Assessment Document: Multivariate Time-Series Prediction Using Deep Learning

Subject:

Predicting Financial Market Movements Using the NASDAQ Historical Daily Prices Dataset

Duration: 7 Days

Objective:

The goal of this assessment is to evaluate your ability to preprocess, analyze, and model multivariate time-series data using deep learning techniques. You will predict financial market movements, such as future stock prices, using historical data from the NASDAQ stock exchange. This project will test your skills in handling large datasets, performing time-series analysis, and developing robust machine learning models tailored for financial predictions.

Overview:

In this assessment, you are tasked with predicting stock prices or other financial indicators using the **NASDAQ Historical Daily Prices Dataset**. This dataset contains daily trading information for all companies currently listed on the NASDAQ stock exchange. The data was collected from **Yahoo Finance** using the `yfinance` Python package, which is commonly used for retrieving financial data.

Your job is to:

1. **Preprocess** the dataset to handle missing values, outliers, and any inconsistencies.
2. **Engineer meaningful features** that can improve prediction accuracy, such as moving averages or volatility indicators.
3. **Design and train a deep learning model** to forecast future stock prices.
4. **Evaluate** the model's performance and provide insights on how to improve the prediction accuracy.

This project simulates real-world financial forecasting tasks, where understanding the data and applying advanced machine learning techniques are crucial for success.

Dataset Description:

- **Dataset Name:** NASDAQ Historical Prices
- **Source:** Yahoo Finance via `yfinance` Python package
- **Data Coverage:** Up to April 1, 2020
- **Size:** Several hundred thousand records (covering multiple stocks)
- **Frequency:** Daily intervals

Key Features in the Dataset:

1. **Date** – The trading date in the format YYYY-MM-DD (e.g., 2020-03-31).
2. **Open** – The price at which the stock started trading at the beginning of the day.
3. **High** – The highest price the stock reached during the trading day.
4. **Low** – The lowest price the stock fell to during the trading day.
5. **Close** – The final price of the stock at the end of the trading day. **(This will be your target variable for prediction.)**
6. **Adj Close** – The closing price adjusted for stock splits and dividend payments.
7. **Volume** – The total number of shares traded on that particular day.

Dataset Description

You will use the NASDAQ Historical Daily Prices Dataset.

Download the dataset from: NASDAQ Historical Prices

Technical Requirements and Tools

Programming Languages and Libraries:

- **Language:** Python (version 3.7 or higher)
- **Essential Libraries:**
    - **Data Manipulation:** Pandas, NumPy
    - **Visualization:** Matplotlib, Seaborn
    - **Machine Learning:** Scikit-learn
    - **Deep Learning:** PyTorch

Environment Setup:

- **Preferred Environment:** Jupyter Notebook / Google Colab
- **Configuration:**
    - Ensure compatibility with the latest versions of the required libraries.
    - Use virtual environments or containerization (e.g., venv, conda, Docker) to manage dependencies.

Assessment Tasks

1. **Data Understanding and Preprocessing**

   Objective: Analyze and prepare the dataset for modeling.

   Subtasks:

    - **Exploratory Data Analysis (EDA):**

      Visualize trends, seasonal patterns, and correlations using line charts, heatmaps, and box plots. Identify patterns related to market movements over time.
    - **Handling Missing Values:**

      Detect missing values in the dataset. Apply appropriate imputation methods (e.g., forward fill, interpolation) or remove records/features with excessive missing data. Document and justify your strategy.

    - **Outlier Detection and Treatment:**

      Identify outliers using statistical methods (e.g., Z-score, IQR) or visualization

techniques (e.g., box plots). Decide whether to remove or adjust outliers based on their impact. Provide rationale.

- **Data Scaling and Normalization:**

  Apply scaling techniques such as Min-Max Scaling or Standardization. Justify the choice of scaling method based on data distribution and model requirements.

- **Data Splitting:**

  Split the dataset into training and testing sets (e.g., 80% train, 20% test). Ensure temporal consistency to prevent data leakage (e.g., use the first 80% of the time series for training and the remaining 20% for testing).

2. **Feature Engineering**

Objective: Create and select features that enhance model performance.

Subtasks:

- **Time-Based Features:**

  Extract features such as day of the week, month, and quarter. Create indicators for trading days vs. non-trading days (holidays).

- **Rolling Averages and Moving Windows:**

  Compute rolling averages (e.g., 5-day, 10-day windows) to smooth trends. Incorporate moving window statistics as additional features.

- **Lagged Features:**

  Create lagged features using past prices and indicators (e.g., close price 1 day ago, 5 days ago). Determine optimal lag periods based on autocorrelation analysis.

- **Interaction Features:**

  Generate interaction terms between relevant features (e.g., volume and price volatility).

- **Domain-Specific Features:**

  Integrate external factors such as sector performance and market indices.

- **Feature Selection:**

  Utilize methods like Recursive Feature Elimination (RFE), feature importance from tree-based models, or correlation analysis to select the most predictive features. Provide justification for the selected features.

3. **Model Development**

Objective: Design and implement a deep learning model to predict financial market movements.

Subtasks:

- **Baseline Models:**

  Develop and evaluate simple models such as Linear Regression or Random Forest to establish performance benchmarks. Use these baselines to compare the effectiveness of deep learning models.

- **Deep Learning Model Design:**

  Choose appropriate architectures (e.g., LSTM, GRU, CNN-LSTM hybrids). Design the network architecture, specifying the number of layers, neurons, activation functions, etc. Experiment with different architectures to identify the best-performing model.

- **Activation Functions and Optimizers:**

  Select suitable activation functions (e.g., ReLU, tanh) for hidden layers. Choose optimizers (e.g., Adam, RMSprop) and justify your selection.

- **Loss Functions:**

  Use appropriate loss functions for regression tasks (e.g., Mean Squared Error).

- **Model Complexity and Regularization:**

  Implement techniques to manage model complexity and prevent overfitting (e.g., dropout layers, regularization terms).

4. **Model Training and Evaluation**

Objective: Train the model and assess its performance.

Subtasks:

- **Training:**

  Train the model using the training dataset. Monitor training and validation loss to assess learning progress.

- **Evaluation Metrics:**
  - **Mean Squared Error (MSE)**
  - **Mean Absolute Error (MAE)**
  - **Root Mean Squared Error (RMSE**
- **Performance Assessment:**

  Evaluate the model on the test set using the selected metrics. Compare the performance against baseline models.

- **Visualization:**

  Plot predicted vs. actual values. Include residual plots to analyze prediction errors. Plot the evaluation metrics if applicable.

5. **Model Optimization**

Objective: Enhance model performance through optimization techniques.

Subtasks:

- **Hyperparameter Tuning:**

  Optimize hyperparameters such as learning rate, batch size, number of epochs, number of layers, and neurons per layer. Use techniques like Grid Search, Random Search, or Bayesian Optimization.

- **Regularization Techniques:**

  Apply dropout layers to prevent overfitting. Experiment with different dropout rates.

- **Early Stopping:**

  Implement early stopping based on validation loss to halt training when performance ceases to improve.

- **Model Evaluation Post-Optimization:**

  Re-evaluate the optimized model on the test set. Compare performance metrics before and after optimization.

6. **Documentation**

Objective: Provide a comprehensive and professional report detailing your approach and findings.

Subtasks:

- **Report Format:** Submit as a PDF or Jupyter Notebook.
- **Report Sections:**
  - Title Page
  - Table of Contents
  - Introduction: Overview of the problem and objectives
  - Data Insights: Findings from EDA with key visualizations
  - Preprocessing: Handling of missing values, outliers, and scaling
  - Feature Engineering: Explanation and justification of features
  - Model Design: Architecture of the deep learning model(s)
  - Results: Evaluation metrics and comparison with baseline models
  - Model Optimization: Techniques applied and performance improvements
  - Challenges and Solutions
  - Conclusion: Key findings and future work
  - References
- **Code Documentation:**

  Ensure all code is well-commented and organized. Provide a README.md in the GitHub repository with instructions on how to run the code, dependencies, and setup steps.

Submission Requirements

1. **Code Submission**

   Platform: GitHub Repository (public or private)

   Repository Structure (Example):

```
Unset
├── data/
│   ├── raw/
│   └── processed/
├── notebooks/
│   └── EDA.ipynb
├── src/
│   ├── data_preprocessing.py
│   ├── feature_engineering.py
│   ├── model.py
│   └── train.py
├── models/
```

```
|   └── trained_model.h5
├── reports/
|   └── report.pdf
├── requirements.txt
└── README.md
```

2. **Code Quality:**
    - Well-documented and modular code
    - Executable and reproducible results
    - Meaningful variable and function names
    - Inclusion of docstrings for functions and classes
3. **Report Submission**

    Format: PDF or Jupyter Notebook

    Content: As detailed in the Documentation section above

    Inclusions: All relevant plots, tables, and visualizations embedded within the report

4. **Additional Files**

    - **requirements.txt:** List all dependencies and their versions
    - **README.md:** Overview of the project, setup instructions, and how to run the code

5. **Deadline:** Submission within 7 days of receiving this assessment

Suggested Timeline

- **Day 1-2:** Data Understanding and Preprocessing
- **Day 3:** Feature Engineering
- **Day 4-5:** Model Development
- **Day 6:** Model Training, Evaluation, and Optimization
- **Day 7:** Documentation and Final Review

Evaluation Criteria

1. **Technical Proficiency (60%)**
    - Data Preprocessing and Cleaning (15%)
    - Feature Engineering (15%)
    - Model Design and Implementation (20%)
    - Model Training and Evaluation (10%)
2. **Logical and Analytical Thinking (20%)**
    - Innovation in Feature Engineering (10%)
    - Justification of Modeling Choices (10%)
3. **Performance (10%)**
    - Achievement of Low MAE and RMSE (10%)
4. **Documentation (10%)**
    - Clarity and Completeness of Report (5%)
    - Code Documentation and Organization (5%)

Conclusion

This assessment is designed to comprehensively evaluate your skills in handling multivariate time-series data, feature engineering, deep learning model development, and overall analytical thinking. By following the outlined tasks and adhering to the submission guidelines, you will demonstrate your proficiency in building predictive models for financial market analysis using real-world historical data from NASDAQ.