Predictive Analysis Using Regression Techniques in Technical Analysis

Adam Jolles
University of Kansas
MATH 582

I. Introduction

A. Technical analysis is a method used in financial markets to forecast the direction of prices by studying past market data, primarily price and volume. It operates on the principle that market trends can be identified through the analysis of patterns and trends in trading activity, as opposed to fundamental analysis, which focuses on company and industry-specific data.

B. Quantitative methods are crucial in market trend prediction, offering a more objective, mathematical approach. These methods use statistical and computational techniques to analyze market trends, helping to uncover patterns and insights that might be overlooked in traditional technical analysis. This data-driven approach enhances accuracy and reliability in market predictions.

C. The project aims to integrate regression techniques into technical analysis for more accurate market trend predictions. It involves collecting historical financial data, applying and evaluating linear, logistic, and polynomial regression models, and comparing their effectiveness. The scope includes data preprocessing, model development and training, performance evaluation, and a comparative analysis to determine the best approach for market trend prediction.

II. Problem Statement

A. Traditional chart pattern recognition in technical analysis is marked by significant limitations. The subjective nature of pattern interpretation often leads to inconsistent analysis, influenced by individual biases. This method relies heavily on historical data, assuming past trends will predict future movements, which may not hold true in rapidly evolving market conditions. Overemphasis on visual patterns can result in overlooking essential market dynamics, and the approach is prone to false signals due to market noise. Additionally, the lack of statistical rigor in traditional methods makes it challenging to quantify risks and validate predictions, while also failing to incorporate broader market indicators and macroeconomic factors.

B. The complexity and volatility of modern financial markets demand a shift towards quantitative, data-driven approaches. These methods, grounded in statistical and computational analysis, minimize the subjectivity and limitations of traditional chart pattern recognition. By incorporating quantitative methods into technical analysis, there's potential for more accurate, consistent, and reliable market trend predictions, which are critical for informed investment decision-making.

III. Contribution

A. This project applies linear, logistic, and polynomial regression models to analyze financial data. By leveraging these models, it seeks to capture different aspects of market behavior, ranging from simple linear trends to more complex, non-linear patterns. The goal is to utilize these models to extract meaningful insights from historical price movements and trading volumes, thus providing a more nuanced understanding of market trends.
B. The project emphasizes a quantitative approach to evaluate the performance of these regression models. Key metrics such as Mean Squared Error (MSE), accuracy, and R-squared values will be used to assess how well each model predicts market trends. This evaluation will help in determining the effectiveness of each regression technique in the context of financial market analysis, guiding the selection of the most suitable model for accurate market trend prediction.

IV. Methodology

A. Data Collection and Preprocessing

1. Source: Financial data for this project will be pulled Yahoo Finance, using the yfinance library.
2. Data types: price movements, trading volumes, etc.
3. Preprocessing steps: cleaning, normalization, etc.

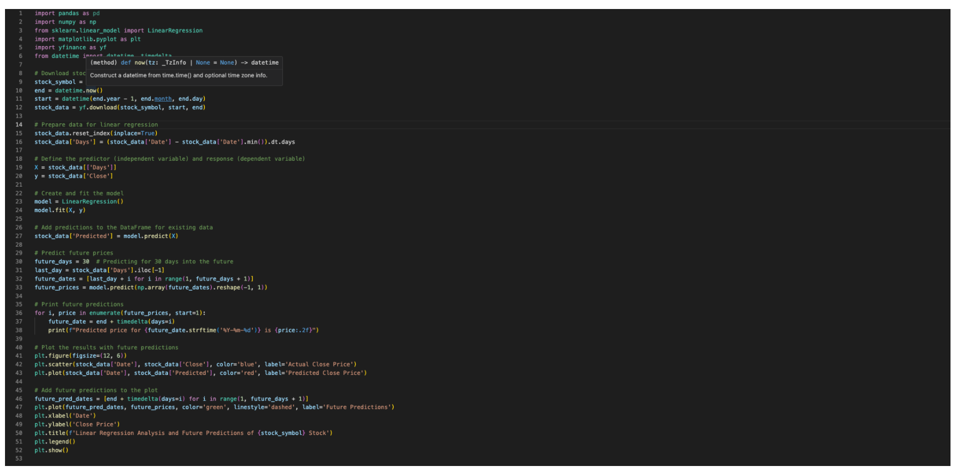
B. Model Development and Training

1. Linear Regression Model: The central principle of linear regression is to determine the best-fitting straight line through the data points that minimizes the sum of the squared differences between the observed values and the values predicted by the model.
2. Logistic Regression Model: Logistic regression is a statistical method used for binary classification. It predicts the possibility that a given data entry belongs to one of two categories based on one or more independent variables. Unlike linear regression, which predicts a continuous outcome, logistic regression estimates the probability of a binary outcome.
3. Polynomial Regression Model: Polynomial regression is a form of regression analysis in which the relationship between the independent variable x and the dependent variable y is modeled as an nth-degree polynomial. It is a special case of multiple regression that allows for the modeling of nonlinear relationships.

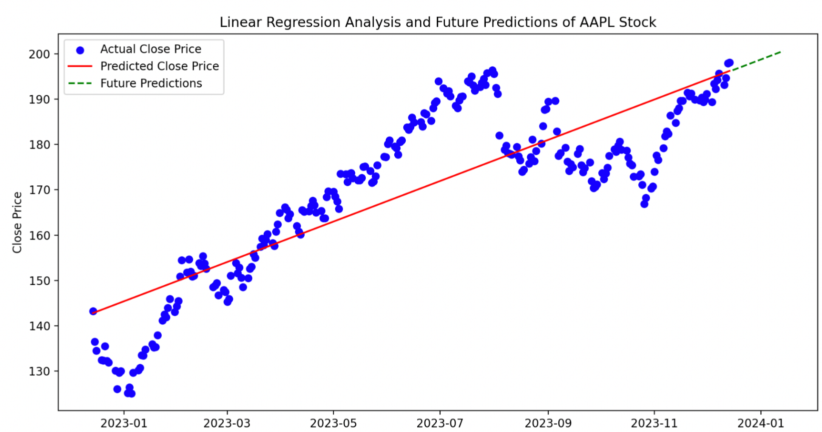
C. Linear Regression

1. While linear regression has its advantages, this form of regression assumes a linear relationship between the dependent and independent variables and may not capture the complexities of the stock market.
2. The figures below show a linear regression model of Apple's closing stock prices throughout time, using historical data from Yahoo Finance. Again, this method assumes that the future trend will follow the same pattern as the past, which is a significant simplification. Stock prices are influenced by many unpredictable factors, making accurate long-term predictions challenging.

3. The Mean Squared Error (MSE) of 115.738 on the test set for the linear regression model indicates a moderate level of error between the model's predictions and the actual stock prices. This value suggests that while the model captures some aspects of the stock's behavior, there is still a significant average discrepancy in the predictions. For instance, the model forecasts a gradual increase in the stock price over a 30-day period, with predictions ranging from $196.52 to $200.73. This steady increase reflects the model's understanding of the trend but does not account for potential market volatility or other influential factors that might cause more substantial fluctuations in the stock price. Given the nature of the stock market, characterized by frequent and sometimes unpredictable changes, an MSE of this magnitude is not unexpected. It highlights the challenges in accurately forecasting stock prices using linear regression, which assumes a consistent relationship over time. This context suggests the model, while useful for understanding general trends, may not be sufficiently robust for precise short-term trading decisions. The analysis points toward the need for incorporating additional variables or using more complex models to better capture the nuances of market movements.



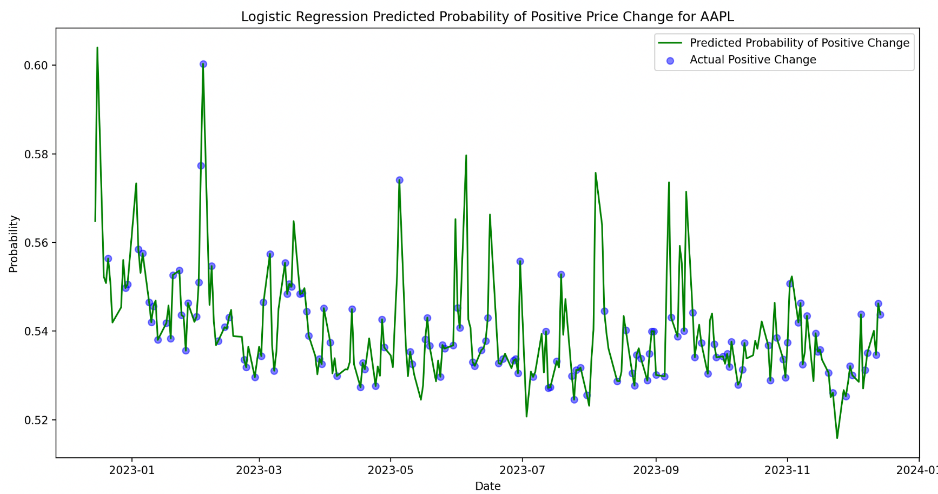
```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import yfinance as yf
from datetime import datetime, timedelta

# Download stoc
stock_symbol =
end = datetime.now()
start = datetime(end.year - 1, end.month, end.day)
stock_data = yf.download(stock_symbol, start, end)

# Prepare data for linear regression
stock_data.reset_index(inplace=True)
stock_data['Days'] = (stock_data['Date'] - stock_data['Date'].min()).dt.days

# Define the predictor (independent variable) and response (dependent variable)
X = stock_data[['Days']]
y = stock_data['Close']

# Create and fit the model
model = LinearRegression()
model.fit(X, y)

# Add predictions to the DataFrame for existing data
stock_data['Predicted'] = model.predict(X)

# Predict future prices
future_days = 30  # Predicting for 30 days into the future
last_day = stock_data['Days'].iloc[-1]
future_dates = [last_day + i for i in range(1, future_days + 1)]
future_prices = model.predict(np.array(future_dates).reshape(-1, 1))

# Print future predictions
for i, price in enumerate(future_prices, start=1):
    future_date = end + timedelta(days=i)
    print(f"Predicted price for {future_date.strftime('%Y-%m-%d')} is {price:.2f}")

# Plot the results with future predictions
plt.figure(figsize=(12, 6))
plt.scatter(stock_data['Date'], stock_data['Close'], color='blue', label='Actual Close Price')
plt.plot(stock_data['Date'], stock_data['Predicted'], color='red', label='Predicted Close Price')

# Add future predictions to the plot
future_pred_dates = [end + timedelta(days=i) for i in range(1, future_days + 1)]
plt.plot(future_pred_dates, future_prices, color='green', linestyle='dashed', label='Future Predictions')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.title(f'Linear Regression Analysis and Future Predictions of {stock_symbol} Stock')
plt.legend()
plt.show()
```

D. Logistic Regression

1. In the context of stock price data, logistic regression is suitable for binary outcomes, risk assessment, classification performance metrics, and non-linear decision boundaries.
2. The figures below show a logistic regression model of Apple's closing stock prices over time.
3. The evaluation metrics indicate that the logistic regression model, after addressing class imbalance, has an overall accuracy of 57.14% in predicting the direction of Apple's stock price change. This performance is a significant improvement from random guessing, which would yield an accuracy of 50% for a balanced dataset. The precision of the model, which indicates how many of the predicted positive changes were actually positive, is 57% for class 0 (no price increase) and 58% for class 1 (price increase). This suggests that the model is slightly better at predicting days when the stock price will increase compared to when it will not. The f1-score is balanced between both classes, at approximately 56-59%. This balance indicates that the model is relatively consistent in its performance across both classes. The confusion matrix provides a detailed breakdown of the model's predictions, showing 17 true negatives (correctly predicted no price increase), 15 true positives (correctly predicted a price increase), 11 false positives (incorrectly predicted a price increase), and 13 false negatives (incorrectly predicted no price increase). Overall, the model's performance is moderate, with nearly equal ability to predict days with and without stock price increases. This level of performance might not be sufficient for high-stakes trading decisions, and further model refinement, more sophisticated algorithms, or additional explanatory variables might be needed to improve predictability.

```python
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt
import yfinance as yf
from datetime import datetime

# Download stock data
stock_symbol = 'AAPL'  # Example with Apple stock
end = datetime.now()
start = datetime(end.year - 1, end.month, end.day)
stock_data = yf.download(stock_symbol, start, end)

# Reset index to make 'Date' a column
stock_data.reset_index(inplace=True)

# Prepare data for logistic regression
stock_data['Price Change'] = stock_data['Close'].diff()
stock_data.dropna(inplace=True)  # Remove NaN values
stock_data['Target'] = np.where(stock_data['Price Change'] > 0, 1, 0)  # 1 for positive change, 0 for negative

# Define the predictor (independent variable) and response (dependent variable)
X = stock_data[['Open', 'High', 'Low', 'Close', 'Volume']]  # Using multiple features
y = stock_data['Target']

# Create and fit the model
model = LogisticRegression()
model.fit(X, y)

# Predict classes and probabilities
stock_data['Predicted Class'] = model.predict(X)
stock_data['Predicted Prob'] = model.predict_proba(X)[:, 1]

# Visualization
# Plot the predicted probabilities
plt.figure(figsize=(14, 7))
plt.plot(stock_data['Date'], stock_data['Predicted Prob'], color='green', label='Predicted Probability of Positive Change')

# Highlight the days where the actual class is '1' (positive change)
plt.scatter(stock_data['Date'][stock_data['Target'] == 1],
            stock_data['Predicted Prob'][stock_data['Target'] == 1],
            color='blue',
            label='Actual Positive Change',
            alpha=0.5)

plt.xlabel('Date')
plt.ylabel('Probability')
plt.title(f'Logistic Regression Predicted Probability of Positive Price Change for {stock_symbol}')
plt.legend()
plt.show()
```

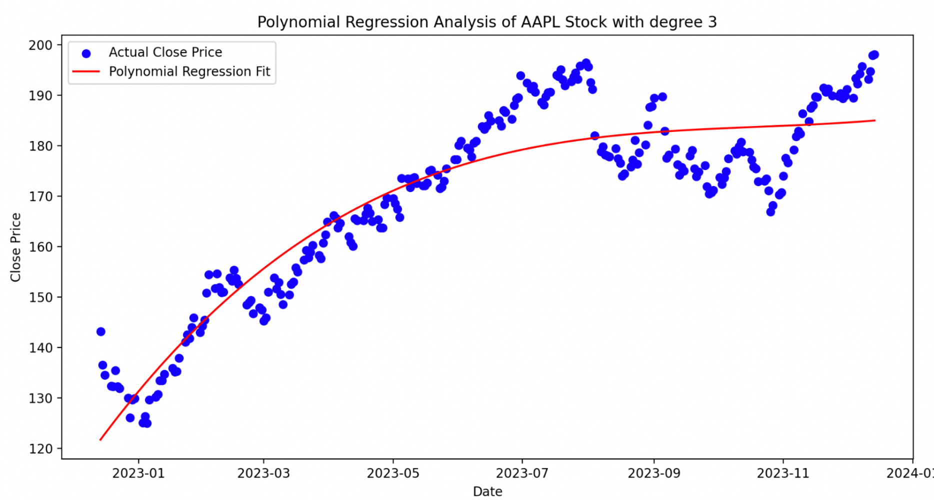


E. Polynomial Regression

1. For stock price data, polynomial regression can be suitable due to flexibility in the modeling curves, trend and pattern discovery, inflection points, and short-term forecasting.
2. The figures below show a polynomial regression model of Apple's closing stock prices over time.
3. The performance metrics of the polynomial regression model, applied to Apple's stock data, reveal insightful aspects of its predictive accuracy. With a Mean Squared Error (MSE) of 44.297 on the test set, the model demonstrates a relatively good fit, suggesting that the predicted values are, on average, approximately 44.297 units squared away from the actual stock prices. This level of MSE indicates a moderate level of error in the context of stock price predictions, where even small inaccuracies can be significant. Furthermore, the R-squared value of 0.883 indicates a strong positive

relationship between the model's predictions and the actual stock prices. An R-squared score close to 1 implies that a large portion of the variance in the stock prices is captured by the model. In this case, approximately 88.3% of the variation in Apple's stock prices can be explained by the polynomial regression model, which is quite high, signifying that the model effectively captures the trends and patterns in the data. Overall, these metrics suggest that the polynomial regression model, with its degree set to 3, provides a robust and reliable fit to the historical stock data. It successfully captures a significant amount of the variability in Apple's stock prices, making it a potentially useful tool for understanding trends and making predictions.

```python
import numpy as np
import pandas as pd
import yfinance as yf
from datetime import datetime
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures

# Download stock data
stock_symbol = 'AAPL'  # Example with Apple stock
end = datetime.now()
start = datetime(end.year - 1, end.month, end.day)
stock_data = yf.download(stock_symbol, start, end)

# Prepare data for polynomial regression
stock_data.reset_index(inplace=True)
stock_data['Days'] = (stock_data['Date'] - stock_data['Date'].min()).dt.days

# Define the predictor and response variables
X = stock_data[['Days']]
y = stock_data['Close']

# Transform the features into polynomial features
degree = 3  # Degree of the polynomial
poly_features = PolynomialFeatures(degree=degree)
X_poly = poly_features.fit_transform(X)

# Create and fit the model
model = LinearRegression()
model.fit(X_poly, y)

# Predict using polynomial regression
stock_data['Predicted'] = model.predict(X_poly)

# Plot the results
plt.figure(figsize=(12, 6))
plt.scatter(stock_data['Date'], stock_data['Close'], color='blue', label='Actual Close Price')
plt.plot(stock_data['Date'], stock_data['Predicted'], color='red', label='Polynomial Regression Fit')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.title(f'Polynomial Regression Analysis of {stock_symbol} Stock with degree {degree}')
plt.legend()
plt.show()
```



V. Comparative Analysis

A. Effectiveness of Each Model in Different Market Conditions

1. Linear regression performs well in markets with a stable trend, providing a clear directional bias which is valuable during prolonged bull or bear markets. However, its effectiveness diminishes in volatile markets with frequent trend reversals, where the assumption of linearity fails to capture the true market dynamics.

2. Logistic regression is particularly effective in range-bound or oscillating markets where the prediction of direction rather than magnitude is more relevant. It excels in classifying short-term price movements as bullish or bearish, which can benefit sideways markets where traders capitalize on minor fluctuations.
3. Polynomial regression, with its ability to model non-linear relationships, is more adaptable in conditions where market movements are not strictly linear, such as during periods of high volatility or when the market reacts to significant news events. The flexibility to fit various curves allows it to conform more closely to complex market behaviors.

B. Insights for developing robust trading strategies

1. Developing robust trading strategies requires a nuanced understanding of each model's capabilities and limitations. A suiting approach would be to utilize linear regression for establishing the broader market trend, logistic regression for identifying likely reversal points or consolidation phases, and polynomial regression for capturing the more nuanced, short-term patterns within the data.
2. Integrating these models within a comprehensive strategy involves using the predictive power of logistic and polynomial regressions for entry and exit points, while aligning with the overall trend identified by linear regression. Risk management can be enhanced by understanding the probability of price movement direction as indicated by logistic regression and by adapting to market volatility as captured by polynomial regression models.
3. Incorporating these models into a trading strategy also involves continuous evaluation against market conditions, with adjustments made as necessary. This might include recalibrating models, selecting different time frames for analysis, or combining model predictions with other technical indicators or fundamental analysis to confirm trading signals.

VI. Conclusion

A. Summary of Key Findings

1. The exploration of regression techniques to predict Apple's stock price movements yielded mixed results. The linear regression model provided a baseline for understanding the overall trend but came with a moderate Mean Squared Error (MSE) of 115.74, indicating that while the model could capture the general direction, it struggled with precision. The polynomial regression model, with an MSE of 44.30 and an R-squared of 0.88, demonstrated a better fit, capturing a significant portion of the variance in stock prices. However, this model's complexity risks overfitting. The logistic regression model achieved a 57% accuracy after addressing class imbalance, suggesting modest predictive power that requires further enhancement for practical trading applications.

B. Recommendations for Applying Regression Techniques in Technical Analysis

1. Complement regression analyses with other technical indicators and fundamental analysis to account for factors these models might not capture.
2. Be mindful of overfitting with complex models like polynomial regression. Cross-validation should be employed to validate the model's performance.
3. Utilize logistic regression for binary outcomes such as direction prediction but remain cautious about its limitations in the context of the stock market's volatility.
4. Regularly update and retrain models with new data to maintain their relevance and accuracy over time.

C. Future Research Directions
1. Incorporating more granular data, such as intraday prices or incorporating news sentiment analysis, to capture short-term market dynamics.
2. Developing a hybrid model that combines the strengths of linear, polynomial, and logistic regression to improve prediction accuracy.
3. Investigating the impact of macroeconomic indicators on stock prices to enhance the models' explanatory power.
4. Examining the applicability of these models to other financial instruments and markets to understand their versatility and limitations.

VII. References

A. Hyndman, R. J., & Athanasopoulos, G. (2018). Forecasting: Principles and Practice (2nd ed.). OTexts. Available at: https://otexts.com/fpp2/regression.html

B. TrendSpider. (n.d.). A Comprehensive Guide to Linear Regression for Traders and Investors. Available at: https://trendspider.com/learning-center/a-comprehensive-guide-to-linear-regression-for-traders-and-investors/

C. Sayah, F. (2020). Stock Market Analysis & Prediction using LSTM. Kaggle. Available at: https://www.kaggle.com/code/faressayah/stock-market-analysis-prediction-using-lstm/notebook

D. NeuroMatch Academy. (n.d.). W1D2_Tutorial1. Computational Neuroscience. Available at: https://compneuro.neuromatch.io/tutorials/W1D2_ModelFitting/student/W1D2_Tutorial1.html

E. Yahoo Finance. (n.d.). Yahoo Finance. Available at: https://finance.yahoo.com/