CLARITY AI

Project Kickoff: Time Series Forecasting for Predicting Sustainable Performance of Companies

Abstract

Clarity AI aims to enhance the predictive accuracy of its estimation models for sustainability indicators, such as greenhouse gas emissions, by incorporating time series forecasting techniques. Currently, gradient-boosting methods serve as the primary approach for estimating sustainability metrics. However, these methods may not fully capture the dynamic nature of these indicators over time. This project aims to delve into and apply different time series forecasting techniques, such as autoregressive models, seasonal decomposition, and exponential smoothing. The goal is to enhance the precision of our predictions, enabling us to forecast sustainability indicators by analyzing historical trends and patterns. By incorporating time series forecasting into our estimation framework, we aim to provide investors and stakeholders with more accurate and actionable insights into the future sustainability performance of companies.

The students involved will engage in tasks such as conducting exploratory data analysis, gaining insights from historical data, selecting appropriate forecasting models, fine-tuning hyperparameters, and evaluating model performance. The anticipated outcomes include robust insights into the effectiveness of these forecasting methods, comparative performance metrics of the different techniques explored, and a clear understanding of the tradeoffs with respect to the current, non-forecasting methods.

Research project

The project aims to address the following research questions:

1. How do different forecasting models perform in capturing the dynamic nature of sustainability metrics over time?
2. What are the key drivers and factors influencing the variability of sustainability indicators over time, and how can they be effectively captured by forecasting models?

In addition to addressing the main research questions, it would be ideal to gain insights such as:

1. Do organizations with explicit commitments towards achieving global net zero emissions, or targets to reduce CO2 emissions, differ in their forecasted sustainability trajectories with respect to those without such commitments and targets?

2. What are the challenges associated with relying on forecasting techniques to predict the future sustainability performance of companies? How would the model perform when limited historical data is available for a given company?
    a. What is the minimal historical depth required to apply any meaningful forecasting technique?
3. How do you anticipate that the integration of forecasting techniques could contribute to providing investors and stakeholders with actionable insights regarding the future sustainability performance of companies? Is there any literature regarding this relationship?

The students will be involved in the following steps of a typical data science project lifecycle:

1. **Exploratory Data Analysis (EDA):** Summarizing the main characteristics of the dataset to understand its underlying structure, patterns, and relationships.
2. **Data Preprocessing:** Cleaning, imputing, and transforming historical sustainability data to ensure data quality and adequacy with forecasting models.
3. **Model Implementation:** Developing and comparing various time series forecasting algorithms, including autoregressive models, seasonal decomposition, and exponential smoothing.
4. **Hyperparameter Optimization:** Fine-tuning model parameters using techniques such as grid search to enhance model's performance.
5. **Performance Evaluation:** Assessing forecasting models based on metrics such as Mean Absolute Error (MAE) and Mean Squared Error (MSE).

Expected Outcome

This hands-on project represents a unique opportunity for students to apply their technical skills to a real-world challenge at the intersection of technology, sustainability, and finance.

In practical terms, each team will be asked to:

1. Prepare a **mid-project presentation** to share progress and gather feedback.
2. Write a **technical report** on the project findings, as well as the methods used to achieve them. The report should not exceed 20 pages including figures, tables, and bibliography.
3. Provide a **code base** following the requirements specified in the section Code Guidelines.
4. "Kaggle"-like competition: In order to foster some healthy competition among the teams, each team will provide a **CSV file with a validation dataset**. We will be comparing the performance of the models on this dataset among the teams.

Expected timelines

Below is a general guideline of the project timelines:

- [May 14th - May 28th]: Kickoff, EDA and literature review
- [May 28th - June 11th]: First modeling "sprint".
- [June 11th - June 18th]: Preparation of a 20-minute mid-project presentation.
- [June 18th - July 2nd]: Second modeling "sprint".
- [July 2nd - July 14th]: Report write-up and code base delivery.

Dataset

Participants will be provided with a comprehensive dataset comprising historical sustainability data, including greenhouse gas emissions, energy consumption, and company characteristics. The dataset will serve as the foundation for analysis, enabling participants to explore the efficacy of time series forecasting techniques in predicting sustainability performance.

The dataset will include the following information:

- Primary key:
  - clarity_id: Clarity unique identifier for organizations
  - metric_year: The year that the datapoint is applicable for.
- Company fundamentals:
  - clarity_industry_code: Clarity code of the industry of the company.
  - clarity_industry_name: Name of the Clarity AI industry.
  - country_code: ISO 3166-1 alpha 2 code of the country of the company's headquarters
  - revenue: Total amount of revenue gained by the company in the given year (million USD)
- Environmental performance:
  - co2directscope1_raw: Scope 1 emissions that occur within a company's organizational boundary from sources that the company owns or controls in tonnes of CO2e (tons)
  - co2directscope1_intensity: Scope 1 emissions that occur within a company's organizational boundary from sources that the company owns or controls in tonnes of CO2e, divided by the company's revenue (tons/revenue)
  - targets_emissions: Has the company set targets or objectives to be achieved on emission reduction?
  - verification_co2directscope1: Have the Scope 1 GHG emissions of a company been verified by a third party?

- sbti_alignment: Has the company made a commitment to the SBT Initiative?
- nz_statement: Indicates if the organization has any kind of target, goal, or pledge towards achieving global net zero emissions
- policy_emissions: Does the company have a policy to improve emission reduction?

Please find a detailed description of the dataset here.

The dataset is split in two separate files: a training dataset and a validation dataset. The training dataset shall be used to run the exploratory analysis and the modeling necessary to build a forecasting model. The validation dataset will be used to evaluate the performance of the proposed model, and contains last year's data for about 300 companies that are present in the training dataset. This dataset does not include information about the CO2 emissions of the company so that it can be used as an objective measure of the proposed model's performance.

The datasets can be located in the corresponding slack channel: training file and validation file

Code Guidelines

The following are **requirements** for the delivery of the code base:

1. The programming language is Python.
2. Use of any external library is allowed.
    a. External dependencies must be specified with an exact version number following Python standards.
    b. Either one of pip (requirements.txt), Pipenv, or Poetry, can be used as dependency management tools.
3. Analysis code can be implemented using notebooks, python scripts, or a combination of both.
4. All of the analysis must be reproducible from a new environment. The codebase must include instructions on how to set up the project and run the analysis that yields the same results included in the technical report.
    a. All random seeds must be fixed to ensure the reproducibility of random variables.
    b. Plots can be created outside the codebase using other tools. In that case, instructions on how to regenerate the source data for the plot and the plot must be included too.

The following are **recommendations and best practices** to help organize and work effectively with the code base:

1. Leverage standard data processing and scientific analysis libraries such as Numpy, Pandas, ScyPy, and Scikit-Learn.

2. Tools for virtual environment and dependency management are encouraged over plain pip + requirement.txt, with Poetry being the preferred one.
3. Separate project tasks (i.e. feature building, model training...) in different notebooks/scripts.
4. Separate the common utility functions from runnable code. A simplified example:

```
Unset
src
├── lib         # Library with utilities to import. No scripts.
│   ├── my_utils.py
│   └── ....py
└── jobs        # Python scripts/notebooks to run analysis (might import lib/)
    ├── data_processing
    │   ├── 01-EDA.ipynb
    │   └── build_features.py
    ├── features
    │   └── build_features.py
    └── model
        ├── train_model.py
        └── predict_model.py
```

5. Use Python Docstrings to document your code. Especially your utility functions.
6. Following a project template such as [cookiecutter-data-science] is highly recommended.
7. Avoid using global variables inside functions and classes that can cause side effects. Instead, parametrize your functions, classes, and/or methods if necessary. A dummy example:

```python
# ❌WRONG
SRC_FILE = "data.csv"
def read_training_data():
    return pd.read_csv(SRC_FILE)
# ✅GOOD
def read_training_data(src_file: str):
    return pd.read_csv(src_file)
```

8. Parametrize the project configuration, such as model hyperparameters or read/write files. This can be done using configuration files (JSON or YAML), command line arguments, or environment variables. Configuration files are recommended as you can version your modeling approaches in different config files. For example: *src/jobs/model/train-config-v0.json* will be your initial model configuration, while *src/jobs/model/train-config-v1-tunned.json* will be the model configuration after hyperparameter tuning.