To address the drug discovery problem, we’ll assume a DataFrame named data with columns input1, input2, input3, and output is provided, with the goal of minimizing the adverse reaction (output) to be as close to zero as possible. The tasks are:

* Perform a feature importance analysis using a Random Forest Regressor.
* Drop input2 as specified and refit a Gaussian Process (GP) model using input1 and input3.
* Use Bayesian optimization to find the next best point to query the black-box function, with input2 fixed at 0.5.
* Print:
  + The best input coordinates from the provided dataset (closest to zero, using abs(output)).
  + The best input coordinates after optimization (from the GP model).
  + The proposed next query point and its expected output.
* Replace the heatmap with a contour plot (using plt.contourf) to visualize the surrogate model predictions and Expected Improvement (EI) in the input1–input3 space, explicitly accounting for input2=0.5 in the model, and marking:
  + The best point in the data (yellow star).
  + The best optimized point (cyan triangle).
  + The next query point (white circle).
* Ensure the previous IndexError in expected\_improvement is fixed by handling array shapes correctly.

Since no new data is provided, the code is written to work with any DataFrame data with the required structure. The contour plots will replace the seaborn.heatmap to visualize the predictions and EI with input2=0.5.

***Complete Solution***

The code below implements the pipeline, using plt.contourf for plotting to show levels of the surrogate model predictions and EI, with input2 fixed at 0.5 in the model.

***Key Changes***

* Heatmap to Contour Plot: Replaced seaborn.heatmap with plt.contourf to create contour plots with 20 levels, showing the surrogate model predictions and EI in the input1–input3 space with input2=0.5 fixed (implicitly via the GP model trained on input1 and input3).
* Best in Data: Uses data['output'].abs().idxmin() to find the output closest to zero, correcting the previous error.
* Markers: Yellow star (best in data), cyan triangle (best optimized), white circle (next query point).

***Explanation***

* Error Fix: The expected\_improvement function uses 1D arrays for sigma and ei, computing EI only where sigma > 0.
* Best in Data: Correctly finds the output closest to zero using abs(output).
* Fixed input2=0.5: The GP model is trained on input1 and input3, and all predictions/optimizations assume input2=0.5.
* Contour Plots: Use plt.contourf with 20 levels for smooth visualization, replacing heatmaps while maintaining clarity.
* Modularity: Code works with any data DataFrame with the required structure.

***Notes***

* Data Requirement: Assumes data has columns input1, input2, input3, output (inputs in [0, 1], output negative, aiming for zero).
* Providing Data: Assign data = your\_dataframe before running.

Stability: Ensure sufficient rows (e.g., >10) for stable GP fitting.

***Next Steps***

* Provide Data: Share the DataFrame data to generate specific outputs and plots.
* Query Black-Box: Use the suggested query point to get the actual output.
* Iterate: Provide the new output to update the dataset and suggest the next query point.