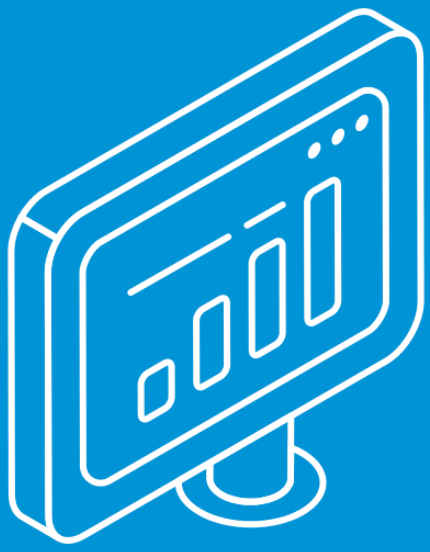
{EPITECH}

TARDIS

PREDICTING THE UNPREDICTABLE



TARDIS

Preliminaries

files to deliver: tardis_eda.ipynb, tardis_model.ipynb, tardis_dashboard.py,
language: Python with pandas, numpy, matplotlib, seaborn, scikit-learn, streamlit
coding style: Your project must be formatted with ruff formatter

You are part of a newly formed **SNCF Data Analysis Service**, dedicated to improving the efficiency of train travel across the country. Your mission? Analyze historical train delay data, uncover hidden patterns, and develop a predictive model that can forecast delays before they happen.

The SNCF has entrusted your team with making the railway system more efficient and transparent. If you succeed, your dashboard will be used by thousands of travelers to better plan their journeys. If you fail… well, expect a lot more unhappy commuters. No pressure!

Using the provided dataset (available along with the subject on the intranet), your job is to clean and analyze historical delay data, develop a simple predictive model, and present your insights through an interactive Streamlit dashboard.

Objectives

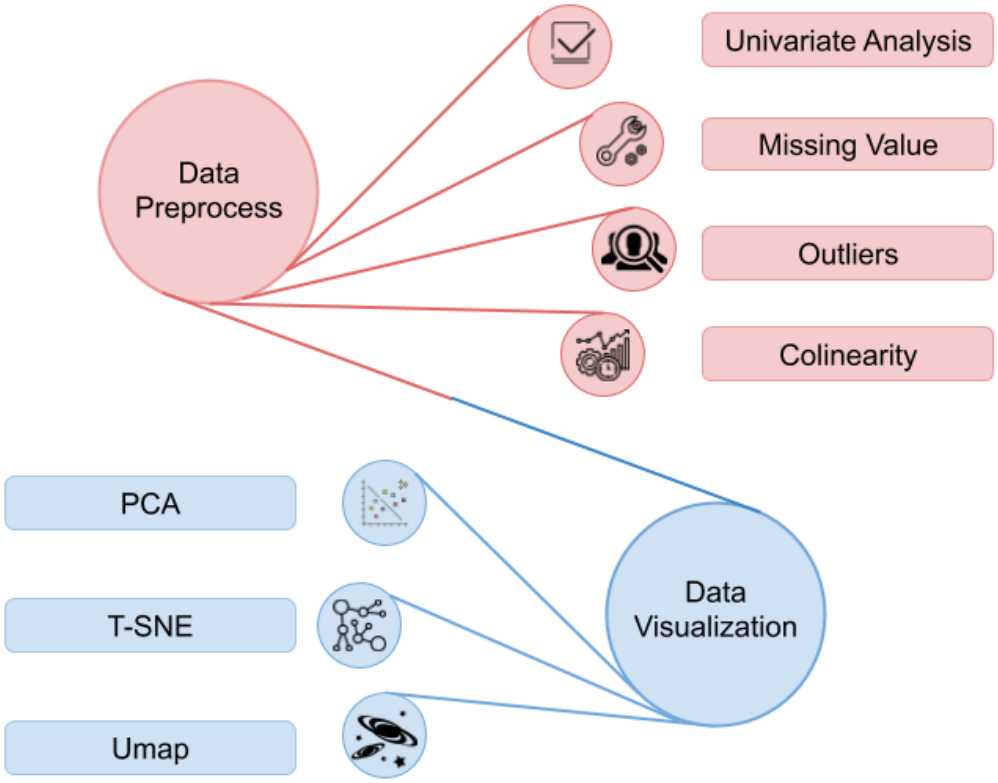
- ✓ Data Cleaning & Preprocessing – Handle missing values, inconsistencies, and prepare the dataset for analysis.
- ✓ Exploratory Data Analysis (EDA) – Generate insightful visualizations to understand trends and correlations.
- ✓ Predictive Modeling – Implement a basic machine learning model to predict train delays.
- ✓ Dashboard Development – Create an interactive web app using Streamlit to display insights and allow user interaction.

Step 1: Data Exploration & Cleaning

Before diving into analysis, ensure the dataset is clean and structured. Start by inspecting its structure and identifying any missing values or duplicate entries. Convert data types where necessary and create additional features, such as day of the week or peak hours, to enhance predictions.

Tasks:

- ✓ Load and inspect the dataset.
- ✓ Handle missing values and remove duplicate entries.
- ✓ Convert columns to appropriate data types.
- ✓ Perform feature engineering to create new useful variables.



EPITECH

Step 2: Data Visualization & Analysis

Once the dataset is clean, use visualizations to understand trends and patterns in train delays. This will help you uncover insights that may improve predictive modeling

Tasks:

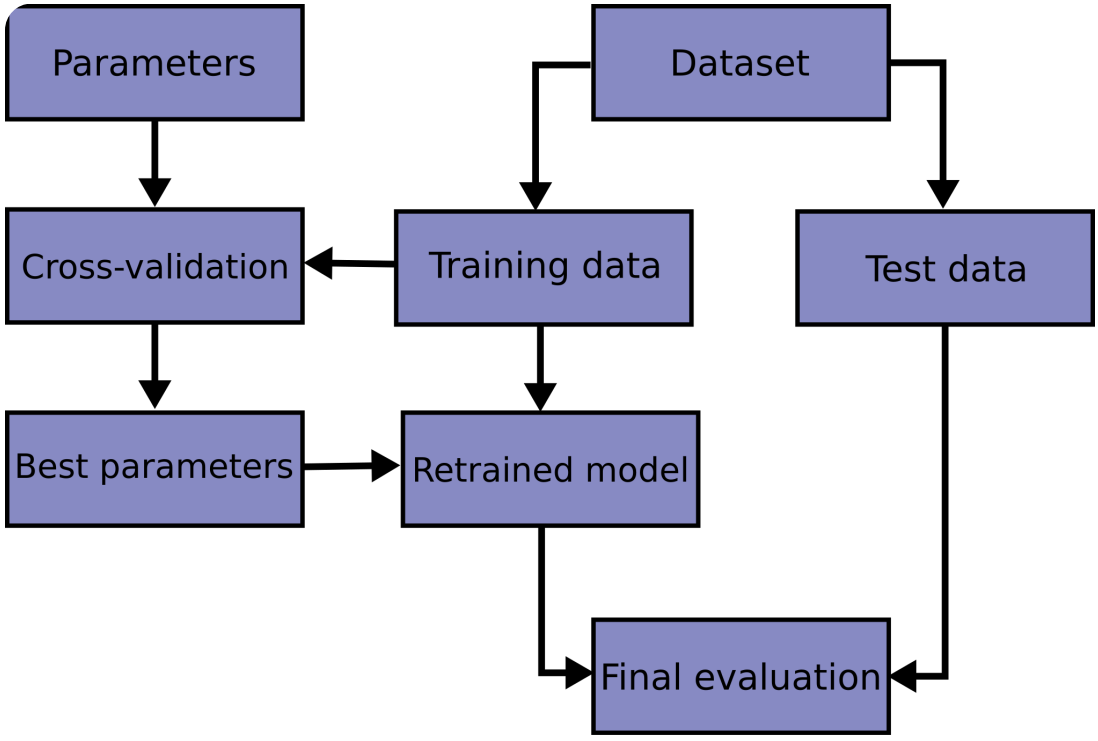
- ✓ Generate summary statistics to understand the dataset better.
- ✓ Plot delay distributions and identify common delay durations.
- ✓ Compare delays across different stations and times of day.
- ✓ Use heatmaps to explore correlations between different variables.

Step 3: Building a Prediction Model

Now it's time to use historical data to predict future delays. Select relevant features, experiment with different models, and evaluate their performance.

Tasks:

- ✓ Select relevant features for training (e.g., departure station, time, past delays).
- ✓ Train a simple machine learning model using scikit-learn (e.g., Linear Regression, Decision Tree, Random Forest).
- ✓ Evaluate performance using metrics like RMSE, R^2, or accuracy.
- ✓ Tune hyperparameters to improve model performance.
- ✓ Compare different models and justify the selection of the best one.



Step 4: Developing the Streamlit Dashboard

Make your analysis accessible with an interactive dashboard. The dashboard should allow users to explore key insights and interact with the predictive model.

Tasks:

- ✓ Delay distribution charts (histograms, boxplots)
- ✓ Station-level analysis (delays per station, cancellation rates)
- ✓ Heatmaps to show correlations between delay factors (external causes, infrastructure, traffic management)
- ✓ Interactive elements allowing users to select routes, stations, or dates
- ✓ Summary statistics like average delays, cancellation counts, and punctuality rates
- ✓ Integrate the trained model for real-time predictions.
- ✓ Deploy the Streamlit application and provide usage documentation.

Which dashboard components will best help users understand train delays? Ensure the dashboard is intuitive and accessible to non-technical users.

Deliverables

- ✓ A `requirements.txt` file listing all the dependencies for the project.
- ✓ A `tardis_eda.ipynb` Jupyter Notebook containing data analysis and cleaning, it should take a `dataset.csv` file as input and output a `cleaned_dataset.csv` file.
- ✓ A `tardis_model.ipynb` Jupyter Notebook containing model training.
- ✓ A `tardis_dashboard.py` Python script containing a functional Streamlit dashboard.
- ✓ A well-documented README.md explaining the approach, installation steps, and usage.

Bonus Challenges:

- ✓ Implement a feature selection technique to optimize the model.
- ✓ Add real-time data updates from SNCF open data.
- ✓ Use advanced visualization techniques like animated charts or geospatial maps.
- ✓ Experiment with deep learning models for improved accuracy.
- ✓ Include an explanatory model component to provide reasons for predictions.

v 1.1