Stock Forecasting Model

——Eclipse Loom Forecast

Team J

Zhang Yanxiao 23222069

Zhou Zixuan 23271558

Zhang Zhenghan 23267925

Zeng Weiming 23267062

Zou Boyu 23258527

Cui Zexu 23259175

01. Problem Statement

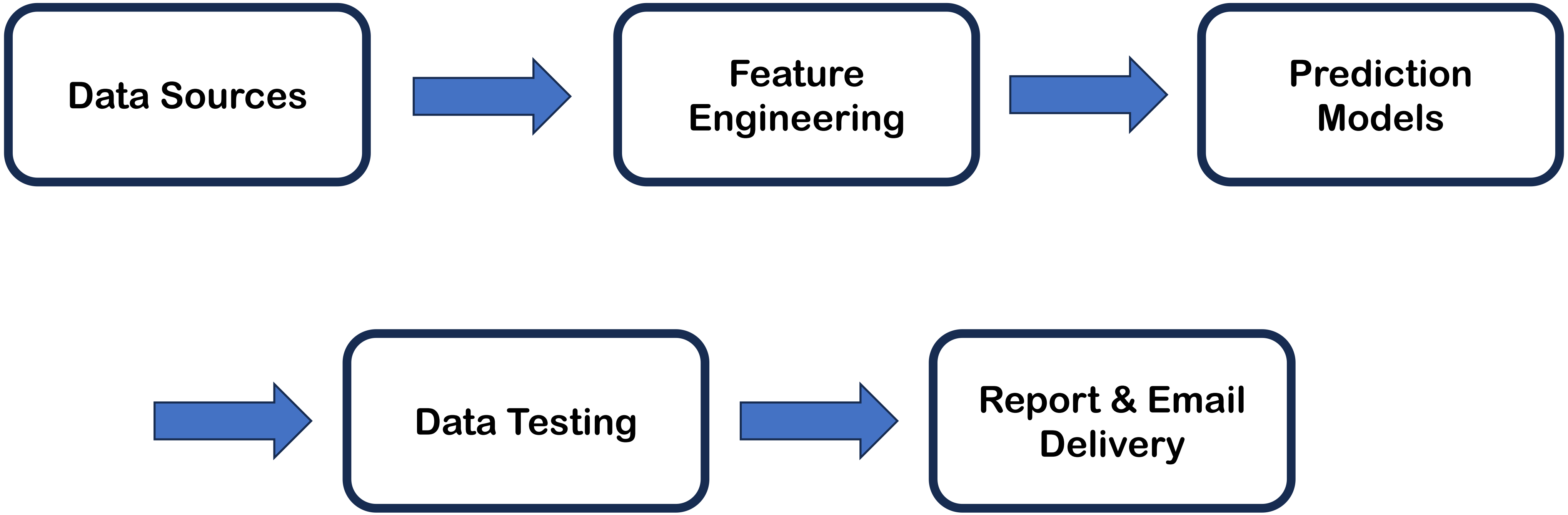
* *Making Investments is a complicated procedure to process*

While investors can identify target stock types and initial combinations, they lack clarity on two key dimensions of their investments:

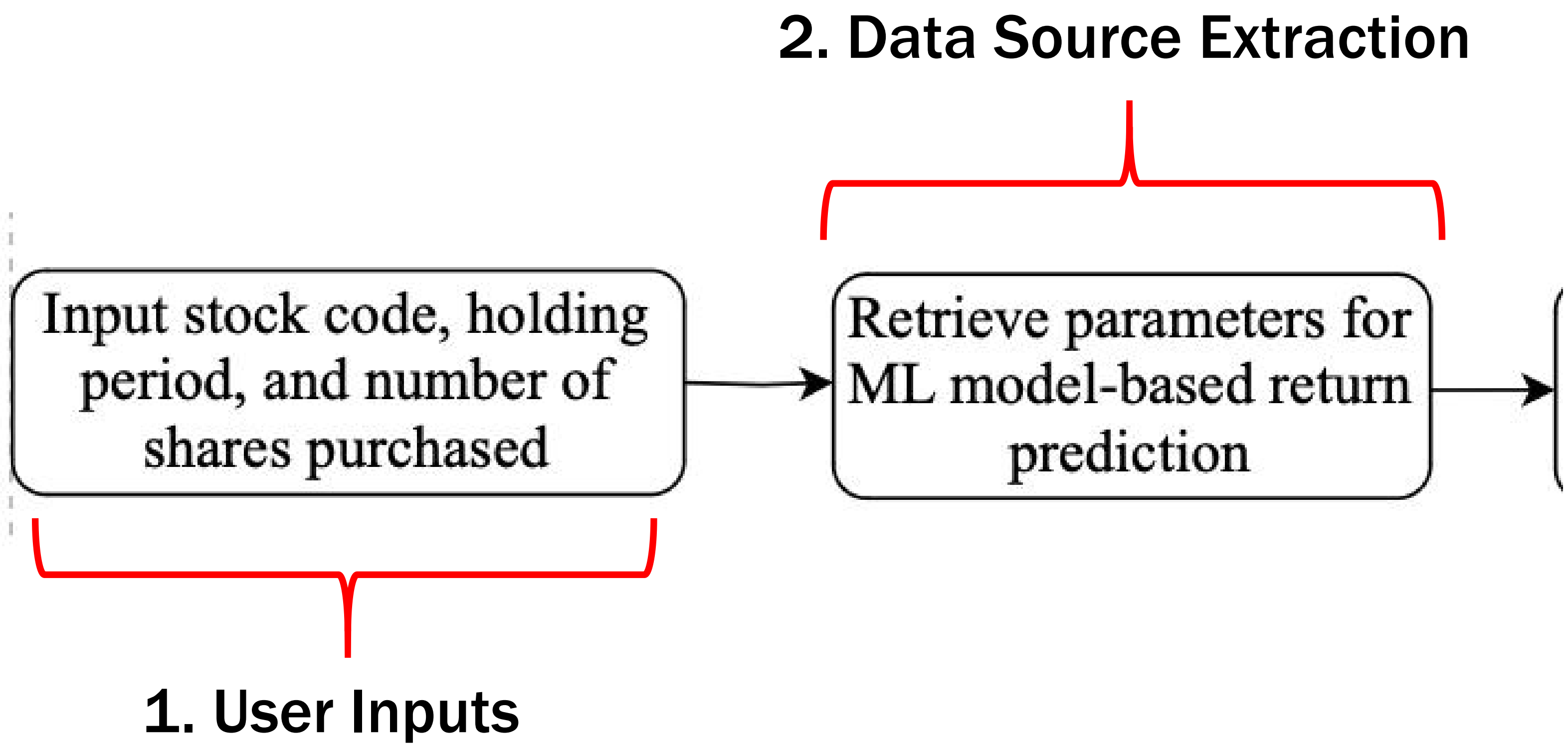
1. ***Forward-looking Performance (e.g., expected returns)***
2. ***Potential market shifts (e.g., sector volatility, macroeconomic events)***

Our project aims to resolve these gaps by developing a **Portfolio Return Prediction System**

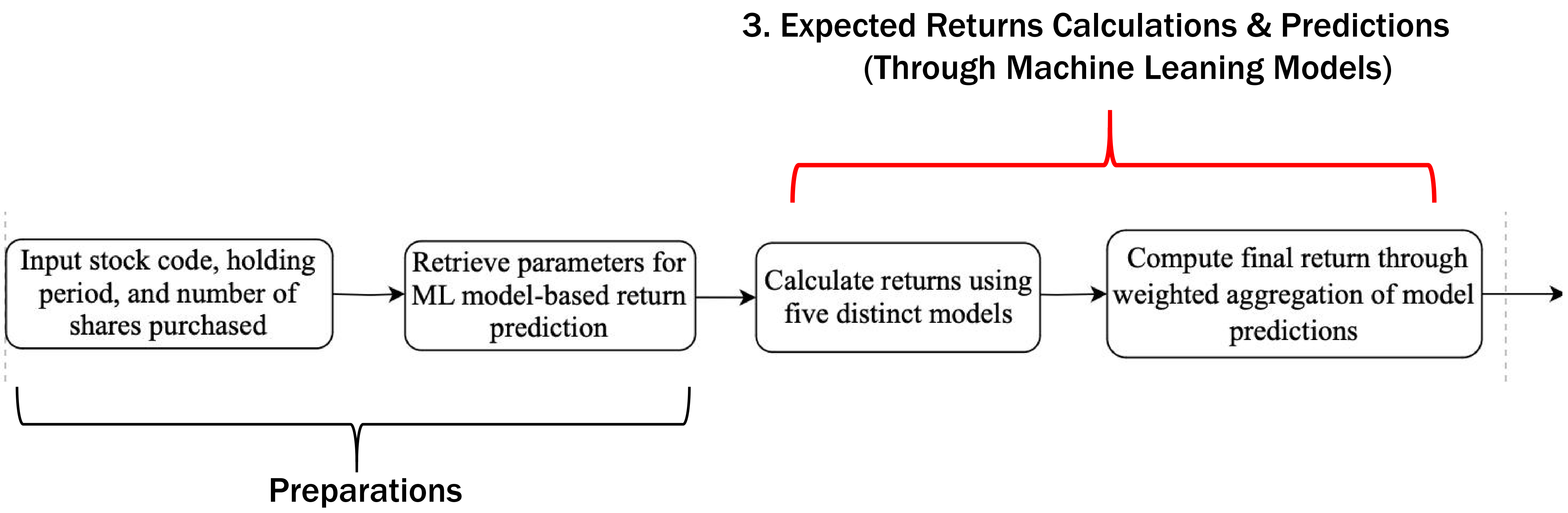
02. Solution Overview



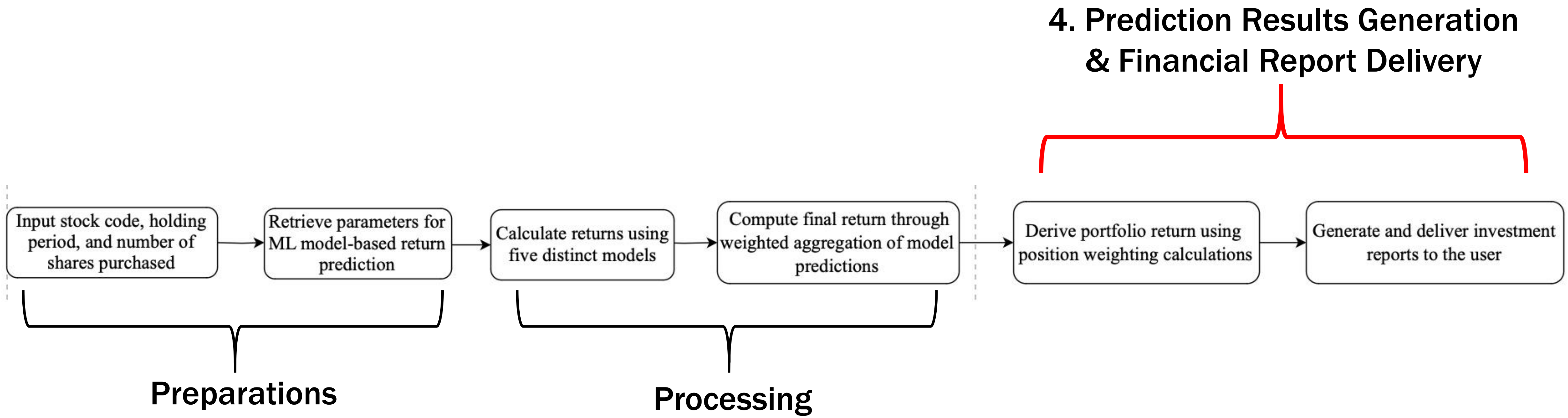
03. Stock Forecasting – Preparation Stage



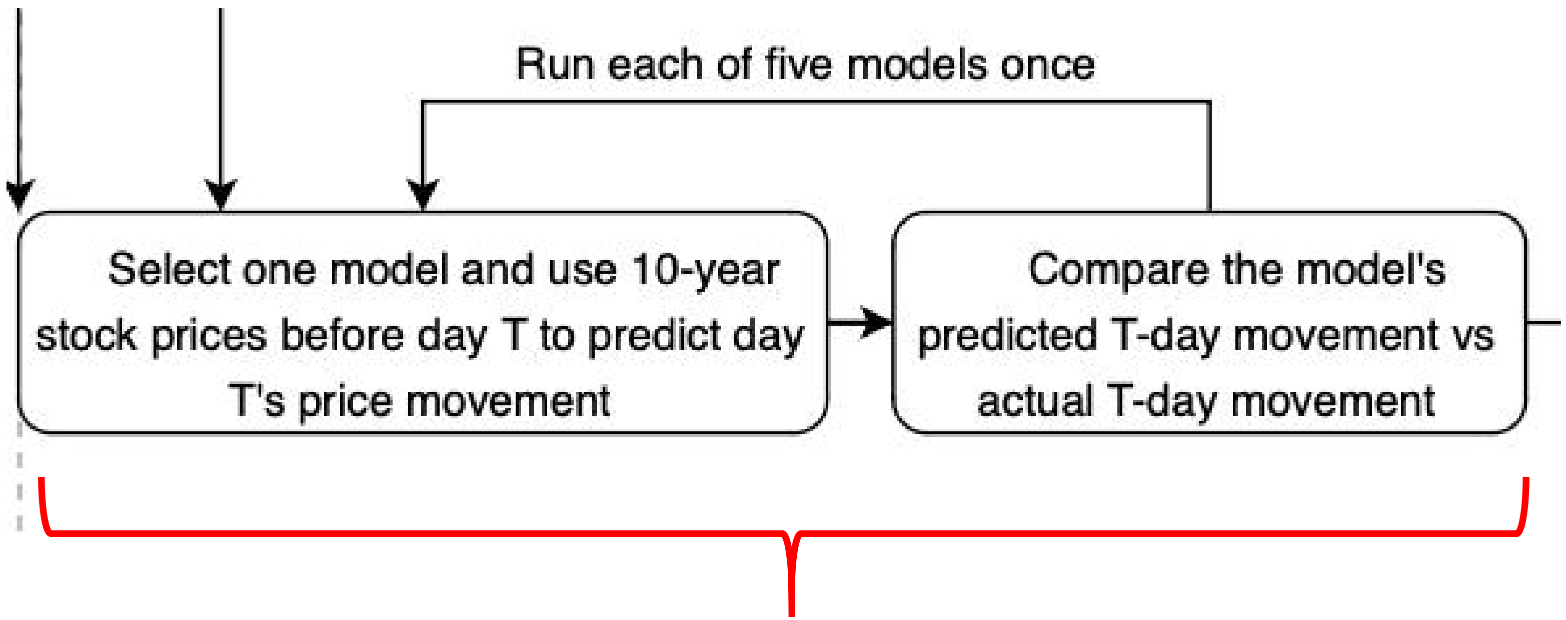
03. Stock Forecasting – Processing Stage



03. Stock Forecasting – Output & Reporting Stage



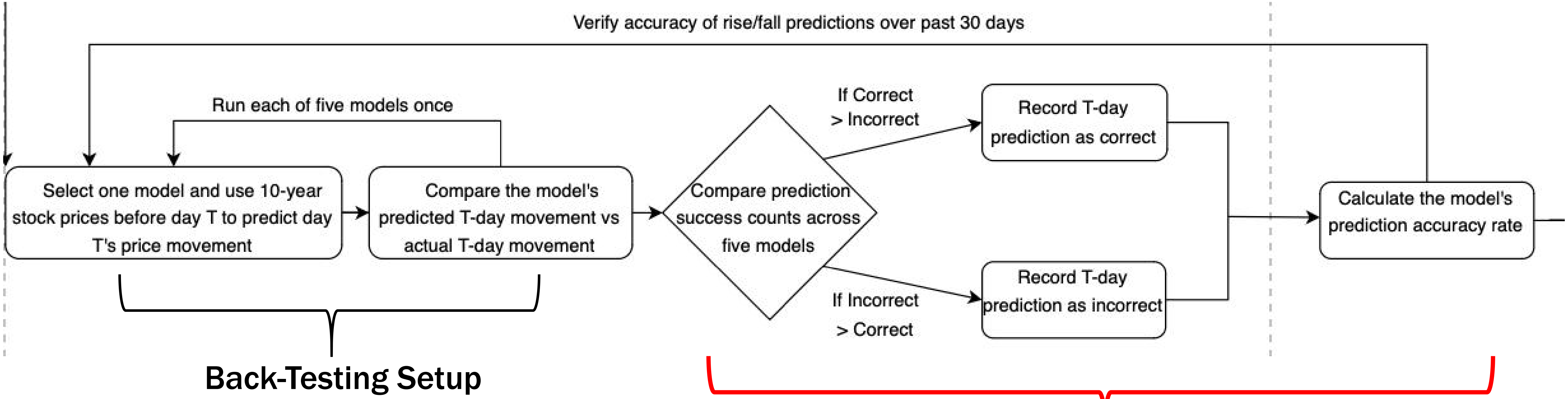
03. Testing



We use "Back-Testing" to validate model performance:

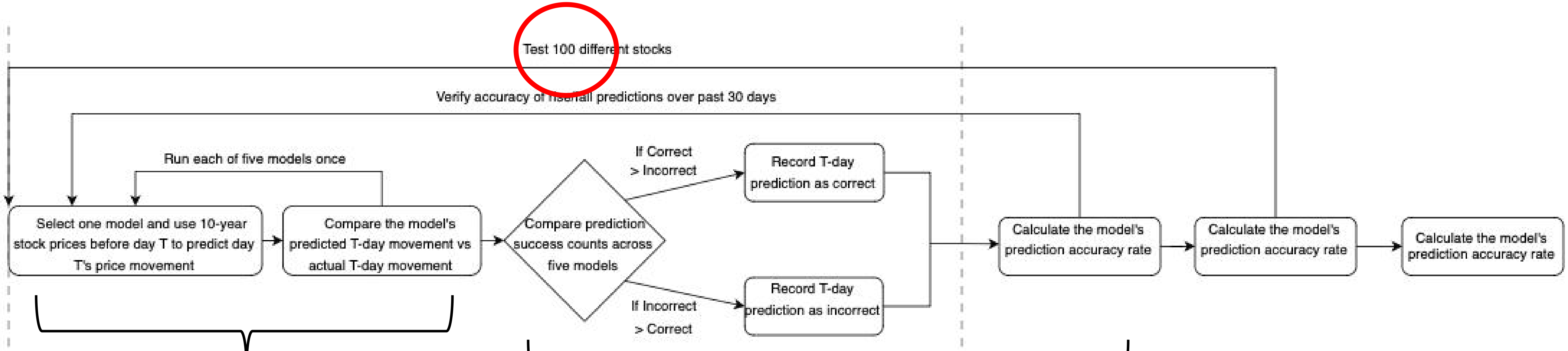
1. Select T days back as an example
2. Use (10-year sources – T days sources) to train and predict the return on the T's day (where index i = 0)
3. Compare the actual return with our predicted return
4. Increment i + 1

03. Testing



Record the ratio of all correct predictions over the total number of comparisons to acquire the estimated prediction accuracy of our system. (one-stock-simulation)

03. Testing



Back-Testing Setup

Accuracy Rate (one-stock-simulation)

Do a hundred-stock-simulation to reduce randomness and get the final average prediction accuracy rate of our system

04. Complex Logic Highlights – Retrieve Data

```python
# get stock data
def get_stock_data(ticker, period="10y"):
    stock = yf.Ticker(ticker)
    df = stock.history(period=period)
    df.index = pd.to_datetime(df.index)
    df.sort_index(inplace=True)
    return df

# get gpr
def get_gpr_data():
    url_gpr = "https://www.matteoiacoviello.com/gpr_files/data_gpr_export.xls"
    try:
        gpr_df = pd.read_excel(url_gpr, engine='xlrd')
        gpr_df["Date"] = pd.to_datetime(gpr_df["month"].astype(str).str.replace("M", "
        return gpr_df
    except Exception as e:
        print("Error from getting GPR data:", e)
        return None


#
def get_fred_data():
    """Getting FRED economy data"""
    try:
        # using pandas_datareader to get FRED data
        import pandas_datareader.data as web
        start = datetime(2000, 1, 1)
        end = datetime.now()

        # get unemployment rate
        unemployment = web.DataReader('UNRATE', 'fred', start, end)
        # get inflation rate
        inflation = web.DataReader('CPIAUCSL', 'fred', start, end)
        inflation = inflation.pct_change(12) * 100  # year inflation
        # get FED rate
        fed_rate = web.DataReader('FEDFUNDS', 'fred', start, end)

        # merge data
        economic_data = pd.concat([unemployment, inflation, fed_rate], axis=1)
        economic_data.columns = ['Unemployment', 'Inflation', 'FedRate']
        return economic_data
    except Exception as e:
        print(" error occurred when getting FED data: ", e)
        return None
```

By using ***yfinance library*** to get real time stock data

Using a url and the excel reader to retrieve Geopolitical Risk (GPR) Index data

Import ***pandas_datareader .data library*** to read FRED economy data

04. Complex Logic Highlights – Check & Merge Data

```python
def merge_stock_gpr(stock_df, gpr_df):
    stock_df = stock_df.copy()
    gpr_df = gpr_df.copy()

    # make sure the index is datetime and no zone
    stock_df.index = pd.to_datetime(stock_df.index)
    stock_df.index = stock_df.index.tz_localize(None)

    # keep the date index as a column and extract the year and month
    stock_df['Date'] = stock_df.index
    stock_df['Year'] = stock_df['Date'].dt.year
    stock_df['Month'] = stock_df['Date'].dt.month

    # deal with gpr_df: turn 'month' column into datetime and clear the zone
    gpr_df['month'] = pd.to_datetime(gpr_df['month'])
    gpr_df['month'] = gpr_df['month'].dt.tz_localize(None)
    gpr_df['Year'] = gpr_df['month'].dt.year
    gpr_df['Month'] = gpr_df['month'].dt.month
```

Aligning different time formats and resampling data across varying frequencies (daily, monthly) requires robust data handling logic.

```python
def is_trading_day(date, ticker):
        ticker = ticker.strip().lstrip('$')
        if ticker.endswith(".HK"):
            exchange = "HKEX"
            calendar = mcal.get_calendar("HKEX")
        elif ticker.endswith(".SS") or ticker.endswith(".SZ"):
            exchange = "SSE"
            calendar = mcal.get_calendar("SSE")
        else:
            exchange = "NYSE"
            calendar = mcal.get_calendar("NYSE")

        date = pd.to_datetime(date).replace(hour=0, minute=0, second=0, microsecond=0)
        date_str = date.strftime('%Y-%m-%d')
        schedule = calendar.valid_days(start_date=date_str, end_date=date_str)

        is_trading = not schedule.empty

        return is_trading
```

Import ***pandas_market_calendars library*** to check whether a given date is a valid trading day for a specific stock exchange

04. Complex Logic Highlights – Train Models

```python
models = {
    'Random Forest': RandomForestRegressor(
        n_estimators=100,
        max_depth=4,
        min_samples_split=15,
        min_samples_leaf=15,
        max_features='sqrt',
        n_jobs=-1,
        random_state=42
    ),
    'Gradient Boosting': GradientBoostingRegressor(
        n_estimators=100,
        learning_rate=0.01,
        max_depth=3,
        subsample=0.7,
        min_samples_split=15,
        min_samples_leaf=15,
        random_state=42
    ),
    'XGBoost': xgb.XGBRegressor(
        n_estimators=100,
        learning_rate=0.01,
        max_depth=3,
        gamma=0.1,
        min_child_weight=7,
        subsample=0.7,
        colsample_bytree=0.7,
        reg_alpha=0.1,
        reg_lambda=1.0,
        n_jobs=-1,
        random_state=42
    ),
```

```python
    'LightGBM': lgb.LGBMRegressor(
        n_estimators=100,
        learning_rate=0.01,
        num_leaves=15,
        max_depth=4,
        min_child_samples=30,
        min_child_weight=1,
        subsample=0.7,
        colsample_bytree=0.7,
        reg_alpha=0.1,
        reg_lambda=1.0,
        n_jobs=-1,
        random_state=42,
        verbose=-1
    ),
    'ElasticNet': ElasticNet(
        alpha=0.005,
        l1_ratio=0.7,
        max_iter=1000,
        tol=0.001,
        random_state=42
    ),
```

Import five predictive model libraries from sklearn:

1. RandomForestRegressor
2. GradientBoostingRegressor
3. XGBoost
4. LightGBM
5. ElasticNet

to predict future returns

04. Complex Logic Highlights – Prediction in Action

```python
# Training test set split (time series split, with the last 20% used as the test set)
split_idx = int(len(X) * 0.8)
X_train, X_test = X.iloc[:split_idx], X.iloc[split_idx:]
y_train, y_test = y.iloc[:split_idx], y.iloc[split_idx:]

# train all models and give the result
models_results = train_and_evaluate_models(X_train, X_test, y_train, y_test, X.columns)

# generate the prediction dataframe
full_pred = pd.DataFrame(index=df_target.index)
full_pred['Actual'] = df_target['Future_Return']

# Add predictions for each model
for name, result in models_results.items():
    full_pred[f'Pred_{name}'] = np.nan
    # Only partially fill in the test set
    full_pred.loc[X_test.index, f'Pred_{name}'] = result['model'].predict(X_test)

# Get the latest forecast for the last day
X_latest = df_target[feature_cols].iloc[[-1]]
predictions = {}
for name, result in models_results.items():
    model = result['model']
    pred = model.predict(X_latest)[0]
    predictions[name] = pred

return predictions, models_results, full_pred
```

Split data into training set and test set (80%/20%)

Train all five predictive models

Extract the features of the last day as X_latest and let models use latest data to predict what will happen in the next N days

04. Complex Logic Highlights – Testing

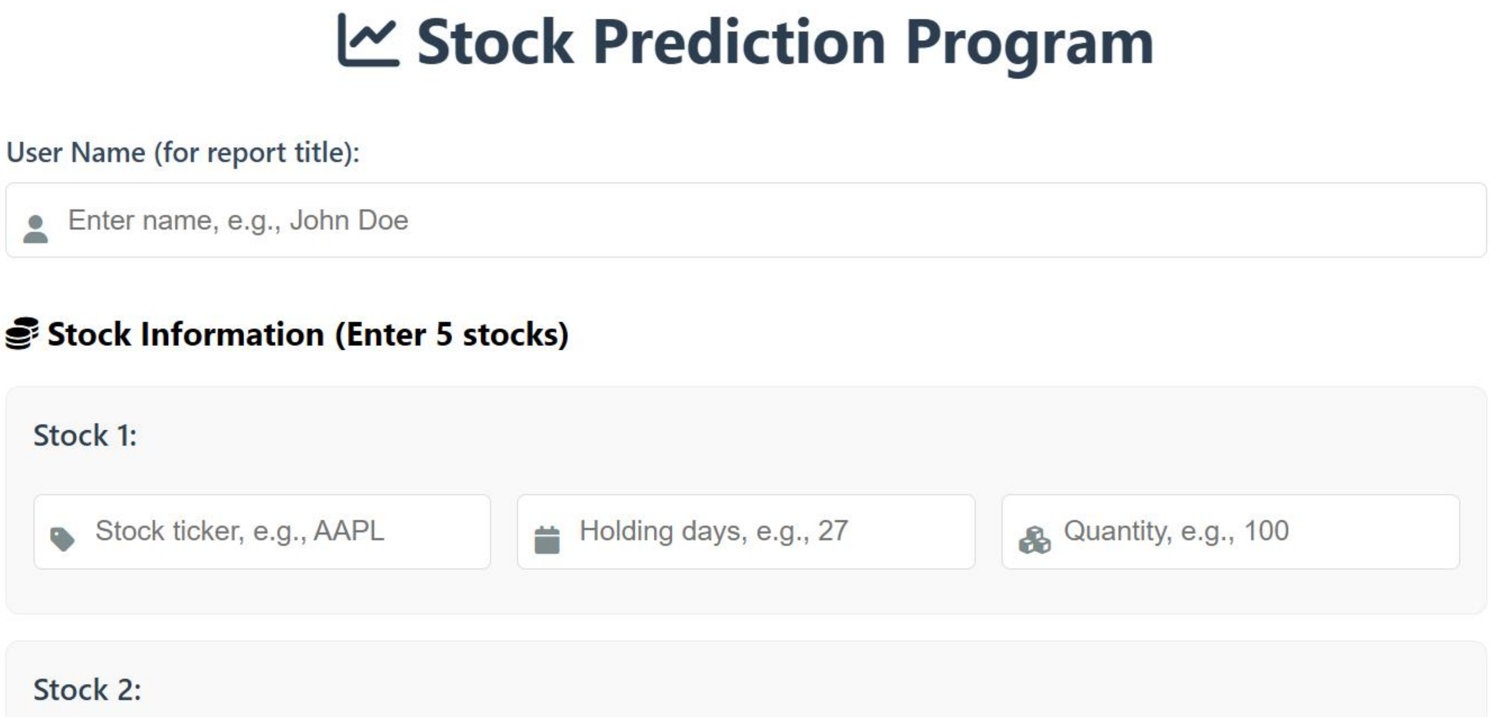
```
feature_df.drop(feature_df.tail(60).index, inplace=True)  # 删除最后60行
for i in range(30):
    count += 1
    df_target = add_future_return(feature_df, N)
    feature_df.drop(feature_df.tail(1).index, inplace=True)
    predictions, models_results, prediction_df = train_and_predict(feature_df, N)
    df_target.drop(df_target.tail(1).index, inplace=True)
    real_return=df_target.tail(1)['Future_Return'].iloc[0]
    k=0
    for j in predictions.values():
        if j>0:
            k+=1

    if k>=3 and real_return>0:
        trueValue+=1
    elif k<3 and real_return<0:
        trueValue+=1
print(trueValue/count)
```

Reduce the impact of the recent tariff events

Compare the predicated returns with actual returns in the past 30 days and record the final accuracy ratio

05. Other Interesting Features



User Input Interface

05. Other Interesting Features

weimingceng37@gmail.com
发送至 我

Enhanced Stock Prediction with Multiple Models

Prediction deadline: Future30 days

===== AAPLpredict results =====
ElasticNet: 1.31%
XGBoost: 0.64%
LightGBM: -0.10%
Gradient Boosting: -1.00%
Random Forest: -1.66%

===== MSFTpredict results =====
ElasticNet: 4.22%
LightGBM: 3.70%
XGBoost: 2.29%
Gradient Boosting: 1.99%
Random Forest: -1.99%

===== TSLApredict results =====
ElasticNet: 5.71%
Random Forest: 3.82%
Gradient Boosting: 1.34%
XGBoost: 0.71%
LightGBM: -0.76%

===== GOOGLpredict results =====
ElasticNet: 5.46%
LightGBM: 3.09%
XGBoost: 2.03%
Gradient Boosting: 1.94%
Random Forest: -2.37%

weimingceng37@gmail.com
发送至 我

Dear Investor,

Your projected portfolio return as of 2025-04-14 21:08:13 is 1.10%.
Total earning value is: 2286.667331314647 and total value is: 209685.66788063105.

Key Features:
- Integrates stock history, geopolitical risks, and macroeconomic data.
- Uses advanced algorithms (Random Forest, XGBoost) to identify volatility drivers.
- Employs ensemble models with 30-day rolling validation for calibrated projections.
- Refines outliers and ensures trend continuity for reliable insights.

Our methodology cross-verifies market signals with quantitative frameworks, balancing academic rigor with practical application. This equips institutional and active investors with actionable, data-driven foresight.

Thank you for your trust!

Best regards,
EclipseLoom Team
Date: 2025-04-14 21:08:13

1 个附件 · 已由 Gmail 扫描

investment_report.pdf
1.8 KB

Results Delivery via Email (with PDF attachment)

05. Other Interesting Features

stockName	AAPL	MSFT	TSLA	GOOGL	AMZN
holdingDays	7	30	26	28	29

Input Stock Names

& Holding Days

Dear Investor,

Your projected portfolio return as of 2025-04-14 21:08:13 is 1.10%.

Total earning value is: 2286.667331314647 and total value is: 209685.66788063105.

Key Features:

- Integrates stock history, geopolitical risks, and macroeconomic data.
- Uses advanced algorithms (Random Forest, XGBoost) to identify volatility drivers.
- Employs ensemble models with 30-day rolling validation for calibrated projections.
- Refines outliers and ensures trend continuity for reliable insights.

Our methodology cross-verifies market signals with quantitative frameworks, balancing academic rigor with practical application. This equips institutional and active investors with actionable, data-driven foresight.

Thank you for your trust!

Predicted Portfolio Return Rate

Expected Earning Prices

Key Features Illustration

Predicted Portfolio Report

Demo

THANK YOU