

# 聖ock価格予測

担当者ルト：

輸入 株価予測の課題と問題点

紹介 事業と今後の展望

著者：ヴェダント・ジェイン

# 私の紹介

名前 - VEDA NT ジエイン

大学 - IIT (ISM) ダンバード

スキル - Python、 C++、 JupyterNotebook、  
Tableau、 Numpy、 パンダ、 Matplotlib、  
シーボーン、 ドッカー およびExcel

趣味 - 好きなこと 時間を投資する

暗号市場と 分析とNFTメタ  
ウェブ3。

「Deepcraft.aiチームの皆様、

このメッセージがあなたの元に届きますように。

概説された目的に沿ってプロジェクトを完了しました。このプロジェクト

Deepcraft.aiの目標に沿うための私の献身と努力を表しています  
ビジョンと目標。

最終的な成果物は次のとおりです。

- 【研修生プロジェクトレポート】
- [Google Collab Notebook と Git-hub リンク]

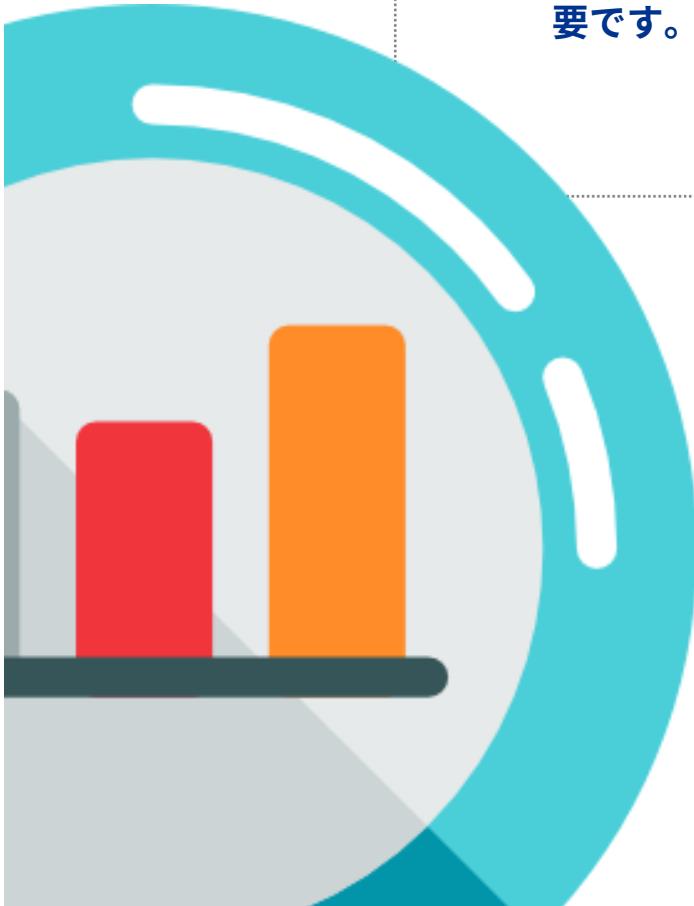
あなたのエキサイティングな活動に貢献する機会をいただき、感謝します  
皆さんのフィードバックをお待ちしています。

このプロジェクトに参加する機会を与えていただき、ありがとうございます。」

# コンテンツの概要 -:

- 1 株価に関する背景調査（課題と重要性）
- 2 データセット分析（EDAおよび前処理）
- 3 技術概要（モデル研究）
- 4 評価指標
- 5 検証結果
- 6 まとめと今後の展望

# 「なぜ D&A が必要なのか??？」



株価予測は、投資収益の最大化、リスク管理、経済政策の指針として重要です。

予測は実行できるリアルタイム市場からのデータを研究することによって。

## 投資 決定

予測精度が 1% 向上すると、数千億ドル相当のポートフォリオを管理する機関投資家にとって、数十億ドルの追加利益につながる可能性があります。



## リスク 管理

投資家はポジションをヘッジし、リスクを管理し、損失を防ぐことができます。



## 経済指標

S&P 500 やダウ・ジョーンズのような株価指数は、経済感情の指標として注目されています。



## 高頻度取引 (HFT)

- 多くの場合、ミリ秒または秒単位に基づく正確な短期価格予測により、トレーダーは急激な価格変動から利益を得ることができます。
- HFT 企業は、アルゴリズムを通じて予測された小さな株価変動を利用して、年間 80 億～100 億ドルの収益を生み出しています。



Orbes は、株式市場の変動が富裕効果を通じて米国の GDP の 5～7% に影響を与える可能性があると推定しています。

# 直面する課題:



## 市場のボラティリティと不確実性:

このような急激で突然の動きを予測することは、

市場。



## 株価の非定常性:

株価は本質的に非定常であり、その統計的特性（平均、分散など）は変化する。

これはARIMAのような従来のモデルの仮定に反しており、特定のモデルを適用することが困難になっています。  
データ変換なしの予測技術。



## 複雑さとノイズ:

株価は、収益報告、マクロ経済データ、金利など、さまざまな要因の影響を受けます。

市場感情。これらの変数を「ノイズ」（ランダムな変動）から切り離すのは複雑です。



## データの品質と可用性

データの品質は株価予測の精度に影響します。データの欠落、誤った報告、

高周波ノイズはモデル結果を歪める可能性がある。多くの場合、データは適切な分析には不十分である。



## 技術的混乱:

季節性の問題、特徴が多すぎると過剰適合につながり、モデルの能力が特定のものに制限される

データの混乱レベル

# グローバル企業におけるD&Aの影響

グローバル企業はリスク管理、市場の変動性、コンプライアンスを理解している。

アップル社

マイクロソフト株式会社

アマゾン

市場のボラティリティ: **15%** **17%** **20%**

規制リスク: **10%** **9%** **20%**

市場センチメント  
行動と  
要因: **10%** **6%** **15%**

# ?? のデータ分析

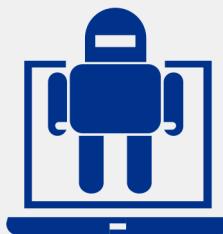
## 前処理

- 欠損データを処理します (例: 補完または削除)。重複を削除して冗長な情報を避けます。外れ値 (例: Zスコア、IQR) を検出して削除します。
- 一貫した特徴の寄与を確保するためにデータをスケーリングまたは正規化します。日付の解析と特徴の抽出 (例: 年、月、日、曜日)

## 電気通信

EDAはデータ分布を理解するのに役立ちます。

相関関係、欠損値、外れ値。



## 生のファイルの特性



価格予測では、「Close」がターゲットとして選択され、残りは特徴として残ります。

# 前処理

```
# Remove outliers using Z-score
data = data[(np.abs(stats.zscore(data['Close'])) < 3)]  
  
# Feature scaling
scaler_X = MinMaxScaler(feature_range=(0, 1))
scaler_y = MinMaxScaler(feature_range=(0, 1))
```

## 外れ値の検出と除去

外れ値は統計モデルや機械学習アルゴリズムを歪める可能性があります。Close列の外れ値はZスコア法で除去

```
# Remove '%' from '变化率 %' and convert to float
stock_data['变化率 %'] = stock_data['变化率 %'].str.replace('%', '').astype(float)  
  
# Check for any missing or invalid data after conversion
stock_data.isnull().sum(), stock_data.head()  
  
# Change columns into English Reference
stock_data.columns = ['Date', 'Close', 'Open', 'High', 'Low', 'Volume', 'Change %']
```

## 欠損データの処理

株価データのような時系列データセットでは欠損値はよく見られます。  
%Change列は浮動小数点型の値に変換され、欠落値は次のように置き換えされました。  
平均またはdropna()関数によって削除される

```
# Convert 'Date' column to datetime format
stock_data['Date'] = pd.to_datetime(stock_data['Date'])  
  
# Set 'Date' as the index for time series analysis
stock_data.set_index('Date', inplace=True)  
  
# Summary statistics of the dataset to check for any anomalies
summary_stats = stock_data.describe()
```

## 日付解析

時系列データの場合はデータ列を追加する必要があります、これはpandas lib funt()を使用して行いました。  
そしてその日付は後に索引として使われた

## データのスケーリングと正規化

正規化（最小最大スケーリング）：スケールデータを0から1の範囲に。

```
# Feature scaling
scaler_X = MinMaxScaler(feature_range=(0, 1))
scaler_y = MinMaxScaler(feature_range=(0, 1))  
  
# Scale only the feature columns (excluding 'Close')
```

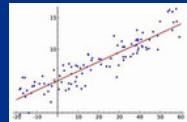
トレーニングとテストの分割

```
# Splitting the data into training and testing sets (80% train, 20% test)
train_size = int(len(scaled_data) * 0.8)
train_data = scaled_data[:train_size]
test_data = scaled_data[train_size:]
```

時系列データの場合、  
ランダムではなく時間に基づいたデータ  
80% トレーニングに使用されるデータ

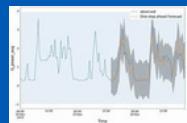
# モデル選択 -

データに最適なモデルを見つけるために合計6つのモデルが使用されました



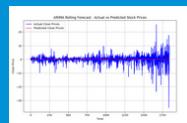
## 線形回帰

基本的なもので、線形関係を前提としており、複雑な関係には適していません。  
時間依存性。



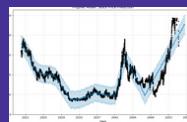
## ARIMA モデル

線形傾向、季節性、自己相関を捉え、  
ただし、单变量データに限定されます。



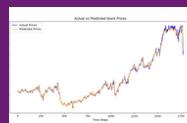
## SARIMA モデル

ARIMAを季節的な要素で拡張し、季節的な変化に適応します。  
データはありますが、依然として線形関係に限定されています。



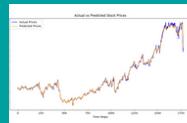
## 預言者 モデル

不規則なデータや欠損データの処理、傾向や季節性の把握には適していますが、複雑なデータの把握には適していません。  
パターンを LSTM として保存します。



## LSTM モデル

長期時系列データにおける複雑で非線形な関係を捉えるのに最も効果的で、多变量 入力をうまく処理し、  
株価のような連続データの場合。

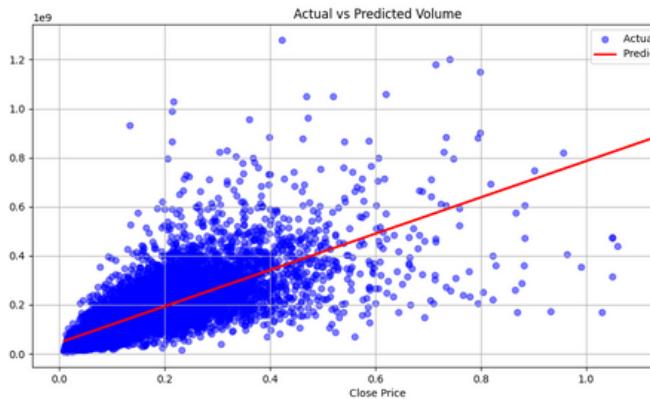


## クロノス LSTM モデル

時系列に特化し、さまざまなディープラーニングモデルをサポートし、  
多变量かつ複雑なパターンを処理するように調整されています。

# 線形回帰

欠点: 基本的な機能のみが描かれており、季節性がない  
と特徴が考慮されたため、モデルは失敗する可能性があります  
リアルタイムで予測する



注目変数（例：[ボリューム]列）  
を確認する

グラフの増加の性質は、ボリュームと終値の直接的な  
関連性を示しており、したがって  
機能として選択できます

```
# Create lag features for Closing Price
df['Lag1'] = df['Close'].shift(1)

# Define features and target Closing Price
X = df[['Lag1', 'Volume']]
Y = df[['Close']]

# Drop any rows with missing values after applying lag features
combined = pd.concat([X, Y], axis=1).dropna()

# Separate the features and target from the cleaned dataset
X = combined[['Lag1', 'Volume']]
Y = combined[['Close']]
```

ラグ機能を作成する:  
例: [閉じる] 列

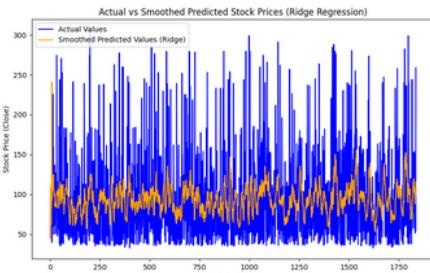
特徴エンジニアリングを使用して、1ブロック  
シフトされた終値のラグ列を開発しました。  
線形モデルの機能として追加されました



過剰適合を取り除いた後の最  
終プロット  
例: 過剰適合を避けるため、すべ  
ての特徴は含まれていない

リッジ回帰はモデルの明確さと効率をさらに高めるために実行されましたが、こ  
れは古く、

時代遅れ



# ARIMAモデル

欠点: 静止データにのみ適用可能で、季節性はあまり考慮されていない

```
# Check for stationarity using the Augmented Dickey-Fuller
adf_test = adfuller(close_prices_clean)
print(f"ADF Statistic (Cleaned Data): {adf_test[0]}")
print(f"p-value (Cleaned Data): {adf_test[1]}")

# If the time series is non-stationary (p-value > 0.05), difference it
if adf_test[1] > 0.05:
    print("Time series is non-stationary. Applying differencing")
    close_prices_clean = close_prices_clean.diff().dropna()

# Check stationarity again after differencing
adf_test = adfuller(close_prices_clean)
print(f"After Differencing - ADF Statistic: {adf_test[0]}")
print(f"After Differencing - p-value: {adf_test[1]}")
```

定常性をチェックする  
例: [閉じる] 列

ARIMAモデルは定常値モデルに使用されるため、p値が0.05より大きい場合、値は定常です。

そうでない場合は、st にするために diffrence が適用されます。

```
# Automatically select the best ARIMA parameters using auto_arima
auto_model = auto_arima(train_data, seasonal=True, m=12, stepwise=True, trace=True,
                        suppress_warnings=True, error_action="ignore", start_p=1)

# Get the best ARIMA order
order = auto_model.order
print(f'Optimal ARIMA order: {order}')

# Perform rolling forecast
history = list(train_data)
predictions = []

for t in range(len(test_data)):
    # Fit the ARIMA model to the current history
    model = ARIMA(history, order=order)
    model_fit = model.fit()

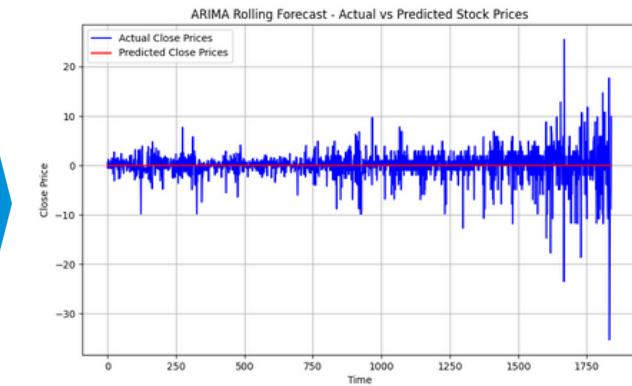
    # Forecast one step ahead
    forecast = model_fit.forecast()[0]
    predictions.append(forecast)

    # Add the actual price to the history
    history.append(test_data[t])
```

ARIMA パラメータ (p、d、q) を選択します。

例: [閉じる] 列

ラグ、一次差分、MA項 (p、q、d) は適用され、モデルはこれらに適合される  
価値観



MSEとMAEを適用した後の最終  
プロット

出力は(0,0,0)の順序で、つまり  
そのモデルはトレンドを見つけられなかつたし、  
データのパターン



SARIMA - 季節に合わせてARIMAを拡張

季節性を表す

$I_c Q^p e^t$  季節データに適しています。サリマの「S」

# SARIMAモデル

欠点：静止データにのみ適用可能であり、  
線形関係に限定される

```
# Check for stationarity using the Augmented Dickey-Fuller (ADF) test
adf_test = adfuller(close_prices_clean)
print(f"ADF Statistic (Cleaned Data): {adf_test[0]}")
print(f"p-value (Cleaned Data): {adf_test[1]}")

# If the time series is non-stationary (p-value > 0.05), difference it
if adf_test[1] > 0.05:
    print("Time series is non-stationary. Applying differencing.")
    close_prices_clean = close_prices_clean.diff().dropna() # First difference

# Check stationarity again after differencing
adf_test = adfuller(close_prices_clean)
print(f"After Differencing - ADF Statistic: {adf_test[0]}")
print(f"After Differencing - p-value: {adf_test[1]}")
```



```
# Step 2: Use Auto ARIMA to find best p, d, q, P, D, Q, m values
stepwise_fit = auto_arima(data['Close'], seasonal=True, m=12,
                           trace=True, suppress_warnings=True)

# Print the best model
print(stepwise_fit.summary())

# Step 3: Fit SARIMA model
model = SARIMAX(data['Close'], order=(1,1,1), seasonal_order=(1,1,1,12))
sarima_result = model.fit(disp=False)
```

定常性をチェックする  
例: [閉じる] 列

SARIMAモデルは定常値モデルに使用される

したがって、p 値が 0.05 より大きい場合、値は定常であり、そうでない場合は、それを定常とするために差異が適用されます。

SARIMA パラメータ (p, d, q, M) を選択します。  
例: [閉じる] 列

ラグ、一次差分、MA 項 (p, q, d) が適用される  
そして、これらの値でモデルを適合させ、季節性を表す「m」を追加することで季節性が追加されます。  
季節性サイクル (m=12 は 12か月を意味します)



SARIMA - ARIMよりも優れていることが証明された  
分析する必要があったので、顔を抱きしめる預言者モデル分析する

あ <sup>b</sup> <sub>aなた</sub> <sup>t</sup> <sub>sのつ</sub> <sup>d</sup> <sub>s</sub> 突然の変動や傾向が見られるまで  
データセットの傾向

# 預言者モデル

欠点: あまり複雑でないパターンに限定されており、より複雑なパターンや予測を扱うことができない。

## 2番目のグラフ

```
# Prepare the data for Prophet
prophet_data = data[['Date', 'Close']].rename(columns={'Date': 'ds', 'Close': 'y'})

# Initialize the Prophet model and add additional seasonality (weekly and monthly pattern)
model = Prophet(daily_seasonality=True, yearly_seasonality=True, weekly_seasonality=True)
model.add_seasonality(name='monthly', period=26, fourier_order=7) # Add monthly seasonality

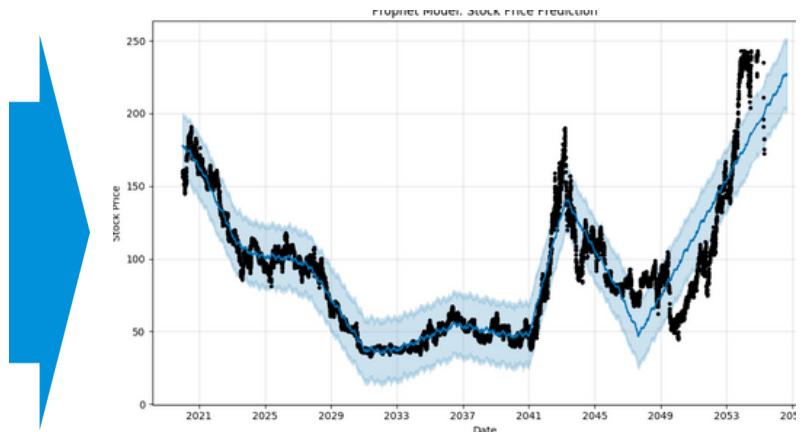
# Fit the model
model.fit(prophet_data)

# Define future data points for prediction (365 business days ahead)
future = model.make_future_dataframe(periods=365, freq='B')

# Make predictions
forecast = model.predict(future)
```

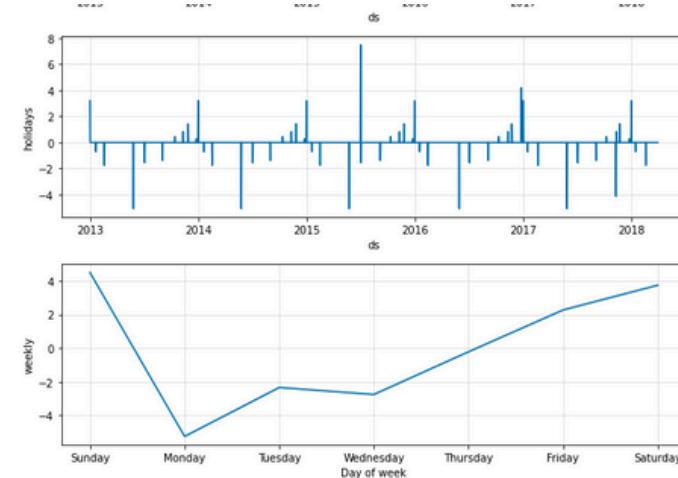
季節性と将来のデータポイントが追加されました

Prophetは、月次、週次、年次の季節性を扱うより高度なモデルです。さらに、X軸の日付列によって将来の傾向を予測します。



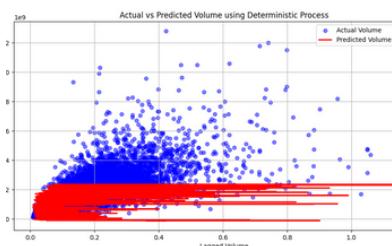
年間トレンドグラフ  
(例) [閉じる]列

将来年における誤差（グラフの終わりの方）  
データが利用できないことが原因です。修正できます  
将来のモデルやより入手可能なデータ



季節性の傾向

週ごと、月ごと、年ごとの傾向も把握でき、Prophetでは  
レスポンシブなデータ分析とチャート



次へ進む前に メートル トレンドとパット 例えば、 $y =$  の代わりに  $y = mx^2 + px^3 + q$  私 れた インプレメートル た私 れた ミニスティックモデル。チェックしました  
t erん 私ん  $x^2n$  d 1つの n d 線形ではなく 3 次方程式。  
x t x + p + y = mx^2 + px^3 + q などに移動しました

あまり成功しなかったが、他のことにも応用できるので言及する価値がある  
データセット

# クロノスモデル

欠点: 実行が複雑で、実行するにはより高度なGPUが必要  
時間がかかりすぎるので

```
# Remove outliers using Z-score method (threshold set to 3)
z_scores = np.abs(stats.zscore(df['Close']))
df = df[z_scores < 3]

# Alternatively, we can use the IQR method (interquartile range) to remove outliers
# Q1 = df['Close'].quantile(0.25)
# Q3 = df['Close'].quantile(0.75)
# IQR = Q3 - Q1
# df = df[(df['Close'] >= Q1 - 1.5 * IQR) & (df['Close'] <= Q3 + 1.5 * IQR)]
```

## 外れ値に使用されるIQR法

外れ値を除外する代わりにZスコアを使うことができます

IQR法を使用して、上限と下限を空いている場所を埋めます。

下位四分位値



```
# Ensure the 'Date' column is in the correct datetime format
df['Date'] = pd.to_datetime(df['Date'])

# Sort the data by date to ensure it is in chronological order
df = df.sort_values(by='Date')

# Select features and target for the model
features = ['High', 'Low', 'Open', 'Volume'] # Modify if needed
target = ['Close']

# Feature scaling (MinMaxScaler for Chronos compatibility)
scaler = MinMaxScaler()
```

## 利用可能なすべての機能を収集しました

このモデルは最も先進的なモデルの1つであり、  
データセットのすべての特徴を使用して集合を形成した  
予測（クロ、オープン、高、安）



クロノスモデルLSTMは、  
そして、同様の結果を達成することができます  
6つの方法すべてにおいて、すべての欠点を排除した予測方法である。

この方法は複雑で、それを実行するにはより高性能なGPUが必要である

# LSTM モデル

欠点：LSTMモデルには、いくつかのハイパーパラメータ（層の数、層あたりのユニット数、ドロップアウト率など）があり、慎重に調整する必要があります。

```
# Reshape the input to be [samples, time steps, features] as required by LSTM
# Here we assume a time step of 1 because your data is structured for each row
X_train_reshaped = np.reshape(X_train, (X_train.shape[0], 1, X_train.shape[1]))
X_test_reshaped = np.reshape(X_test, (X_test.shape[0], 1, X_test.shape[1]))

# Build the LSTM model
model = Sequential()

# Reshape the input to be [samples, time steps, features] as required by LSTM
# Here we assume a time step of 1 because your data is structured for each row as or
X_train_reshaped = np.reshape(X_train, (X_train.shape[0], 1, X_train.shape[1]))
X_test_reshaped = np.reshape(X_test, (X_test.shape[0], 1, X_test.shape[1]))

# Build the LSTM model
model = Sequential()
```

## データ列の形状を変更する

データ列は、サンプル、時間ステップ、および3DモデルのLSTMフィッティングデータに従って再形成されます。

データ形式

```
# Adding a second LSTM layer
model.add(LSTM(units=50, return_sequences=False))
model.add(Dropout(0.2))

# Adding the output layer (since we are predicting a single value, the Close price)
model.add(Dense(units=1))

# Compiling the model
model.compile(optimizer='adam', loss='mean_squared_error')

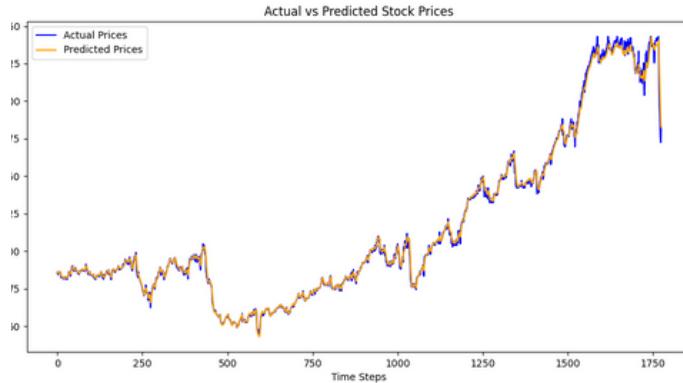
# Training the model
history = model.fit(X_train_reshaped, y_train, epochs=20, batch_size=32, validation_data=(X_test_reshaped, y_test))

# Evaluate the model
loss = model.evaluate(X_test_reshaped, y_test)
```

## ニューラルネットワークのようなLSTMのレイヤーを追加する

入力層は3Dデータで構成され、LSTM層は複雑なパターンを処理し、

最後のレイヤー - ドロップアウトレイヤー



## 最終プロット

最終プロットは、2つのレイヤーを減らして時間を短縮した後に復活しました。

効率に影響を与えるエポックも20に減少しました

これらすべてのモデルを研究した後、**最低賃金** ま メートル。d e l わs 1つ ふ かけ e メートル S t s あなたが成功し、複雑さが少ないSに採用できるモデル

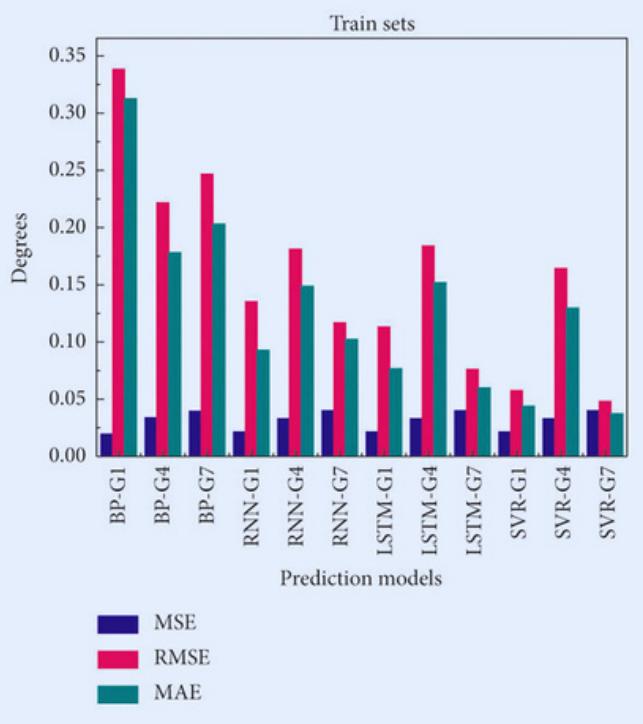
t o n a b i c e p t 以下の機能により、edition が採用されました。



- メモリセル：LSTMはメモリセルを使用して長期間にわたって情報を保存し、モデルが複数の期間にわたる依存関係を学習できるようにします。
- ゲート：LSTMは入力ゲート、出力ゲート、忘却ゲートを利用して情報の流れを制御するため、シーケンス予測タスクに効果的です。
- ノイズに対する堅牢性：LSTMは時系列データ内のノイズや外れ値を処理できるため、実際のアプリケーションに適しています。

# 評価指標

すべてのモデルには長所がありますが、LSTMは複雑な関係と時間的ダイナミクスを効果的にモデル化できる点で際立っています。



## 指標

### 平均絶対誤差

## 説明

- 一連のエラーの平均の大きさを測定します  
方向を考慮せずに予測します。MAEが低いほど、予測パフォーマンスが優れていることを示します。
- の二乗の平均を計算します  
エラーを分類し、エラーが大きいほど重み付けが高くなります。MSEが低いほど、モデルの精度が高くなります。
- 精度をパーセンテージで測定します。  
解釈が容易になります。MAPE値が低いほど、パフォーマンス。

### 平均二乗誤差

### 平均パーセンテージ誤差

すべてのモデルには長所がありますが、LSTMは複雑な関係と時間的ダイナミクスを効果的にモデル化する能力が際立っています。さまざまな評価指標を組み合わせると、通常、LSTMモデルは一貫してMAE、MSE、およびRMSE値により、特に株価のような変動の大きい領域での時系列予測に最適です。

# 検証結果

- **仮説1: 特徴選択によるモデルの改善**

株価予測に最も貢献する関連特徴を選択することで、  
モデルの精度と効率を向上させ、予測性能を向上させる  
より幅広い機能セットを使用する場合と比較して。

- 相関分析:相関係数を使用して、相関の高い特徴を特定し、  
相関性の高い各ペアから 1 つの特徴のみを保持します。
- モデルからの特徴の重要性: アルゴリズム (決定論的モデル、リッジ回帰など) を使用して特徴の重要性を評価し、重要度スコアに基づいて上位 N 個の特徴を保持します。
- 再帰的特徴除去 (RFE) : RFEを実装して、特徴を再帰的に削除し、  
最適な特徴数に達するまでモデルのパフォーマンスを評価します。

## 相関分析

相関係数を使用して相関の高い特徴を識別し、1つの特徴のみを保持する  
相関性の高い各ペアから。

例:線形回帰では、モデルが過剰適合を示し、精度が低下したため、すべての特徴を含めませんでした。この問題を回避するために、Lagと  
IQR、LAGなどの相関性の高い特徴のみを含む傾向特性



### 期待される結果:

予測精度の向上: 特徴量を適用した後、モデルは精度指標（MAE、MSE、RMSEの低下など）が向上することが期待されます。

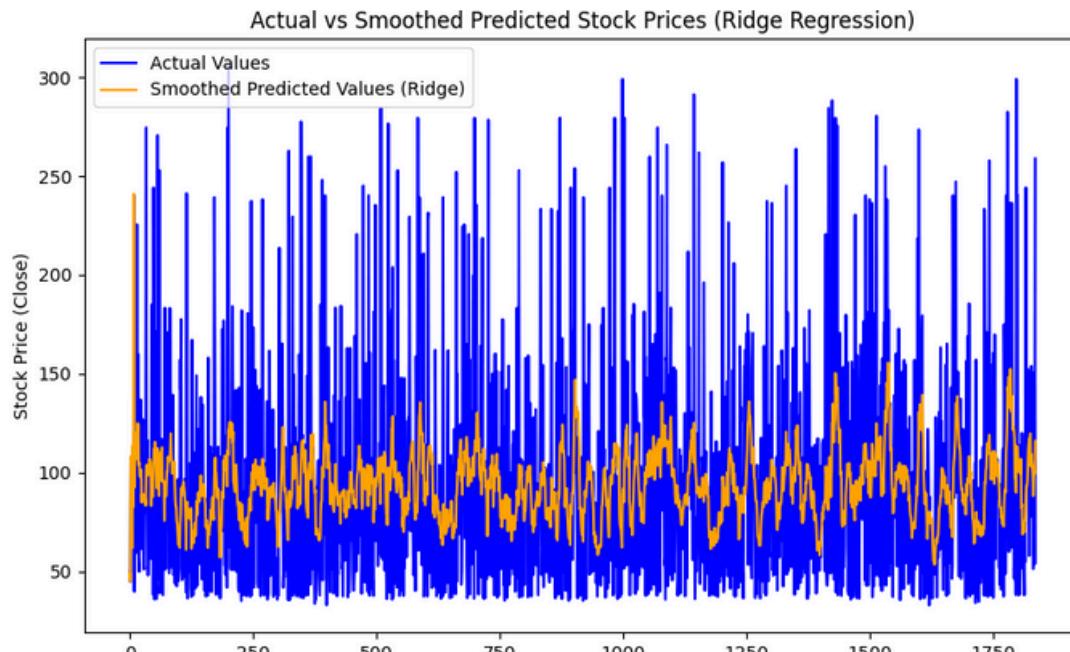
選択。

# 機能の重要性

アルゴリズム（決定論的モデル、リッジ回帰など）を使用して特徴の重要性を評価し、重要度スコアに基づいて上位N個の特徴を保持する

例: 線形モデルでは複雑なパターンが隠れてしまうことが多いため、精度を最大限に高めるには、代替機能を使用して上位N個の機能をリストし、Quadのように制限します。

線形モデルの代わりに、各反復でデータ特徴を更新する



## 期待される結果:

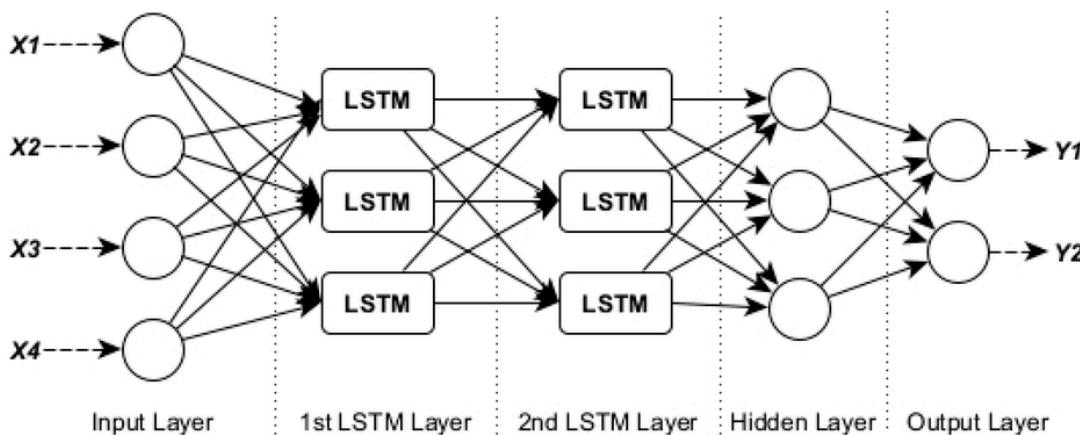
トレーニング時間の短縮：機能セットが削減されたため、トレーニングプロセスが速くなり、より多くの効率的な実験。

- 同じデータセットのクアッドポリマーモデルのリッジ回帰は、  
線形モデル

# 再帰的特徴除去 (RFE)

RFEを実装して、特徴を再帰的に削除し、モデルのパフォーマンスを評価します。  
最適な機能数に達しました。

- `lstm_regressor = KerasRegressor(build_fn=create_lstm_model、エポック=50、バッチサイズ=32、詳細=0)`
- セレクター = RFE(推定器=lstm\_regressor、n\_features\_to\_select=5、ステップ=1) セレクター =  
`セレクター.fit(X_train_reshaped、y_train)`
- `selected_features = selector.support_.print(f"選択された機能: {np.where(selected_features)[0]}")`



## 期待される結果:

より良い一般化: 選択された特徴セット  
は、新しい、より一般的なモデルへの一  
般化能力を高める可能性が高い。  
目に見えないデータにより、検証およびテス  
トセットのパフォーマンスが向上します。

- LSTMはRFEモデルを使用して、効率を損なうことなく時間の遅れを排除することができます。

# 検証結果

- **仮説 2: データ前処理が精度に与える影響。**

欠損値や外れ値の処理などの技術を含む効果的なデータ前処理

除去、正規化、特徴選択により、精度が大幅に向上します。

株価予測のための予測モデル。

- 欠損値の処理 - 欠損値の補完
- 外れ値除去 - 生データセットからのノイズ除去
- 正規化とスケーリング - データのスケーリングにより予測結果が向上します
- 特徴選択 - データセットに最適な特徴を選択し、独自の特徴を追加します

分析

# 欠損値の処理

不正確な予測や偏った結果につながる可能性があります。適切なモデル、または KNN 補完などの高度な手法を使用すると、貴重な情報を得ることができます。

## 期待される結果:

欠損値を適切に処理したデータセットでトレーニングされたモデルは、

未解決の行方不明者への  
価値観。

## 期待される結果:

外れ値が除去されたデータセットでトレーニングされたモデルは、エラー率が低くなり、予測パフォーマンスが向上します。

# 外れ値の除去

外れ値の影響: 外れ値は結果を歪め、モデルの学習プロセスを誤らせる可能性があります。Zスコアや IQRなどの技術を使用して外れ値を特定して除去すると、データを安定させることができます。  
分布。

## 正規化とスケーリング

特徴の正規化または標準化（例：最小値、すべての特徴がモデルtに等しく寄与する

保証する  
veに

機能スケール



### 期待される結果:

- 正規化されたデータセットを使用するモデルは、より良い収束を示し、未加工のスケーリングされていない機能を使用した場合と比較したパフォーマンス メトリック。



### 期待される結果:

厳選された機能のサブセットは、より高い利用可能なすべてのモデルを使用したモデルと比較して、精度と複雑さが低い特徴。

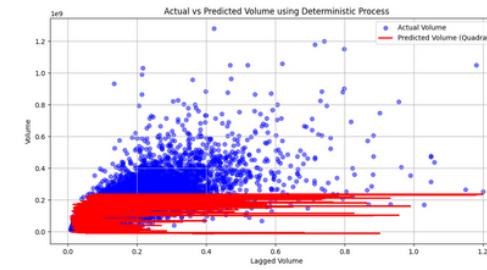
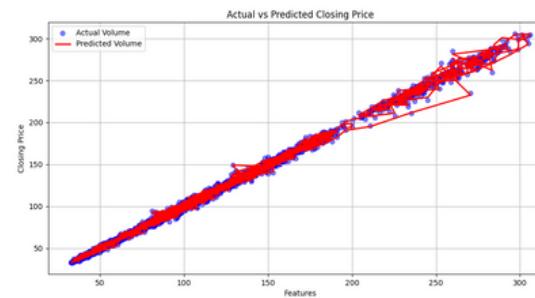
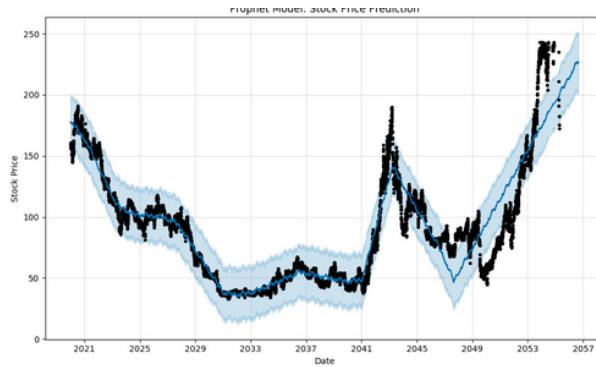
## 機能選択

再帰的特徴除去（RFE）やその他の特徴選択方法などの技術を使用するモデルのトレーニングに最も関連性の高い特徴を特定するのに役立ちます。

## 結果の概要

- 線形回帰：シンプルなモデルだが、非線形性と在庫の時間依存性に苦労したデータが不足し、予測が最適ではなくなります。
- ARIMA/SARIMA: 定常データには適しているが、次のような複雑なパターンを扱う柔軟性に欠けている。季節性と長期にわたる傾向。
- 預言者：季節性やトレンドの変化を効率的に検出できるが、短期的な予測精度には限界がある。より複雑なモデルと比較した用語予測。
- LSTM: 時系列データにおける長期的な依存関係と連続パターンを捉える能力により、他のモデルよりも優れた性能を発揮しました。非線形性と株価の時間的相関関係。

- LSTMの優位性: LSTMは、時間依存性に苦労する線形回帰などのより単純なモデルや、非定常データの動的パターンの処理に制限があるARIMA/SARIMAなどのより伝統的なモデルよりも優れたパフォーマンスを発揮しました。トレンドや季節性を考慮して設計されたモデルでさえ、  
Prophetのような検出技術は、短期的な価格変動を捉える精度が低かった。  
LSTMと比較して。



- 評価指標: LSTMは、次のような主要な評価指標で優れたパフォーマンスを示しました。  
平均二乗誤差 (MSE) 、平均二乗平方根誤差 (RMSE) 、平均絶対誤差 (MAE) 、  
株価予測の精度と取り扱いが重要なタスクに適していることを確認した。  
連續パターンが重要です。

# 今後の展望

今後、より高度な機能を統合することで、LSTMのパフォーマンスをさらに向上させることができます。

エンジニアリング、ハイパープラメータの最適化、外れ値の除去や正規化などのデータ前処理技術の使用など。

機能エンジニアリング

ハイパープラメータの最適化

前処理技術

外れ値の除去

# 高度な特徴エンジニアリング

株価予測には、次のような機能があります移動平均、ローリングフォーキャスト、早期停止またはテクニカル指標（例：相対力指数、ボリンジャーバンド）を追加することができます。これらの機能は、LSTMに学習するための追加のシグナルを提供します。

始値、高値、安値、終値、出来高等の基本的な入力を超えた、株価の複雑な傾向とパターン。

```
# Perform rolling forecast
history = list(train_data)
predictions = []

for t in range(len(test_data)):
    # Fit the ARIMA model to the current history
    model = ARIMA(history, order=order)
    model_fit = model.fit()

    # Forecast one step ahead
    forecast = model_fit.forecast(steps=1)
    predictions.append(forecast[0])

    # Update the history with the new value (use actual value from test_data)
    history.append(test_data.iloc[t])
```

ローリング予測では、モデルは過去のデータのウィンドウでトレーニングされ、新しいデータが到着するたびに定期的に更新されます。新しいデータ ポイントが追加されるたびに、モデルは最新のデータ ポイントセット(スライディング ウィンドウ)を使用して再トレーニングされ、変化する傾向に適応できるようになります。

```
model.add(Dense(units=1))

# Compiling the model
model.compile(optimizer='adam', loss='mean_squared_error')

# Early stopping to avoid overfitting
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

# Training the model
history = model.fit(X_train, y_train, epochs=20, batch_size=64, validation_data=(X_test, y_test), callbacks=[early_stopping], verbose=1)
```

早期停止は、  
トレーニング。検証セットでのモデルのパフォーマンスを監視し、一定数のエポック後にパフォーマンス（精度や損失など）が悪化し始めたら、  
トレーニングへの過剰適合を避けるためにトレーニングは停止されるデータ。

# ハイパーパラメータの最適化

ハイパーパラメータは学習プロセスの前に設定される値であり、例えば学習率、層の数、各層のユニット、ドロップアウト率、エポック、バッチサイズモデルパラメータとは異なり、ハイパーパラメータはモデルの学習方法を制御する。

そしてそれがどれだけ速く収束するかです。

```
# Adding the first LSTM layer and Dropout
model.add(LSTM(units=50, return_sequences=True, input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Dropout(0.2))

# Adding a second LSTM layer
model.add(LSTM(units=50, return_sequences=False))
model.add(Dropout(0.2))

# Adding the output layer
model.add(Dense(units=1))

# Compiling the model
model.compile(optimizer='adam', loss='mean_squared_error')

# Early stopping to avoid overfitting
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

# Training the model
history = model.fit(X_train, y_train, epochs=20, batch_size=64, validation_data=(X_test, y_test), callbacks=[early_stopping], verbose=1)

# Evaluate the model
loss = model.evaluate(X_test, y_test)
print(f'Model Loss: {loss}')

# Make predictions
```

- グリッドサーチ: 範囲内のハイパーパラメータのすべての組み合わせを試します。指定された範囲。網羅的ではあるが、計算上特にLSTMモデルでは高価ですハイパーパラメータ。
- ランダムサーチ: 範囲内のハイパーパラメータの組み合わせをランダムにサンプリングします。グリッドよりも効率的です。検索して、適切な組み合わせをより早く見つけることができます。
- ベイズ最適化: より洗練されたアプローチで、確率モデルを構築し、最適なハイパーパラメータを予測します。以前の試行に基づいて、検索プロセスを効率的に最適化します。
- 自動ハイパーパラメータチューニングツール（AutoML）: OptunaやHyperptなどのツールを使用すると、LSTM モデルに最適なハイパーパラメータ。

# 前処理技術と外れ値の除去

これらの技術は、モデルの予測精度を高めるだけでなく、より信頼性が高く堅牢な予測を保証します。

データ内のノイズや無関係な変動を最小限に抑えることで、将来の見通しがどのように形成されるかを説明します。LSTM モデル:

```
# 1. **Smoothing using Rolling Window (moving average)**
df['Close_smoothed'] = df['Close'].rolling(window=5).mean()

# 2. **Outlier Removal using Z-Score**
z_scores = np.abs(stats.zscore(df['Close_smoothed'].fillna(df['Close'])))
df = df[(z_scores < 3)] # Removing outliers where z-score > 3

# 3. **Scaling (MinMaxScaler) for features**
scaler = MinMaxScaler()
df[features] = scaler.fit_transform(df[features])
df[[target]] = scaler.fit_transform(df[[target]])
```

- 平滑化手法（移動平均、指数平滑化など）は、短期的な変動を取り除くのに役立つ。  
変動と長期的な傾向を強調する  
株価予測のような時系列データ
- 今後の展望: より洗練されたスムージング  
季節分解やウェーブレット変換などの技術を使用して、時系列を傾向、季節、および  
残留成分を投入する前に  
LSTM モデル、トレンド予測の強化  
正確さ。