Generative AI for Statistical Modeling – Coding Exercises

Exercise 1: Feature Engineering with LLMs

Goal: Use an LLM (e.g., OpenAI API) to generate engineered features for a dataset.

Steps:

1. Load a dataset (e.g., the **NASA C-MAPSS dataset** for predictive maintenance).
2. Extract basic statistical features (mean, std, rolling averages).
3. Use an LLM to generate additional feature ideas based on the dataset description.
4. Implement and integrate the suggested features.
5. Compare model performance before and after adding the AI-generated features.

Hints:

- Use the `openai` package to interact with ChatGPT.
- Ask: *"What additional features could improve predictions for remaining useful life (RUL)?"*
- Validate new features using correlation analysis and feature importance.

Libraries: `pandas`, `openai`, `sklearn`, `tsfresh`

Exercise 2: Synthetic Data Generation for Class Imbalance

Goal: Use a GAN (Generative Adversarial Network) to create synthetic fraud cases for a classification task.

Steps:

1. Load a fraud detection dataset (e.g., `creditcard.csv` from Kaggle).
2. Analyze class imbalance (percentage of fraudulent vs. non-fraudulent transactions).
3. Train a GAN (or use `ctgan`) to generate synthetic fraud transactions.
4. Augment the dataset with synthetic samples.
5. Train a classifier (e.g., RandomForest) on both original and augmented data.
6. Compare model performance before and after synthetic data augmentation.

Hints:

- Use the `ctgan` library for tabular GAN-based synthetic data generation.
- Ensure synthetic data distributions match real data characteristics.
- Evaluate using F1-score and AUC-ROC metrics.

Libraries: `pandas`, `torch`, `ctgan`, `sklearn`

Exercise 3: Time Series Forecasting with GenAI

Goal: Enhance a time series forecasting model using TimeGPT and synthetic time-series data.

Steps:

1. Load a time-series dataset (e.g., **energy demand forecasting dataset** from `UCI ML Repository`).
2. Train a baseline ARIMA or LSTM model for forecasting.
3. Use `TimeGPT` to generate improved forecasts.
4. Apply a **diffusion model** to generate synthetic time series scenarios.
5. Compare forecast accuracy (RMSE, MAE) before and after enhancements.

Hints:

- Use `prophet` for quick time series modeling.
- Experiment with fine-tuning `TimeGPT` for better results.
- Synthetic scenarios help test model robustness to rare events.

Libraries: `pandas`, `prophet`, `nixtla/TimeGPT`, `sklearn`

Exercise 4: Explainability & Counterfactual Analysis with LLMs

Goal: Use SHAP and LLM-generated explanations to interpret model decisions.

Steps:

1. Train a classifier on a tabular dataset (e.g., **loan approval dataset**).
2. Use SHAP to analyze feature importance.
3. Generate **natural language explanations** of model decisions using an LLM.
4. Create counterfactual examples: *"What if the applicant's income was higher?"*
5. Compare how the model responds to counterfactual changes.

Hints:

- Use `shap.Explainer()` for local & global model interpretability.
- LLMs can help generate understandable business-level explanations.
- Counterfactuals provide insights into model biases.

Libraries: `shap`, `openai`, `sklearn`, `lime`

Bonus Challenge: Hybrid Approach

Combine **feature engineering (LLMs) + synthetic data (GANs) + forecasting (TimeGPT) + interpretability (SHAP/LLMs)** into a single **end-to-end GenAI-enhanced modeling pipeline**. Try applying it to **your own dataset**!

How to Use These Exercises?

Beginners – Start with feature engineering and interpretability.

Intermediate learners – Try synthetic data and forecasting improvements.

Advanced users – Build a hybrid GenAI-powered modeling pipeline!

Keep experimenting & refining your workflows!