The main reasoning, as well as the **Exploratory Data Analysis (EDA)**, is presented in the Jupyter Notebook **test.ipynb**. Based on the analysis, two models were selected for further work in the project: **CatBoost** and **Linear Regression**.

The main code is located in the **/src** folder. The **src/main.py** file allows you to run training and prediction for three predictors:

- `docker-compose build` – builds the Docker image
- `docker-compose up income` – trains the predictor to forecast the **income** of 10,000 users for August
- `docker-compose up outcome` – predicts the **expenses** of 10,000 users for August
- `docker-compose up difference` – predicts the **difference** between income and expenses for 10,000 users in August

All three predictors are **independent** and can be run separately. After execution, each predictor saves two models in the **/models** folder:

- `'CatBoost_{mode_type}'`
- `'LogReg_{mode_type}'`

Predictions are stored in the **/predictions** folder as pairs of `user_id - predicted value for August`.
Visualization of the model testing results is stored in the **/storage** folder.

Evaluation Metrics

For evaluation, I used **loss metrics** such as:

- **Mean Absolute Error (MAE)**
- **Mean Squared Error (MSE)**
- **R^2 Score**

However, these metrics are more suitable for comparing models rather than assessing absolute performance. Instead, I relied more on graphical representation of the results and consider them **satisfactory**.

Potential Improvements

To improve the results, I would:

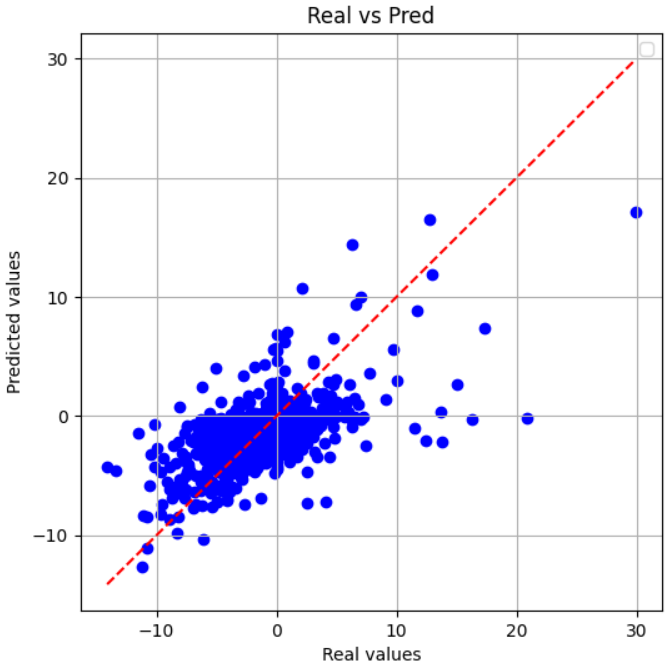
- Use more data
- Conduct a deeper analysis
- Reformulate the problem as a **classification task**, since classification results are more interpretable and easier to evaluate

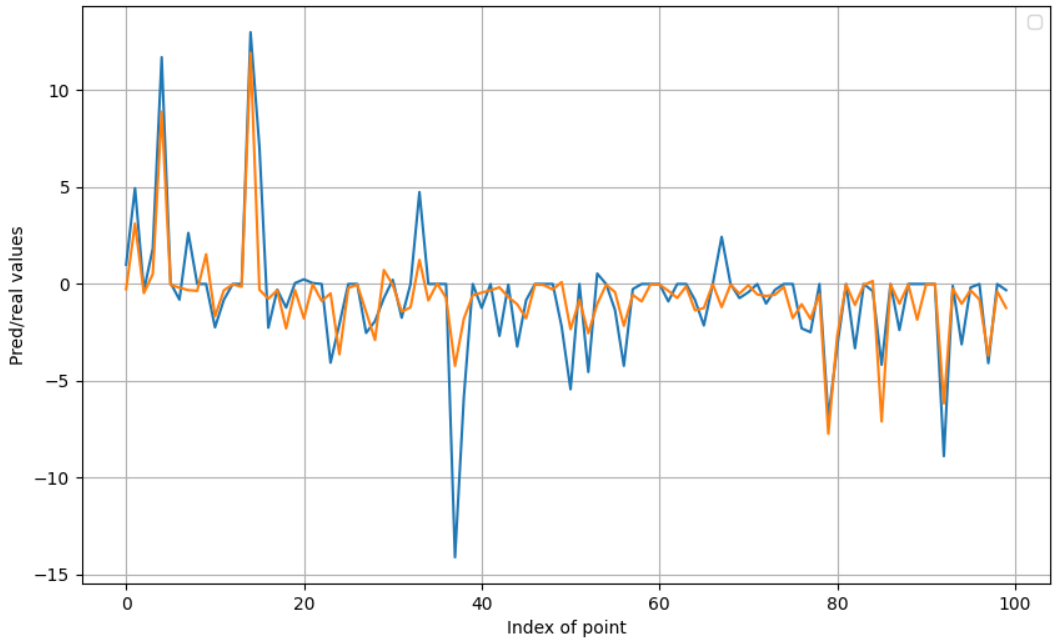
Obtained models could be used the next way. For example, we can subtract predicted expenses from predicted income, and if the difference is **positive**, we can classify a user as **potentially creditworthy**. The model that directly predicts the **difference** between income and expenses is particularly useful for this purpose. I prefer this approach because it uses a **single model instead of multiple**, reducing error accumulation and improving accuracy.

Additionally, model accuracy can be assessed by calculating the real difference between income and expenses, labeling it as **0 or 1**, and doing the same for predictions. Then, classification metrics such as **Precision** and **Recall** can be used for evaluation.

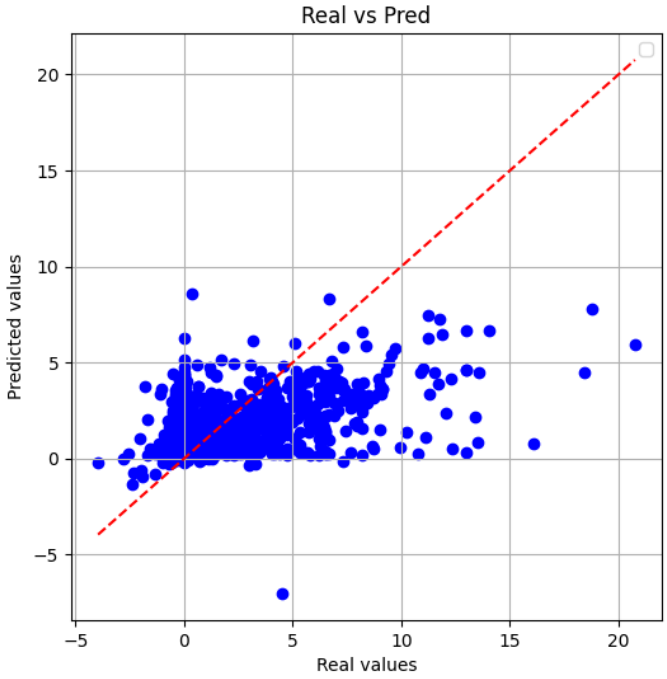
Below are examples of some graphical representations of the model results.

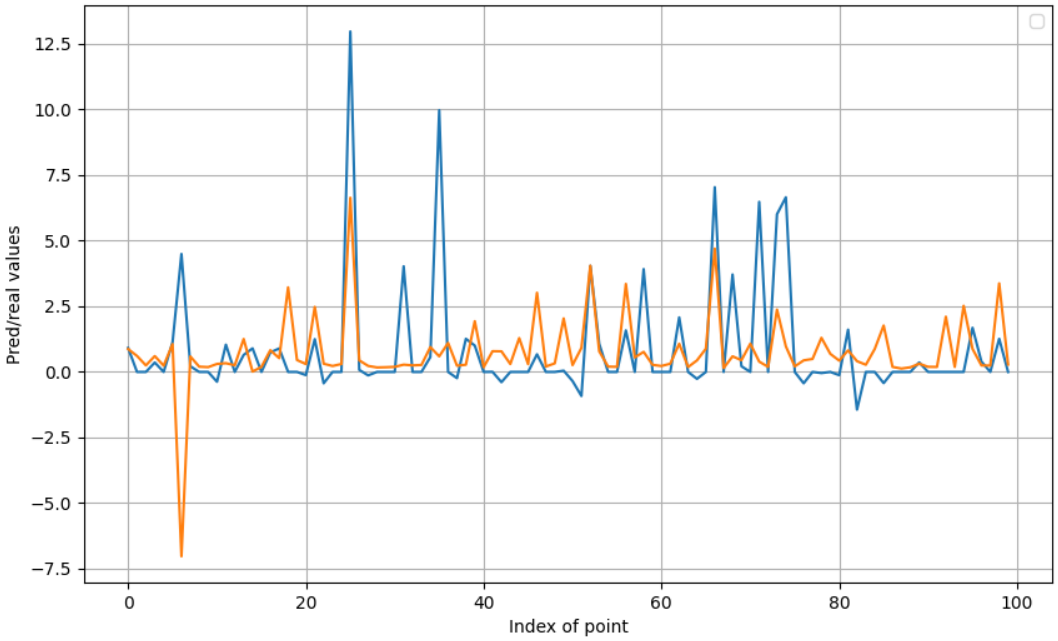
Outcome prediction using **Linear Regression**:



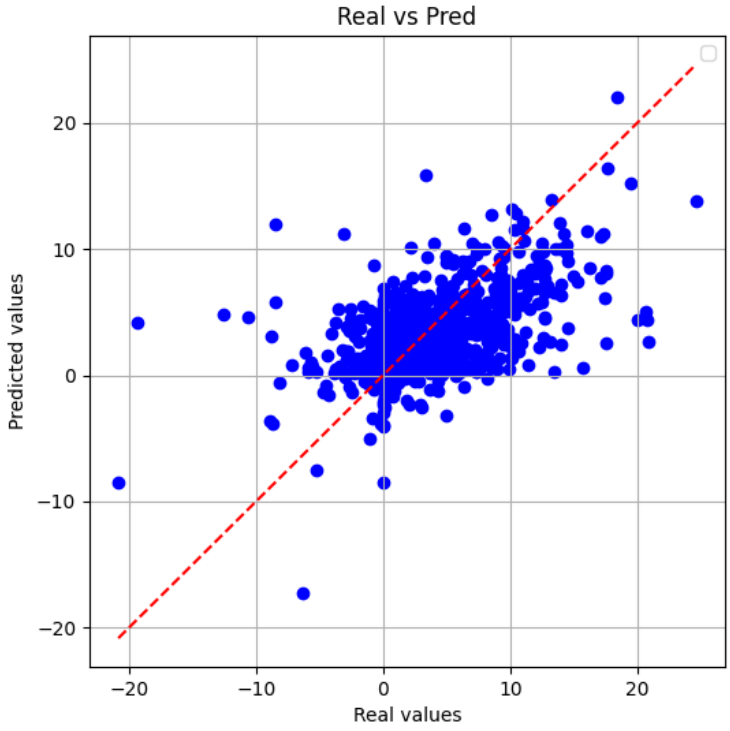


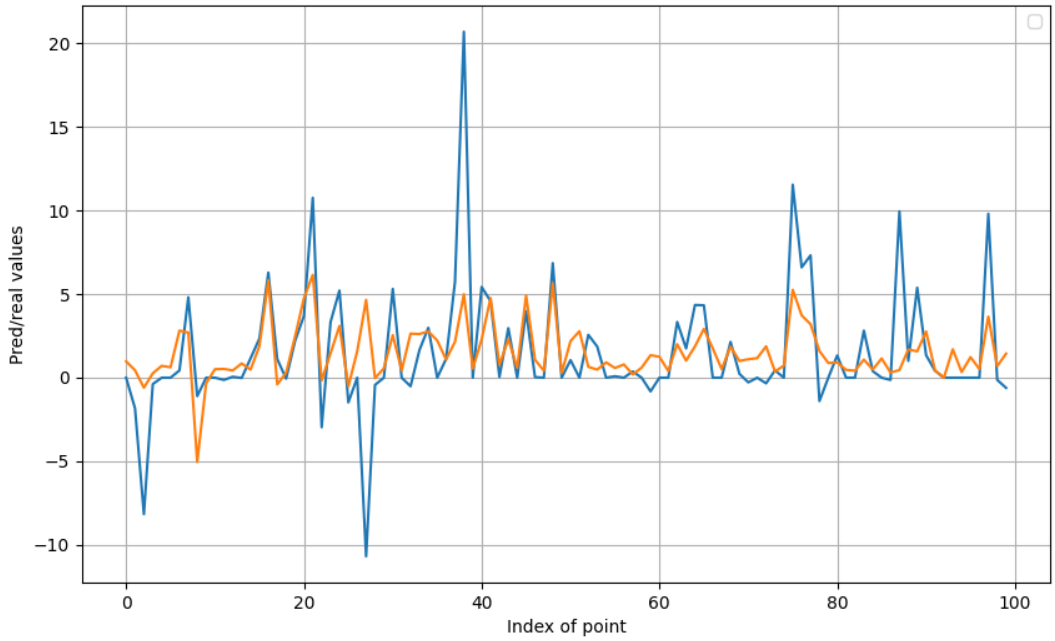
Income prediction using **CatBoost**:





Difference prediction using **Linear Regression**:





Potential Integration

Containerization has already been completed, so the model service can be deployed using an API for predictions—by creating a **REST API** with **FastAPI**, for example. This API will provide access to the models. For instance, the banking system sends a request with user data, the provided code processes it, the saved models generate predictions, and the result is returned via the API.