Project: Interactive Data Analysis with Streamlit

Python - M2

2025

Objective

Build a **multi-page Streamlit application** with the following functionalities:

1. Exploration of Titanic dataset (Part 1)
2. Scatter matrices and ellipses on Gaussian data (Part 2)
3. Ellipses on two selected numeric columns of Titanic dataset (Part 3)

Part 0: Set up a Github repository and a Python environment

- **Create a Repository:**
  - Log into your GitHub account and navigate to the repositories section.
  - Click on `New` to create a new repository.
  - Name the repository.
  - Set the visibility to `Private` if the project should remain confidential; otherwise, use `Public`.
  - Initialize the repository with a `README.md` file to document the project description and instructions.
  - Add a `.gitignore` file to ignore unnecessary files.
- **Create a Python Environment:**
  - Create a new conda environment for the project.
  - Activate the new environment.
  - Install the libraries you think you will need (e.g., `numpy`, `pandas`, ...).
  - Export the environment configuration to a file:

```
conda env export > environment.yml
```

- **Upload `environment.yml` to GitHub:**
  - Commit and push the file to the main branch.

  - This file will allow other collaborators to recreate the same environment using:

    ```
    conda env create -f environment.yml
    ```

- **Protect the Main Branch:**
  - On GitHub, go to the repository settings and navigate to `Branches`.
  - Add a branch protection rule for `main` to ensure that changes can only be made through Pull Requests (PRs).
  - Enable options like `Require pull request reviews before merging` to ensure code quality.
  - Each collaborator should create their own branch for their specific feature or bug fixes. You can put your initials in the branch name.
  - All changes should be merged into `main` only through PRs, after review.

Part 1: Titanic Data Visualization

- Load the Titanic dataset into a DataFrame.
- Display the first rows of the dataset.
- Make relevant data visualizations to present the dataset, using plotly for interactive plots.
- Add interactive widgets (for instance `st.selectbox`) to select the feature.

Part 2: Gaussian Scatter Matrices

- Generate 2D Gaussian data using `numpy`.
- Allow user to adjust mean, variance, and covariance with sliders.
- Check that covariance matrix is positive-semidefinite. If it not, raise a warning with `st.warning`.
- Add optional outliers: allow the user to decide if they want outliers with a checkbox. The user can set the means of the outliers, but the covariance is the same as the previously generated data.
- Fit `EmpiricalCovariance` estimator from `sklearn.covariance`. What are the methods and attributes of `EmpiricalCovariance` ?
- Create a function to generate ellipses.
- Display a scatter plot with an ellipse representing the covariance.
- Show the covariance matrix in Streamlit using `st.dataframe`.

Hints:

- To check that the covariance matrix is positive semidefinite, check that its eigenvalues are real and nonnegative.
- Modularize the code: generate data → fit estimators → create ellipses → plot.
- Put the functions that will be needed in Part 3 in a separate file called `utils`.

Part 3: Ellipses on Titanic Dataset

- Select two numeric columns from the Titanic dataset (optional: let the user select them).
- Drop rows with missing values for the selected columns, and add a warning stating the numbers of deleted rows.
- Fit `EmpiricalCovariance` on these columns.
- Plot ellipse on scatter plot of selected columns.

Hints:

- Reuse code/logic from Part 2 for covariance estimation and ellipse plotting.

Deliverables

- Multi-page Streamlit app:
  - `pages/1_Exploration_Titanic.py`
  - `pages/2_Scatter_Matrices_Gaussian.py`
  - `pages/3_Titanic_Ellipses.py`
- `utils.py`
- Optional: brief README explaining each page

Tips

- Test each page independently
- Use small datasets first to debug