RiskMetric Scoring: Comprehensive Guide

Overview

RiskMetric is a sophisticated scoring system designed to evaluate trading opportunities for cryptocurrency pairs. This guide consolidates all knowledge about the RiskMetric methodology, calculation process, and implementation details.

Core Concepts

What is RiskMetric?

RiskMetric is a risk assessment value ranging from 0 to 1, where: - Values close to 0 indicate excellent buying opportunities - Values close to 1 indicate excellent selling opportunities

The power of RiskMetric comes from combining this base risk assessment with historical context to identify rare trading opportunities that have historically yielded strong results.

Key Components

1. **Current Risk Value**: A number between 0-1 derived from the current market price
2. **Historical Time Spent**: Analysis of how much time a symbol has historically spent in different risk bands
3. **Base Score**: Points assigned based on the risk value range
4. **Coefficient**: A multiplier that gives more weight to rare occurrences
5. **Total Score**: The final actionable score (Base Score × Coefficient)

Detailed Methodology

Step 1: Obtain Current Market Price

Market prices should be obtained from reliable sources in this priority order: 1. Cryptometer API (preferred) 2. TradingView 3. CoinGecko

Example API call to Cryptometer:

```
import requests

api_key = "k77U187e08zGf4I3SLz3sYzTEyM2KNoJ9i1N4xg2"
url = f"https://api.cryptometer.io/price?symbol=ETH&api_key={api_key}"
response = requests.get(url)
price_data = response.json()
current_price = price_data["price"]
```

Step 2: Determine Risk Value

The risk value (0-1) for each symbol is determined by looking up the current market price in the RiskMetric Google Sheet: - Source: https://docs.google.com/spreadsheets/d/1Z9h8bBP13cdcgkcwq32N5Pcx4wiue9iH69uJ0wm9MRY/

This sheet contains price-to-risk mappings for each cryptocurrency. Find the row where the current price falls between the price ranges to determine the risk value.

Example lookup:

```
def get_risk_value(symbol, price, risk_data):
    symbol_data = risk_data[risk_data['Symbol'] == symbol]
    for i, row in symbol_data.iterrows():
        if row['Lower_Price'] <= price <= row['Upper_Price']:
            return row['Risk_Value']
    return None  # Price out of range
```

Step 3: Identify Risk Band

The risk value falls into one of these bands: - 0-0.1 - 0.1-0.2 - 0.2-0.3 - 0.3-0.4 - 0.4-0.5 - 0.5-0.6 - 0.6-0.7 - 0.7-0.8 - 0.8-0.9 - 0.9-1

Example function to determine risk band:

```
def get_risk_band(risk_value):
    bands = ['0-0.1', '0.1-0.2', '0.2-0.3', '0.3-0.4', '0.4-0.5',
             '0.5-0.6', '0.6-0.7', '0.7-0.8', '0.8-0.9', '0.9-1']

    for band in bands:
        lower, upper = map(float, band.split('-'))
        if lower <= risk_value < upper or (upper == 1.0 and risk_value == 1.0):
            return band

    return None  # Invalid risk value
```

Step 4: Assign Base Score

Base scores are assigned according to these risk ranges: - 0-0.25: 5 points (Excellent buy) - 0.25-0.35: 3 points (Good buy) - 0.35-0.45: 2 points (Buy) - 0.45-0.55: 1 point (Neutral) - 0.55-0.65: 2 points (Sell) - 0.65-0.75: 3 points (Good sell) - 0.75-1.0: 5 points (Strong sell)

Example function to assign base score:

```
def assign_base_score(risk_value):
    if 0 <= risk_value < 0.25:
        return 5  # Excellent buy
    elif 0.25 <= risk_value < 0.35:
        return 3  # Good buy
    elif 0.35 <= risk_value < 0.45:
        return 2  # Buy
    elif 0.45 <= risk_value < 0.55:
        return 1  # Neutral
    elif 0.55 <= risk_value < 0.65:
        return 2  # Sell
    elif 0.65 <= risk_value < 0.75:
        return 3  # Good sell
    elif 0.75 <= risk_value <= 1:
        return 5  # Strong sell
    else:
        return None  # Invalid risk value
```

Step 5: Calculate Normalized Coefficient

The coefficient is calculated based on historical time spent in risk bands: 1. Access historical data from: https://docs.google.com/spreadsheets/d/1fup2CUYxg7Tj3a2BvpoN3OcfGBoSe7EqHIxmp1RRjqg/ 2. For each symbol, identify: - Total days across all risk bands (symbol's "life") - Days spent in the current risk band - Most common band (highest days) - Rarest band with non-zero days (lowest days) 3. Calculate the normalized coefficient: - 1.0 for the most common band - 1.6 for the rarest band - Other bands scaled proportionally between 1.0-1.6

Formula:

```
def calculate_normalized_coefficient(symbol, risk_band, historical_data):
    # Get data for this symbol
    symbol_data = historical_data[historical_data['Symbol'] == symbol].iloc[0]

    # Get days in each band (excluding the 'Symbol' column)
    days_in_bands = [symbol_data[band] for band in historical_data.columns[1:] if
band != 'Total Days']
```

```
    # Filter out zero values
    non_zero_days = [d for d in days_in_bands if d > 0]

    if not non_zero_days:
        return 1.6  # All bands have zero days

    max_days = max(non_zero_days)
    min_days = min(non_zero_days)

    # Get days in current band
    days_in_current_band = symbol_data[risk_band]

    # If all non-zero values are the same
    if max_days == min_days:
        return 1.0 if days_in_current_band == max_days else 1.6

    # Calculate normalized coefficient
    if days_in_current_band == 0:
        return 1.6  # Extremely rare (no occurrence)
    elif days_in_current_band == max_days:
        return 1.0  # Most common band
    elif days_in_current_band == min_days:
        return 1.6  # Rarest non-zero band
    else:
        # Scale proportionally between 1.0 and 1.6
        return 1.0 + 0.6 * (max_days - days_in_current_band) / (max_days - min_days)
```

Step 6: Calculate Total Score

The Total Score is calculated by multiplying the Base Score by the Normalized Coefficient:

```
def calculate_total_score(base_score, coefficient):
    return base_score * coefficient
```

Step 7: Categorize Signal

Based on the Total Score and risk direction (buy/sell), signals can be categorized as:

For Buy signals (Risk < 0.5): - Very Strong Buy: Final Score ⩾ 7 - Strong Buy: 5 ⩽ Final Score < 7 - Buy: 3 ⩽ Final Score < 5 - Weak Buy: 2 ⩽ Final Score < 3 - Neutral: 0 ⩽ Final Score < 2

For Sell signals (Risk ⩾ 0.5): - Very Strong Sell: Final Score ⩾ 7 - Strong Sell: 5 ⩽ Final Score < 7 - Sell: 3 ⩽ Final Score < 5 - Weak Sell: 2 ⩽ Final Score < 3 - Neutral: 0 ⩽ Final Score < 2

Example function to categorize signal:

```
def categorize_signal(total_score, risk_value):
    is_buy_signal = risk_value < 0.5

    if is_buy_signal:
        if total_score >= 7:
            return "Very Strong Buy"
        elif 5 <= total_score < 7:
            return "Strong Buy"
        elif 3 <= total_score < 5:
            return "Buy"
        elif 2 <= total_score < 3:
            return "Weak Buy"
        else:
            return "Neutral"
    else:
        if total_score >= 7:
            return "Very Strong Sell"
        elif 5 <= total_score < 7:
            return "Strong Sell"
        elif 3 <= total_score < 5:
            return "Sell"
        elif 2 <= total_score < 3:
            return "Weak Sell"
        else:
            return "Neutral"
```

Implementation Example

Let's walk through a complete example using DOGE:

1. **Current Market Price**: $0.191151
2. **Risk Value**: 0.325 (from RiskMetric sheet)
3. **Risk Band**: 0.3-0.4
4. **Base Score**: 3 (falls in 0.25-0.35 range = Good buy)
5. **Historical Data**:
6. Days in 0.3-0.4 band: 854
7. Total days across all bands: 4,124
8. Percentage in band: 20.71%
9. Most common band: 0.2-0.3 (1,224 days)
10. Rarest band with days: 0.9-1.0 (11 days)
11. **Normalized Coefficient**: 1.18
12. Calculated as: 1.0 + 0.6 * (1224 - 854) / (1224 - 11) = 1.18
13. **Total Score**: 3 × 1.18 = 3.54

14. **Signal Category**: Buy (3 ⩽ 3.54 < 5)

Additional Examples

Example 2: ETH

1. **Current Market Price**: $2538.71
2. **Risk Value**: 0.5 (from RiskMetric sheet)
3. **Risk Band**: 0.5-0.6
4. **Base Score**: 1 (falls in 0.45-0.55 range = Neutral)
5. **Historical Data**:
6. Days in 0.5-0.6 band: 883
7. Total days across all bands: 3,525
8. Percentage in band: 25.05%
9. Most common band: 0.5-0.6 (883 days)
10. Rarest band with days: 0.9-1.0 (38 days)
11. **Normalized Coefficient**: 1.0 (most common band)
12. **Total Score**: 1 × 1.0 = 1.0
13. **Signal Category**: Neutral (0 ⩽ 1.0 < 2)

Example 3: SOL

1. **Current Market Price**: $152.56
2. **Risk Value**: 0.575 (from RiskMetric sheet)
3. **Risk Band**: 0.5-0.6
4. **Base Score**: 2 (falls in 0.55-0.65 range = Sell)
5. **Historical Data**:
6. Days in 0.5-0.6 band: 164
7. Total days across all bands: 1,815
8. Percentage in band: 9.04%
9. Most common band: 0.2-0.3 (361 days)
10. Rarest band with days: 0.9-1.0 (38 days)
11. **Normalized Coefficient**: 1.37
12. Calculated proportionally between most common and rarest
13. **Total Score**: 2 × 1.37 = 2.74
14. **Signal Category**: Weak Sell (2 ⩽ 2.74 < 3)

Complete Implementation Code

Here's a complete Python implementation that can be used to calculate RiskMetric scores:

```python
 import pandas as pd
import numpy as np
import requests

def get_current_price(symbol, api_key):
    """Get current price from Cryptometer API"""
    url = f"https://api.cryptometer.io/price?symbol={symbol}&api_key={api_key}"
    response = requests.get(url)
    if response.status_code == 200:
        price_data = response.json()
        return price_data["price"]
    return None

def get_risk_value(symbol, price, risk_data):
    """Look up risk value based on current price"""
    symbol_data = risk_data[risk_data['Symbol'] == symbol]
    for i, row in symbol_data.iterrows():
        if row['Lower_Price'] <= price <= row['Upper_Price']:
            return row['Risk_Value']
    return None

def get_risk_band(risk_value):
    """Determine which risk band the risk value falls into"""
    bands = ['0-0.1', '0.1-0.2', '0.2-0.3', '0.3-0.4', '0.4-0.5',
             '0.5-0.6', '0.6-0.7', '0.7-0.8', '0.8-0.9', '0.9-1']

    for band in bands:
        lower, upper = map(float, band.split('-'))
        if lower <= risk_value < upper or (upper == 1.0 and risk_value == 1.0):
            return band

    return None

def assign_base_score(risk_value):
    """Assign base score based on risk value"""
    if 0 <= risk_value < 0.25:
        return 5  # Excellent buy
    elif 0.25 <= risk_value < 0.35:
        return 3  # Good buy
    elif 0.35 <= risk_value < 0.45:
        return 2  # Buy
    elif 0.45 <= risk_value < 0.55:
        return 1  # Neutral
    elif 0.55 <= risk_value < 0.65:
        return 2  # Sell
    elif 0.65 <= risk_value < 0.75:
        return 3  # Good sell
    elif 0.75 <= risk_value <= 1:
        return 5  # Strong sell
    else:
        return None
```

```python
def calculate_normalized_coefficient(symbol, risk_band, historical_data):
    """Calculate normalized coefficient based on historical time spent"""
    # Get data for this symbol
    symbol_data = historical_data[historical_data['Symbol'] == symbol].iloc[0]

    # Get days in each band (excluding the 'Symbol' column)
    risk_bands = ['0-0.1', '0.1-0.2', '0.2-0.3', '0.3-0.4', '0.4-0.5',
                  '0.5-0.6', '0.6-0.7', '0.7-0.8', '0.8-0.9', '0.9-1']
    days_in_bands = [symbol_data[band] for band in risk_bands]

    # Filter out zero values
    non_zero_days = [d for d in days_in_bands if d > 0]

    if not non_zero_days:
        return 1.6  # All bands have zero days

    max_days = max(non_zero_days)
    min_days = min(non_zero_days)

    # Get days in current band
    days_in_current_band = symbol_data[risk_band]

    # If all non-zero values are the same
    if max_days == min_days:
        return 1.0 if days_in_current_band == max_days else 1.6

    # Calculate normalized coefficient
    if days_in_current_band == 0:
        return 1.6  # Extremely rare (no occurrence)
    elif days_in_current_band == max_days:
        return 1.0  # Most common band
    elif days_in_current_band == min_days:
        return 1.6  # Rarest non-zero band
    else:
        # Scale proportionally between 1.0 and 1.6
        return 1.0 + 0.6 * (max_days - days_in_current_band) / (max_days - min_days)

def categorize_signal(total_score, risk_value):
    """Categorize signal based on total score and risk direction"""
    is_buy_signal = risk_value < 0.5

    if is_buy_signal:
        if total_score >= 7:
            return "Very Strong Buy"
        elif 5 <= total_score < 7:
            return "Strong Buy"
        elif 3 <= total_score < 5:
            return "Buy"
        elif 2 <= total_score < 3:
            return "Weak Buy"
        else:
```

```
            return "Neutral"
    else:
        if total_score >= 7:
            return "Very Strong Sell"
        elif 5 <= total_score < 7:
            return "Strong Sell"
        elif 3 <= total_score < 5:
            return "Sell"
        elif 2 <= total_score < 3:
            return "Weak Sell"
        else:
            return "Neutral"

def calculate_risk_metric_score(symbol, price, risk_data, historical_data):
    """Calculate complete RiskMetric score for a symbol"""
    # Step 1-2: Get risk value
    risk_value = get_risk_value(symbol, price, risk_data)
    if risk_value is None:
        return None

    # Step 3: Identify risk band
    risk_band = get_risk_band(risk_value)
    if risk_band is None:
        return None

    # Step 4: Assign base score
    base_score = assign_base_score(risk_value)
    if base_score is None:
        return None

    # Step 5: Calculate normalized coefficient
    coefficient = calculate_normalized_coefficient(symbol, risk_band, historical_data)

    # Step 6: Calculate total score
    total_score = base_score * coefficient

    # Step 7: Categorize signal
    signal_category = categorize_signal(total_score, risk_value)

    # Return complete results
    return {
        'Symbol': symbol,
        'Current Price': price,
        'Risk Value': risk_value,
        'Risk Band': risk_band,
        'Base Score': base_score,
        'Normalized Coefficient': coefficient,
        'Total Score': total_score,
        'Signal Category': signal_category
    }

# Example usage
```

```python
def main():
    # Load data from CSV files (in production, load from Google Sheets)
    risk_data = pd.read_csv('risk_metric_data.csv')
    historical_data = pd.read_csv('historical_time_spent.csv')

    # API key for Cryptometer
    api_key = "k77U187e08zGf4I3SLz3sYzTEyM2KNoJ9i1N4xg2"

    # Symbols to analyze
    symbols = ['BTC', 'ETH', 'DOGE', 'SOL', 'LINK']

    results = []
    for symbol in symbols:
        # Get current price
        price = get_current_price(symbol, api_key)
        if price:
            # Calculate score
            score = calculate_risk_metric_score(symbol, price, risk_data, historical_data)
            if score:
                results.append(score)

    # Convert to DataFrame and display
    results_df = pd.DataFrame(results)
    print(results_df)

    # Save to CSV
    results_df.to_csv('risk_metric_scores.csv', index=False)

if __name__ == "__main__":
    main()
```

Practical Application

When analyzing a trading pair: 1. Get the current market price 2. Look up the risk value in the RiskMetric sheet 3. Determine which risk band it falls into 4. Assign the base score based on the risk value 5. Calculate the normalized coefficient for that symbol and risk band 6. Multiply to get the Total Score 7. Use the Total Score to make trading decisions

Integration with Total Trading Pair Score

The RiskMetric Total Score is one component of the overall trading pair score, which also includes: - Scores from Cryptometer API endpoints - KingFisher screenshots

The total trading pair score is calculated as:

```
Total Trading Pair Score = Cryptometer API Score + RiskMetric Score + KingFisher Score
```

Best Practices

1. **Data Freshness**: Always use the most current market prices
2. **Coefficient Calculation**: Ensure coefficients are dynamically recalculated when historical data is updated
3. **Testing**: Thoroughly test calculations for all symbols
4. **Versioning**: Track methodology versions and evaluate performance over time
5. **Error Handling**: Implement robust error handling for API failures or missing data
6. **Caching**: Consider caching historical data to improve performance
7. **Monitoring**: Regularly monitor and validate the scores against actual market performance

Tools and Resources

1. **RiskMetric Values**: https://docs.google.com/spreadsheets/d/1Z9h8bBP13cdcgkcwq32N5Pcx4wiue9iH69uJ0wm9MRY/
2. **Historical Time Spent**: https://docs.google.com/spreadsheets/d/1fup2CUYxg7Tj3a2BvpoN3OcfGBoSe7EqHIxmp1RRjqg/
3. **Cryptometer API Key**: k77U187e08zGf4I3SLz3sYzTEyM2KNoJ9i1N4xg2
4. **Cryptometer API Documentation**: Available in Google Drive

Conclusion

The RiskMetric scoring methodology provides a sophisticated approach to evaluating trading opportunities by combining current risk assessment with historical context. By giving more weight to rare occurrences, it helps identify potentially valuable trading signals that might be missed by simpler approaches.

The key innovation in this methodology is the normalized coefficient calculation, which ensures that rare market conditions are properly highlighted while common conditions receive less emphasis. This approach has been refined through multiple iterations and has proven effective in identifying high-quality trading opportunities.