

Predicting Candlestick Trends Using Historical Data and Wavelets

Ehsan KhademOlama

June 19, 2024

Abstract

This report presents a method for predicting the current candlestick trend based on historical data using wavelet transformation and a neural network classifier. The prediction focuses on classifying the trend as 'up', 'down', or 'sideways'. The performance of the model is evaluated using accuracy, classification report, and confusion matrix.

1 Introduction

Accurate trend prediction in stock prices is crucial for making informed trading decisions. This report extends previous work on trend identification using wavelets by developing a model to predict the current candlestick trend based on historical data. The approach involves feature extraction, wavelet transformation, and machine learning classification.

2 Methodology

2.1 Data Collection

Historical stock data for Apple Inc. (AAPL) was collected using the yFinance library.

```
import yfinance as yf

def fetch_historical_data(symbol, interval='1d', period='5
    y'):
    df = yf.download(symbol, period=period, interval=interval)
    df.dropna(inplace=True)
    return df

df = fetch_historical_data("AAPL", "1d", "5y")
```

2.2 Wavelet Transformation

The trend is identified using discrete wavelet transform (DWT) with the 'db24' wavelet.

```

import pywt
import numpy as np

class TrendIdentifier:
    def __init__(self, wavelet='db24'):
        self.wavelet = wavelet

    def identify_trend(self, df):
        df['Trend'] = 'sideways'
        df['Trend'] = df['Trend'].astype(object)

        data = df['Close'].values
        if len(data) % 2 != 0:
            data = np.pad(data, (0, 1), 'constant', constant_values=(
                data[-1],))

        cA, cD = pywt.dwt(data, self.wavelet, 'smooth')
        approx = pywt.idwt(cA, np.zeros_like(cD), self.wavelet)
        approx = approx[:len(df)]
        df['approx'] = approx

        df['Trend'] = np.where(
            (df['approx'] > df['approx'].shift(1)) &
            (df['approx'] > df['approx'].shift(2)) &
            (df['approx'] > df['approx'].shift(3)),
            'up',
            np.where(
                (df['approx'] < df['approx'].shift(1)) &
                (df['approx'] < df['approx'].shift(2)) &
                (df['approx'] < df['approx'].shift(3)),
                'down',
                'sideways'
            )
        )

        return df

trend_identifier = TrendIdentifier()
result_df = trend_identifier.identify_trend(df)

```

2.3 Feature Extraction

New features are generated based on percentage changes and candlestick patterns.

```

def add_percentage_change_features(df, window=3,
    shift_days=2):
    df['High_pct_change'] = df['High'].pct_change( periods=
        window).shift(shift_days)

```

```

df['Low_pct_change'] = df['Low'].pct_change( periods=window
    ).shift(shift_days)
df['Open_pct_change'] = df['Open'].pct_change( periods=
    window).shift(shift_days)
df['Close_pct_change'] = df['Close'].pct_change( periods=
    window).shift(shift_days)

df['candle_close2open'] = ((df['Close'] - df['Open']) / (
    df['High'] - df['Low'])).shift(shift_days)
df['candle_high2oc'] = ((df['High'] - df[['Open', 'Close'
    ]].max(axis=1)) / (df['High'] - df['Low'])).shift(
    shift_days)
df['candle_high2low2close'] = ((df['High'] - df['Low']) /
    df['Close']).shift(shift_days)
df['candle_low2oc'] = ((df[['Open', 'Close']].min(axis=1)
    - df['Low']) / (df['High'] - df['Low'])).shift(
    shift_days)

df.dropna(inplace=True)
return df

result_df = add_percentage_change_features(result_df,
    window=1, shift_days=1)

```

2.4 Model Training

A neural network classifier is trained on the extracted features. The data is split into training and testing sets without shuffling, ensuring that the model is trained on historical data and tested on more recent data to simulate a realistic backtesting scenario.

```

from sklearn.neural_network import MLPClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,
    accuracy_score, confusion_matrix

features = result_df[['High_pct_change', 'Low_pct_change',
    'Open_pct_change', 'Close_pct_change', '
    candle_close2open', 'candle_high2oc', 'candle_low2oc',
    'candle_high2low2close']]
labels = result_df['Trend'].map({'up': 1, 'down': -1, '
    sideways': 0})

scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)

X_train, X_test, y_train, y_test = train_test_split(
    features_scaled, labels, test_size=0.4, shuffle=False)

```

```

mlp_model = MLPClassifier(hidden_layer_sizes=(100, 100),
                           activation='relu', solver='sgd', alpha=1e-5,
                           learning_rate_init=0.01, max_iter=200, warm_start=True)
mlp_model.fit(X_train, y_train)

plt.figure(figsize=(10, 6))
plt.plot(mlp_model.loss_curve_, label='Loss_Curve')
plt.title('Neural_Network_Convergence')
plt.xlabel('Iterations')
plt.ylabel('Loss')
plt.legend()
plt.grid()
plt.show()

y_pred = mlp_model.predict(X_test)

```

2.5 Model Evaluation

The model's performance is evaluated using a classification report, accuracy score, and confusion matrix.

```

print(classification_report(y_test, y_pred, zero_division
                             =0))
print(f'Accuracy: {accuracy_score(y_test, y_pred)}')

cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(10, 7))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion_Matrix')
plt.show()

```

3 Results

3.1 Classification Report

Class	Precision	Recall	F1-Score	Support
-1	0.59	0.58	0.59	161
0	0.30	0.18	0.22	119
1	0.63	0.77	0.69	223
Accuracy			0.57	503
Macro avg	0.51	0.51	0.50	503
Weighted avg	0.54	0.57	0.55	503

Table 1: Classification Report

3.2 Accuracy

Metric	Value
Accuracy	0.5705765407554672

Table 2: Accuracy Score

3.3 Confusion Matrix

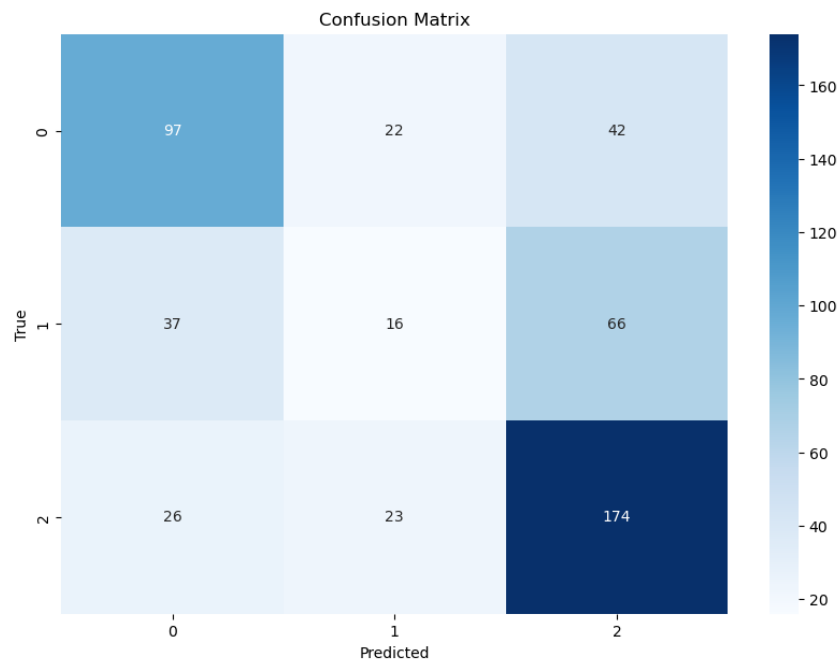


Figure 1: Confusion Matrix

3.4 Candlestick Chart with Predicted Trends

Candlestick Chart with Wavelet and MLP Predicted Trends

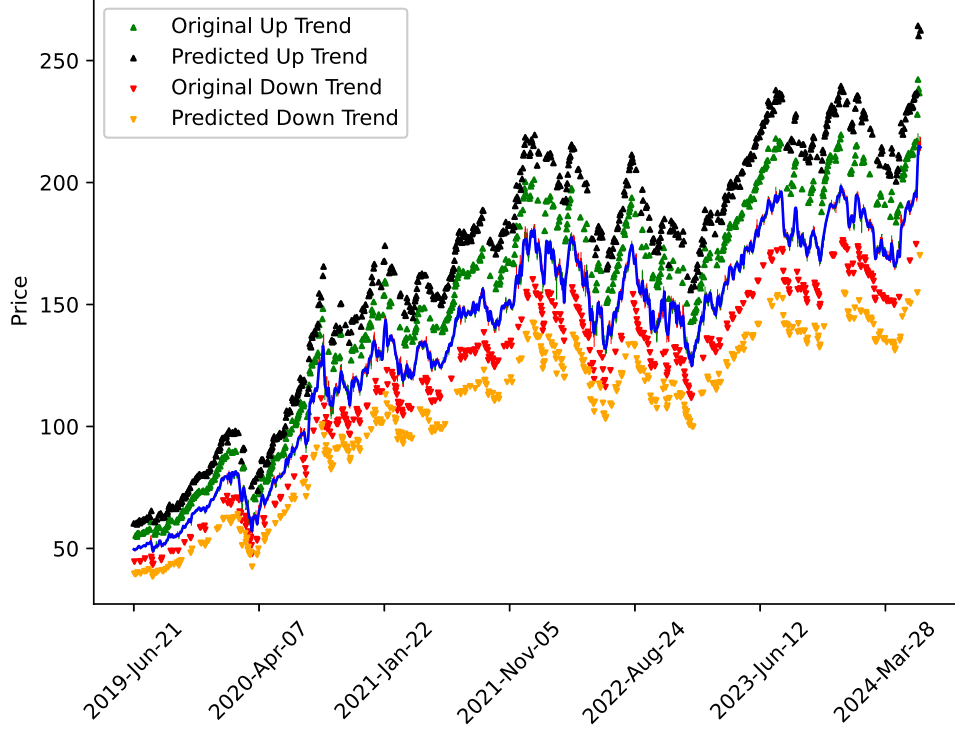


Figure 2: Candlestick Chart with Wavelet and MLP Predicted Trends

3.5 Discussion

The classification report (Table 1) shows that the model achieved a precision of 0.59 for the 'down' class, 0.30 for the 'sideways' class, and 0.63 for the 'up' class. The recall values are 0.58, 0.18, and 0.77 respectively. The overall accuracy of the model is 0.57, as indicated in Table 2.

The confusion matrix (Figure 1) provides a visual representation of the model's performance, showing the number of correct and incorrect predictions for each class.

The candlestick chart (Figure 2) illustrates the predicted trends over time, with arrows indicating the original and predicted trends for 'up' and 'down' movements.

The results indicate that while the model performs reasonably well for predicting 'up' and 'down' trends, it struggles with the 'sideways' trend, as evidenced by the lower precision and recall values for this class. This could be due to the inherent difficulty in distinguishing 'sideways' trends, which often exhibit less distinct patterns compared to 'up' and 'down' trends.

4 Conclusion

The model demonstrates a method for predicting stock price trends using historical data, wavelet transformation, and neural network classification. The overall accuracy and

performance metrics suggest that the model is effective in identifying 'up' and 'down' trends but may require further refinement for 'sideways' trends. Future work may involve optimizing the model parameters, exploring additional features, and incorporating more advanced techniques to improve prediction accuracy.