance as yf
ticker = yf.Ticker("BTC-USD")
hist = ticker.history(period="1y", interval="1d")
```

Rate Limits: Unofficial API; rate limiting may apply during heavy usage

Cost: Free (no API key required)

1.2 CoinMarketCap API

Purpose: Real-time cryptocurrency market data, rankings, and metadata

Official Documentation: - API Documentation: https://coinmarketcap.com/api/documentation/v1/ - Developer Portal: https://pro.coinmarketcap.com/ - API Pricing: https://coinmarketcap.com/api/pricing/

Data Provided: - Real-time prices - Market capitalization - 24-hour trading volume - Circulating/total supply - Price change percentages (1h, 24h, 7d) - Market rank - Cryptocurrency metadata (name, symbol, logo, description)

API Endpoints Used:

Endpoint	Purpose	Documentation Link
`/v1/cryptocurrency/listings/latest`	Top cryptocurrencies by market cap	Link
`/v1/cryptocurrency/quotes/latest`	Current price quotes	Link
`/v1/cryptocurrency/info`	Metadata and descriptions	Link

Access Method:

```
import requests
headers = {'X-CMC_PRO_API_KEY': 'your-api-key'}
```

```
url = 'https://pro-api.coinmarketcap.com/v1/cryptocurrency/listings/latest'
response = requests.get(url, headers=headers, params={'limit': 100})
```

Rate Limits: | Plan | Monthly Credits | Daily Limit | |------|----------------|------------| | Basic (Free) | 10,000 | 333 calls | | Hobbyist | 40,000 | 1,333 calls | | Startup | 120,000 | 4,000 calls | | Standard | 400,000 | 13,333 calls |

Cost: Free tier available; paid plans for higher limits

API Key Location: Stored as environment secret `COINMARKETCAP_API_KEY`

1.3 Yahoo Finance RSS News Feed

Purpose: Financial news headlines for sentiment analysis

Official Feed URL: https://finance.yahoo.com/rss/

Feed Categories: - Market News: `https://finance.yahoo.com/rss/topstories` - Stock-specific: `https://finance.yahoo.com/rss/headline?s=SYMBOL`

Data Provided: - News headlines - Publication timestamps - Article summaries - Source attribution

Access Method:

```
import feedparser
feed = feedparser.parse('https://finance.yahoo.com/rss/headline?s=AAPL')
for entry in feed.entries:
    print(entry.title, entry.published)
```

Rate Limits: No official limits; respectful usage recommended

Cost: Free

---

2. Data Storage Locations

2.1 Local File Cache

Location: `cache/` directory

File Path	Content	Format
`cache/crypto_data.json`	Cached cryptocurrency OHLCV	JSON
`cache/stock_data.json`	Cached stock OHLCV	JSON
`cache/top_coins.json`	Top 100 coins metadata	JSON
`cache/news_cache.json`	Cached news articles	JSON

Cache Expiry: 24 hours for price data; 1 hour for news

2.2 Model Storage

Location: `models/cache/` directory

File Pattern	Content	Format
`models/cache/{symbol}_rf_model.joblib`	Trained Random Forest	Joblib
`models/cache/{symbol}_xgb_model.joblib`	Trained XGBoost	Joblib
`models/cache/{symbol}_scaler.joblib`	Feature scalers	Joblib

2.3 PostgreSQL Database

Connection: `DATABASE_URL` environment variable

Tables:

Table Name	Purpose	Key Fields
`trades`	Trade execution log	id, symbol, action, price, timestamp
`positions`	Current holdings	id, symbol, quantity, avg_price
`portfolio`	Portfolio performance	id, total_value, returns, date

Table Name	Purpose	Key Fields
`trade_alerts`	Price alerts	id, symbol, target_price, direction
`options_chains`	Cached options data	id, symbol, expiry, strike, type
`user_profiles`	User risk preferences	id, email, risk_tier, investment_amount
`esignature_records`	Legal consent records	id, user_id, timestamp, ip_address

Database Documentation: - PostgreSQL: https://www.postgresql.org/docs/ - Replit Database: https://docs.replit.com/hosting/databases/postgresql

---

3. API Configuration Details

3.1 Configuration File Location

File: `config/config.py`

```
# Key configuration parameters
COINMARKETCAP_API_KEY = os.environ.get('COINMARKETCAP_API_KEY')
COINMARKETCAP_BASE_URL = 'https://pro-api.coinmarketcap.com/v1'
YAHOO_FINANCE_TIMEOUT = 30  # seconds
CACHE_EXPIRY_HOURS = 24
```

3.2 Environment Variables

Variable	Purpose	Required
`COINMARKETCAP_API_KEY`	CoinMarketCap API authentication	Yes
`DATABASE_URL`	PostgreSQL connection string	Yes

---

4. Data Refresh Rates

Data Type	Source	Refresh Frequency
Historical OHLCV	Yahoo Finance	Daily (after market close)
Real-time Crypto Prices	CoinMarketCap	Every 5 minutes
Crypto Rankings	CoinMarketCap	Hourly
News Headlines	Yahoo RSS	Every 15 minutes
Options Chains	Yahoo Finance	On-demand
Technical Indicators	Calculated	With each price update

---

5. Data Quality Considerations

5.1 Known Limitations

Source	Limitation	Mitigation
Yahoo Finance	Occasional missing data points	Forward-fill interpolation
Yahoo Finance	Delayed quotes (15-20 min)	Use for historical analysis only
CoinMarketCap	Rate limiting on free tier	Implement caching layer
CoinMarketCap	API may change without notice	Version lock API calls

5.2 Data Validation Rules

1. **Price Validation:**
2. High must be >= Low
3. Close must be within High-Low range

4. Prices must be > 0

5. **Volume Validation:**
6. Volume must be >= 0
7. Zero volume days flagged for review
8. **Timestamp Validation:**
9. No future dates
10. No duplicate timestamps
11. Chronological ordering enforced

5.3 Outlier Detection

- Z-score filtering: Remove points > 4 standard deviations
- Price jump detection: Flag moves > 50% in single day
- Volume spike detection: Flag > 10x average volume

---

6. External Documentation Links

Official API Documentation

Resource	URL
Yahoo Finance (yfinance)	https://github.com/ranaroussi/yfinance
CoinMarketCap API	https://coinmarketcap.com/api/documentation/v1/
CoinMarketCap API Pricing	https://coinmarketcap.com/api/pricing/
PostgreSQL Documentation	https://www.postgresql.org/docs/

Python Library Documentation

Library	URL	Purpose
yfinance	https://pypi.org/project/yfinance/	Yahoo Finance data fetching

Library	URL	Purpose
pandas	https://pandas.pydata.org/docs/	Data manipulation
scikit-learn	https://scikit-learn.org/stable/documentation.html	ML models
XGBoost	https://xgboost.readthedocs.io/	Gradient boosting
SHAP	https://shap.readthedocs.io/	Model explainability
Streamlit	https://docs.streamlit.io/	Dashboard UI

Research and Methodology References

Topic	Resource
Technical Analysis Indicators	https://www.investopedia.com/terms/t/technicalindicator.asp
RSI Calculation	https://www.investopedia.com/terms/r/rsi.asp
MACD Calculation	https://www.investopedia.com/terms/m/macd.asp
Bollinger Bands	https://www.investopedia.com/terms/b/bollingerbands.asp
Random Forest	https://scikit-learn.org/stable/modules/ensemble.html#forest
XGBoost Paper	https://arxiv.org/abs/1603.02754

---

7. Data Access Code Examples

7.1 Fetching Cryptocurrency Data

```
from data.data_ingestion import DataIngestion

ingestion = DataIngestion()
```

```
crypto_data = ingestion.fetch_crypto_data(
    symbols=['BTC', 'ETH', 'SOL'],
    period='1y',
    interval='1d'
)
```

7.2 Fetching Stock Data

```
stock_data = ingestion.fetch_stock_data(
    symbols=['AAPL', 'MSFT', 'GOOGL'],
    period='2y',
    interval='1d'
)
```

7.3 Fetching Mixed Data

```
mixed_data = ingestion.fetch_mixed_data(
    crypto_symbols=['BTC', 'ETH'],
    stock_symbols=['AAPL', 'TSLA'],
    period='1y',
    interval='1d'
)
```

Document Version: 1.0
Last Updated: December 2024
Maintainer: IntelliTradeAI Development Team