

AI モデルの開発・検証から、手動取引、そして完全自動売買までを一つの環境で実現する、非常に本格的で包括的な **AI 取引プラットフォーム**です。

以下に、アプリ全体の概要と各機能の詳細、そしてそれらがどのように連携して動作するのかを解説します。

---

## アプリ全体の概要

このアプリケーションは、大きく分けて以下の 4 つの役割を持つコンポーネントで構成されています。

1. **AI 開発・検証エリア** (AI 未来予測, バックテスト): AI モデルをトレーニングし、取引戦略の有効性を過去データで検証する場所。
2. **手動取引エリア** (デモ取引, 実取引): Bot に頼らず、ユーザー自身の判断で取引を行うためのツール。
3. **自動売買管理エリア** (Bot 管理ダッシュボード): 開発した AI と戦略を搭載した Bot を作成・管理・監視する司令塔。
4. **アカウント管理エリア** (アカウント設定): ユーザー情報や取引に必要な API キーを安全に管理する場所。

これらが連携し、`bot_worker.py` という 24 時間稼働の取引エンジンを動かすことで、完全な自動売買を実現します。

---

## 各機能の詳細と動作ロジック

### 1. アカウント設定 (5\_アカウント設定.py)

- **目的:** ユーザー情報の管理と、取引の準備を行うためのページです。
- **主な機能:**
  - **プロフィール更新:** 名前、メールアドレス、LINE 通知用の ID、Bot が計算に使う初期資金額を変更できます。
  - **パスワード変更:** ログインパスワードを安全に変更します。
  - **API キー設定:** Bybit での実取引に必要な API キーとシークレットキーを、暗号化して安全にデータベースに保存します。

- **動作ロジック:** streamlit-authenticator ライブラリで認証を管理し、utils.py の暗号化関数を使って API キーを保護します。全ての情報は database.py で定義された User テーブルに保存されます。
- 

## 2. 🧠 AI 未来予測 (1\_AI 未来予測.py)

- **目的:** 価格予測 AI (Price Predictor) のモデルをトレーニングし、その性能を視覚的に確認するための「AI 開発室」です。
  - **主な機能:**
    - **モデルの学習:** 通貨ペアを指定し、Bybit から大量の過去データを取得して、価格予測用の Keras モデルをゼロから学習させます。
    - **モデルの保存:** 学習させたモデル (.keras) と、学習に使ったスケーラー (.joblib) を saved\_models/フォルダに保存します。
    - **未来予測:** 保存済みのモデルを読み込み、未来の価格変動を予測してチャートに表示します。
  - **動作ロジック:** utils.py に実装されたデータ取得、特徴量計算、モデル学習・予測の関数群を呼び出します。ここで保存されたモデルが、bot\_worker.py で稼働する価格予測 AI の「頭脳」そのものになります。
- 

## 3. 📉 バックテスト (2\_バックテスト.py)

- **目的:** core\_logic.py に定義された**基本取引戦略**が、過去の相場でどれほどの成績を上げたかを検証する「戦略シミュレーター」です。
- **主な機能:**
  - **パラメータ調整:** RSI や ADX のしきい値、リスク管理設定などを自由に変更できます。
  - **パフォーマンス測定:** 指定した期間とパラメータでシミュレーションを実行し、最終資産、勝率、プロフィットファクターなどの性能指標を確認できます。
  - **可視化:** 資産の推移（エクイティカーブ）や、チャート上のどこで売買シグナルが出たかを確認できます。

- **動作ロジック:** このページは、3 体の AI を使わずに、基本戦略そのものの強さをテストします。utils.run\_parameterized\_backtest 関数が、指定された過去データとパラメータに基づいて取引をシミュレートし、その結果を返します。ここで良い成績を出すパラメータを見つけることが、自動売買成功の第一歩となります。
- 




#### 4. 🎮 デモ取引 & 📈 実取引 (3\_デモ取引.py, 4\_実取引.py)

- **目的:** AI や Bot を介さず、ユーザーが直接、裁量で取引を行うためのトレーディングターミナルです。
  - **主な機能:**
    - **デモ取引:** 実際のお金を使わずに、本番そっくりの環境で取引の練習ができます。ポートフォリオは PC 内にファイルとして保存されます。
    - **実取引:** 登録した API キーを使い、ご自身の Bybit アカウントで実際に注文・決済ができます。口座残高や現在のポジションもリアルタイムで確認できます。
  - **動作ロジック:** どちらのページも、utils.py の get\_current\_price や place\_live\_market\_order といった関数を直接呼び出して動作します。これらは Bot の自動売買とは完全に独立しています。
- 

#### 🤖 Bot 管理ダッシュボード (app.py) と総合的な取引戦略

このページこそが、アプリケーション全体の司令塔です。

- **目的:** これまで開発・検証してきた AI と戦略を搭載した、オリジナルの自動売買 Bot を作成し、その稼働を管理・監視します。
- **主な機能と判断フロー:**
  1. **Bot の作成と設定:** ユーザーはこの画面で Bot を作成し、取引する通貨ペアや、バックテストページで見つけた最適な戦略パラメータを設定します。そして、3 体の AI (価格予測 📊、センチメント 🗣️、レジーム 📈) のうち、どれを有効にするかを個別に選択します。

2. **Bot の稼働**: ユーザーが「稼働」ボタンを押すと、データベースの Bot 情報が「アクティブ」になります。
3. **bot\_worker.py の検知**: PC の裏側で 24 時間動き続けている bot\_worker.py が、「アクティブ」な Bot を検知し、処理を開始します。
4. **情報収集と AI 分析**: bot\_worker は、価格データ (ccxt) とニュースデータ (collectors) をリアルタイムで収集し、有効になっている AI に分析を依頼します。
5. **基本シグナルの生成**: core\_logic.py が、テクニカル指標に基づき「買い」か「売り」の基本シグナルを生成します。
6. **AI による検証 (フィルター)**: 生成された基本シグナルは、有効化されている AI 専門家チームによるレビューを受けます。
  - **\*\*レジーム AI \*\***が「今の相場環境はこの戦略に合っているか？」をチェック。
  - **センチメント AI ** が「市場心理はシグナルに同調しているか？」をチェック。
  - **\*\*価格予測 AI \*\***が「短期的な値動きの確率が高いか？」をチェック。
7. **最終判断と実行**: 全ての AI の承認が得られたシグナルのみが「最終承認」となり、utils.py を通じて取引所に注文が出されます。
8. **結果の記録と表示**: 実行された取引の結果はデータベースに記録され、この app.py の「現在のオープンポジション」や「取引履歴」にリアルタイムで反映されます。

この一連の流れにより、単純なテクニカル分析の弱点を、複数の AI が多角的な視点で補う、非常に高度で堅牢な自動売買システムが完成しています。

今回アプリに導入した 3 体の AI について、それぞれの機能、動作ロジック、そして取引戦略における役割を詳しく解説します。

---

## システム全体の動作コンセプト

このアプリケーションは、まず `core_logic.py` の基本的なテクニカル分析によって「買い」または「売り」の**基本シグナル**を生成します。その後、有効化されている AI が\*\*「専門家チーム」\*\*のようにそのシグナルを多角的に検証し、最終的な取引の是非を判断する、という多層的な構造になっています。

---

### 1. 価格予測 AI (Price Predictor) 🧠

この AI は、短期的な価格変動を数値で予測する「**テクニカルアナリスト**」の役割を担います。

- **目的:** 「次の足の終値は、現在の終値からおよそ何%変動するか?」という問いに答えます。
- **ロジック:**
  - **入力:** `core_logic.py` で計算された、移動平均線(SMA, EMA)、RSI、ボリンジャーバンド、ATR など、多数のテクニカル指標。
  - **処理:** `saved_models/` フォルダに保存されている学習済み Keras モデル (TensorFlow) とスケーラー (`scaler_*.joblib`) を読み込みます。入力されたテクニカル指標の最新パターンを、過去の膨大なデータと照合し、最も可能性の高い未来の価格を算出します。
  - **出力:** `predicted_change_rate` という数値。例えば「+0.15%」のように、具体的な価格変動率を返します。
- **取引戦略における役割:** 量的フィルターとして機能します。例えば、基本戦略から「買い」シグナルが出ても、この AI の予測が「-0.1%」のようにマイナスであったり、閾値を下回る弱い上昇予測だったりした場合、そのエントリーは見送られます。逆に、強い上昇予測が出ていれば、シグナルの信頼性を補強します。決済時にも、相場の反転を予測して早期利確・損切りを行う判断材料となります。

---

## 2. センチメント分析 AI (Sentiment Analyzer) 🧠

この AI は、市場に参加している人々の「感情」や「心理」を読み解く「**市場心理アナリスト**」です。

- **目的:** 「現在、市場参加者はこの通貨に対して楽観的か、悲観的か？」を判断します。
- **ロジック:**
  - **入力:** collectors/text\_data\_collector.py が NewsAPI を通じてリアルタイムで収集した、最新のニュースヘッドライン（英語）。
  - **処理:** models/sentiment\_analyzer/フォルダに保存されている学習済みモデル（sentiment\_model.pkl）とトークナイザー（tokenizer.pkl）を使用します。トークナイザーがニュースの単語を数値ベクトルに変換し、モデルがその数値からポジティブ/ネガティブの確率を判定します。複数のニュースを集計し、最終的なセンチメントスコアを算出します。
  - **出力:** -1.0（極度に悲観）から+1.0（極度に楽観）までの sentiment\_score と、Positive, Negative, Neutral といった sentiment\_label を返します。
- **取引戦略における役割:** 定性的・心理的フィルターとして機能します。例えば、基本戦略で「買い」シグナルが出ていても、センチメント AI が「市場心理は非常にネガティブだ」と判断した場合、「今は買うべきではない危険な局面だ」としてエントリーを見送ります。これにより、テクニカル指標だけでは捉えきれない、突然の悪材料や市場の恐怖感に起因するダマシのシグナルを回避します。

---

## 3. 市場レジーム分類 AI (Regime Classifier) 📈




この AI は、より大きな時間軸で相場全体を俯瞰し、現在の市場環境を判断する「**ストラテジスト（戦略家）**」です。

- **目的:** 「現在の相場は、明確なトレンドが発生しているか、それとも方向感のないレンジ相場か？」を分類します。

- **ロジック:**
    - **入力:** ccxt から取得した長期（4 時間足など）の価格データ（OHLCV）。
    - **処理:** models/regime\_classifier/フォルダの学習済みモデル（regime\_model.pkl）とスケーラーを使用します。長期移動平均線の傾きやボラティリティといった特徴量を計算し、現在の相場が「**Bull Trend**（上昇トレンド）」「**Bear Trend**（下降トレンド）」「**Ranging**（レンジ相場）」のどれに最も近いかを分類します。
    - **出力:** regime というラベル（例: Bull Trend）を返します。
  - **取引戦略における役割:** 戦略的フィルターとして機能し、取引戦略と市場環境のミスマッチを防ぎます。例えば、基本戦略がトレンドフォロー型の場合、この AI が「Ranging」と判断すれば、無駄なエントリー（往復ビンタ）を避けるためにシグナルを見送ります。また、「買い」シグナルが出ても、市場全体が「Bear Trend」の状況下であれば、より慎重な判断を下すことができます。
- 

### 3 体の AI の連携と最終判断

bot\_worker.py の run\_bot\_logic 関数内で、これら 3 体の AI は以下のように連携します。

1. core\_logic が基本シグナル（例：「買い」）を生成。
2. **\*\*レジーム AI**  **\*\***がチェック：「市場は上昇トレンドか？」（OK なら次へ）
3. **センチメント AI**  **\*\***がチェック：「市場心理は悪くないか？」（OK なら次へ）
4. **\*\*価格予測 AI**  **\*\***がチェック：「短期的な上昇が見込めるか？」（OK なら次へ）
5. 全ての AI の承認が得られた場合、**最終的にエントリーが実行される**。

いずれかの AI が「待った」をかければ、その取引は見送られます。これにより、単純なテクニカル分析だけでは防ぎきれない多くのリスクを回避し、より精度の高い取引を目指します。