**🚀 End-to-End Data Science Project Guide**

**Project:** Stock Market Trend Prediction  
🎯 *Built to mirror real industry expectations — from planning to deployment & stakeholder-ready presentation*

**📂 Phase 1: Project Planning (Business Understanding)**

**🎯 Goal:**

Define the objective, scope, and value of the project — just like you'd do in a real company.

**✅ Steps:**

* **Define the Problem Statement:**

“Predict next-day stock market movement (Up/Down) using historical price trends and sentiment analysis from financial news.”

* **Set Business Goals:**
  + Help traders or investors make better entry/exit decisions.
  + Automate insights for faster decision-making.
* **Set Success Criteria (KPIs):**
  + Classification Accuracy ≥ 70%
  + Model Confidence > 0.7 for Buy/Sell signal
  + Minimal lag in prediction (can run daily)

**🗃️ Phase 2: Documentation & Tools Setup**

**📁 Tools & Structure (Just like a company would use):**

* **GitHub**: Version control (with branches like dev, main)
* **Notion / Google Docs**: Project planning + task board
* **Trello / Jira**: Sprint-based tracking (optional)
* **Folder Structure**:

bash

CopyEdit

├── data/ # Raw & processed

├── notebooks/ # EDA, modeling experiments

├── scripts/ # Preprocessing, training, prediction

├── models/ # Saved models

├── app/ # Streamlit app (optional)

├── tests/ # Model testing / unit tests

├── README.md

├── requirements.txt

└── reports/ # Visuals, evaluation results

**📊 Phase 3: Data Collection & Cleaning**

**✅ Steps:**

* Collect stock price data using yfinance
* Collect financial news headlines (News API or static CSV)
* Clean:
  + Missing values
  + Outliers
  + Format datetime
* Store clean version in /data/processed

**🛠 Tools:**

* pandas, numpy, yfinance, datetime, matplotlib, seaborn

**📈 Phase 4: EDA + Feature Engineering**

**✅ Tasks:**

* Generate indicators: SMA, EMA, RSI, MACD
* Create lag features
* Create sentiment scores
* Plot correlations and trends
* Identify what drives stock movement

**📂 Output:**

* /notebooks/eda\_stock.ipynb
* /data/features.csv
* Feature importance charts

**🧠 Phase 5: Modeling**

**✅ Tasks:**

* Baseline: Logistic Regression
* Main: XGBoost / Random Forest
* Bonus: LSTM (time-series aware)
* Target = “Trend” column (Up/Down)

**📊 Metrics:**

* Accuracy, Precision, Recall, Confusion Matrix

**💬 Phase 6: Sentiment Analysis Integration (NLP)**

**✅ Tasks:**

* Score news headlines using TextBlob, VADER, or BERT
* Merge sentiment data with price data
* Test model performance with and without sentiment

**🧪 Phase 7: Testing and Evaluation**

**✅ Tasks:**

* Write test cases for:
  + Preprocessing
  + Feature pipeline
  + Model inference
* Create a /tests/ folder
* Evaluate predictions against actuals with visualization

**💻 Phase 8: Deployment (Streamlit / Flask)**

**✅ Tasks:**

* Build a simple app:
  + Select stock
  + View trend prediction
  + See indicators & sentiment visuals
* Deploy on:
  + **Streamlit Cloud** (easy)
  + Or **Render / Heroku** (interview-ready)
* Write clear deployment steps in README.md

**🐞 Phase 9: Bug Tracking & Feedback**

**✅ Tasks:**

* Use **GitHub Issues** to log bugs, test failures, or improvements
* Add a CHANGELOG.md to track progress
* Create a feedback form (Google Form) in app if deployed

**📢 Phase 10: Stakeholder Communication**

**✅ Outputs:**

* **Presentation Deck (PDF or PPT)** with:
  + Problem, Process, Insights, Results
  + Visuals + Business Recommendations
* **README.md** with:
  + Clean project summary
  + Installation & usage
  + Screenshots / gif of app

**💬 How to Explain in Interviews:**

“In this project, I followed a full DS lifecycle: from business problem to data collection, modeling, testing, and finally deploying a working demo app. I used industry tools like GitHub for version control, sentiment modeling with VADER, and visualized trends using Streamlit. I also simulated stakeholder interaction by preparing a results report and tracking improvements via GitHub issues.”

Start

**✅ Phase 1: Business Understanding & Problem Framing**

**🧾 Client Brief:**

**Client Need**:  
*"We want a system that can predict the trend or price of a stock for the next 7 days. It should be reliable enough to guide short-term trading or investment decisions."*

**Expectations**:

* Predict whether the stock will **go up or down** for each of the next 7 days
* Or even provide an **estimated closing price**
* Understand **what factors are influencing the movement**
* Be able to **test it on multiple stocks**
* Bonus: Show predictions on a dashboard/app

**Asked Client for Inputs**

“To begin modeling, we’ll need the following inputs from you to ensure accuracy and domain alignment”:

✅ Confirm the stock(s) to be used (e.g., Tesla, Reliance, Nifty 50)  
✅ Historical data period (e.g., past 2 years daily data)  
✅ News data or sentiment source preference (can be auto-collected or provided)  
✅ Clarify if you need:

* Buy/Sell/Hold signal
* Price range forecasts
* Confidence score with each prediction  
  ✅ Preferred format of output (CSV? app? report?)

**✅ Client Requirements – Finalized**

| **Client Need** | **Details ✅** |
| --- | --- |
| 📈 Stocks | Any stock to start (we’ll pick a liquid, popular one — e.g., Tesla or Reliance) |
| 📅 Historical Data | 1 year of daily data |
| 💡 Sentiment Analysis | Not needed in v1 — can be added in a future version |
| 🧠 Outputs Needed | Buy/Sell/Hold signal, predicted price range, confidence score |
| 🌐 Format | Simple and clean website/app to view predictions |

**🔁 How We Translate This into a Data Science Workflow**

| **Area** | **Plan** |
| --- | --- |
| ✅ Problem Framing | Trend + price prediction for next 7 days using historical data |
| 📦 ML Task Types | Classification (Buy/Sell/Hold), Regression (price range) |
| 📈 Data Frequency | Daily OHLCV data for 1 year |
| 📊 Output | - Signal (Buy/Sell/Hold)  - Predicted Price Range  - Confidence score |
| 💻 Deployment | Streamlit web app hosted online |
| 📁 Deliverables | Clean GitHub repo + deployed app + professional documentation |