Assessment Document: Multivariate Time-Series Prediction Using Deep Learning

Subject: Appliance Energy Prediction Using the Energy Dataset

Duration: 7 Days

Objective: Evaluate the candidate's ability to preprocess, analyze, and model multivariate time-series data using deep learning techniques to predict energy consumption.

Overview

In this assessment, you are tasked with predicting **energy consumption (`Appliances`)** using the multivariate Appliance Energy Prediction Dataset. The dataset contains environmental, time-based, and energy-related features recorded at 10-minute intervals over an extended period.

Your goal is to preprocess the dataset, engineer meaningful features, and design a deep learning model to predict future energy consumption.

Dataset Description

You will use the **Appliance Energy Prediction Dataset**.
Download the dataset from the Share Energy Dataset.

Accessing the Dataset

- **Download Link:** Appliance Energy Prediction Dataset

Dataset Details

- **Size:** Approximately 20,000 records
- **Interval:** 10-minute intervals
- **Duration Covered:** Several months

Key Features:

- **Date:** Timestamp of each observation (YYYY-MM-DD HH:MM:SS).
- **Appliances:** Energy consumption in Wh (target variable).
- **Lights:** Energy consumption of lights in Wh.
- **T1-T6:** Indoor temperature readings from different areas of the building (in Celsius).
- **RH_1–RH_6:** Indoor humidity readings corresponding to temperature sensors.
- **T_out:** Outdoor temperature in Celsius.
- **RH_out:** Outdoor humidity percentage.
- **Windspeed:** Outdoor wind speed (m/s).
- **Visibility:** Outdoor visibility (km).
- **Press_mm_hg:** Atmospheric pressure outside the building (mmHg).
- **NSM:** Numerical step counter starting at midnight (seconds elapsed since 00:00:00).
- **WeekStatus:** Indicates whether a day is a Weekday or Weekend.
- **Day_of_week:** Day of the week as a number (e.g., 0 = Sunday, 6 = Saturday).

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: `TensorFlow` or `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:

1. **Exploratory Data Analysis (EDA):**
    - Visualize trends, seasonal patterns, and correlations using plots such as line charts, heatmaps, and box plots.
    - Identify patterns related to energy consumption over time.
2. **Handling Missing Values:**
    - Detect missing values in the dataset.
    - Apply appropriate imputation methods (e.g., mean/mode imputation, interpolation) or remove records/features with excessive missing data.
    - Document the strategy used and justify your choice.
3. **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 impute outliers based on their impact.
    - Provide rationale for the chosen approach.
4. **Data Scaling and Normalization:**
    - Apply scaling techniques such as Min-Max Scaling or Standardization.
    - Justify the choice of scaling method based on the data distribution and model requirements.
5. **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:

1. **Time-Based Features:**
   - Extract features such as hour of the day, day of the week, month, etc.
   - Create indicators for Weekday vs. Weekend.
2. **Rolling Averages and Moving Windows:**
   - Compute rolling averages (e.g., 1-hour, 3-hour windows) to smooth trends.
   - Incorporate moving window statistics as additional features.
3. **Lagged Features:**
   - Create lagged features using past energy consumption values (e.g., consumption 10 minutes ago, 30 minutes ago).
   - Determine optimal lag periods based on autocorrelation analysis.
4. **Interaction Features:**
   - Generate interaction terms between relevant features (e.g., temperature and humidity).
5. **Domain-Specific Features:**
   - Integrate external factors if applicable (e.g., special events, holidays).
6. **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 energy consumption.

Subtasks:

1. **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.
2. **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.
3. **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.
4. **Loss Functions:**
    - Use appropriate loss functions for regression tasks (e.g., Mean Squared Error).
5. **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:

1. **Training:**
    - Train the model using the training dataset.
    - Monitor training and validation loss to assess learning progress.
2. **Evaluation Metrics:**
    - **Mean Absolute Error (MAE):** The average of the absolute differences between predicted and actual values.
    - **Root Mean Squared Error (RMSE):** The square root of the average of squared differences between predicted and actual values.
    - Optionally, include **Mean Absolute Percentage Error (MAPE)** and **R-squared** for comprehensive evaluation.
3. **Performance Assessment:**
    - Evaluate the model on the test set using the selected metrics.
    - Compare the performance against baseline models.
4. **Visualization:**
    - Plot predicted vs. actual energy consumption values.
    - Include residual plots to analyze prediction errors.
    - Plot the evaluation metrics also if applicable.

5. Model Optimization

Objective: Enhance model performance through optimization techniques.

Subtasks:

1. **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.
2. **Regularization Techniques:**
    - Apply dropout layers to prevent overfitting.
    - Experiment with different dropout rates.
3. **Early Stopping:**
    - Implement early stopping based on validation loss to halt training when performance ceases to improve.
4. **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:

1. **Report Format:**
    - Submit as a PDF or Jupyter Notebook.
2. **Report Sections:**
    - **Title Page:** Include the assessment title, your name, and date.
    - **Table of Contents:** For easy navigation.
    - **Introduction:** Brief overview of the problem and objectives.
    - **Data Insights:**
        - Summarize findings from exploratory data analysis.
        - Include key visualizations and observations.
    - **Preprocessing:**
        - Detail the steps taken to clean and prepare the data.
        - Discuss handling of missing values, outliers, and scaling.
    - **Feature Engineering:**
        - Explain the features created and selected.
        - Justify the relevance of each engineered feature.
    - **Model Design:**
        - Describe the architecture of the deep learning model(s).
        - Justify choices regarding layers, activation functions, optimizers, and loss functions.
    - **Results:**
        - Present evaluation metrics and compare them against baseline models.
        - Include plots such as predicted vs. actual values.
    - **Model Optimization:**
        - Describe the optimization techniques applied.
        - Discuss improvements in performance metrics.
    - **Challenges and Solutions:**
        - Highlight any difficulties encountered and how they were addressed.
    - **Conclusion:**
        - Summarize key findings and potential areas for future work.
    - **References:** Cite any external resources or literature used.
3. **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):

```
├── 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
```

- **`If incase of individual .py file, you can also perform all the tasks in .ipynb files. Ensure that the code is modular and readable.`**

- **Code Quality:**
  - Well-documented and modular code.
  - Should be executable.
  - Use of meaningful variable and function names.
  - Inclusion of docstrings for functions and classes.

2. Report Submission

- **Format:** PDF or Jupyter Notebook
- **Content:** As detailed in the Documentation section above.
- **Inclusions:** All relevant plots, tables, and visualizations should be embedded within the report.

3. Additional Files

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

4. Deadline

- **Submission Window:** Within 7 days of receiving this assessment.

Suggested Timeline

To effectively manage your time over the 7-day period, consider the following milestones:

- **Day 1-2:** Data Understanding and Preprocessing
  - Conduct EDA, handle missing values, detect and treat outliers, scale data, and split datasets.
- **Day 3:** Feature Engineering
  - Create time-based, rolling, lagged, interaction, and domain-specific features. Perform feature selection.
- **Day 4-5:** Model Development
  - Develop baseline models. Design and implement deep learning models. Experiment with different architectures.
- **Day 6:** Model Training, Evaluation, and Optimization
  - Train models, evaluate performance, apply optimization techniques, and finalize the best model.
- **Day 7:** Documentation and Final Review
  - Compile the report, finalize code documentation, ensure repository structure, and perform a thorough review before submission.

Additional Resources and Support

Reference Materials:

- **Time Series Forecasting:**
  - Time Series Forecasting with LSTM Neural Networks (TensorFlow Tutorial)
- **Feature Engineering for Time Series:**
  - Kaggle - Time Series Feature Engineering
- **Deep Learning Guides:**
  - Deep Learning with Python (Francois Chollet)
  - PyTorch Official Tutorials

Evaluation Criteria

Your submission will be evaluated based on the following criteria:

1. Technical Proficiency (60%)

- **Data Preprocessing and Cleaning (15%):**
  - Effectiveness in handling missing values, outliers, and scaling.
- **Feature Engineering (15%):**
  - Creativity and relevance of engineered features.
  - Appropriate feature selection methods.
- **Model Design and Implementation (20%):**
  - Appropriateness of the chosen deep learning architecture.
  - Quality and efficiency of the code.
- **Model Training and Evaluation (10%):**
  - Correct implementation of training procedures.
  - Accurate computation and interpretation of evaluation metrics.

2. Logical and Analytical Thinking (20%)

- **Innovation in Feature Engineering (10%):**
  - Novelty and effectiveness of the features created.
- **Justification of Modeling Choices (10%):**
  - Clear and logical reasoning behind architecture and hyperparameter selections.

3. Performance (10%)

- **Achievement of Low MAE and RMSE (10%):**
  - Demonstrated ability to build models with high predictive accuracy.

4. Documentation (10%)

- **Clarity and Completeness of Report (5%):**
  - Well-structured and comprehensive report with all required sections.
- **Code Documentation and Organization (5%):**
  - Clean, readable, and well-documented codebase.

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 energy consumption.