Machine Learning Optimization and Multimodal Challenge Report

Filip Rupniewski

8.11.2024

1 Introduction

This report outlines the analysis and model training process conducted on the given dataset, followed by evaluations using multiple machine learning models (XGBoost, Random Forest, Support Vector Regression, ElasticNet, Neural Networks). The tasks include data preprocessing, dimensionality reduction using PCA, model evaluation with and without PCA, and optimization of hyperparameters. We also examine the impact of embedding the description column and corresponding pictures.

2 Disclaimer

In this report, I opted to use "we" instead of "I" to give the impression that I had an entire team of researchers working alongside me — even if that team was just me and my ~~coffee~~ tea. This choice of language is my way of pretending that I'm part of an elite research group, rather than just one person typing away in a quiet room.

3 Challenge Description

This task consists of two steps, both involving the prediction of a target variable using tabular data. In the second step, text and image data are also utilized to improve model performance.

A provided CSV file contains approximately 40 000 rows and 80 features, including the target and a ‘description‘ column. Additionally, a folder contains images corresponding to the descriptions in the tabular data. The objective is to build models that optimize the prediction of the target variable, employing appropriate preprocessing and modeling techniques.

Step 1: Optimization with Tabular Data

- Use only the tabular data.
- Drop the ‘description‘ column for this step.
- Perform necessary exploratory data analysis (EDA) and feature engineering, as the data is not fully cleaned.
- Build and optimize a machine learning model to predict the target variable.

Step 2: Multimodal Learning with Text and Images

- Incorporate the ‘description‘ column and associated images from the `spacecraft_images` folder.
- **Text Data:** Encode the ‘description‘ using a neural network (e.g., embeddings, transformers). Each description is assumed to be unique.
- **Image Data:** Preprocess images and encode them into latent representations using a CNN or similar method.
- Build a multimodal model that integrates tabular features, text embeddings, and image embeddings to predict the target variable.

Step 3: Dockerfile

- Provide a `Dockerfile` that builds an image containing all dependencies required to run your scripts.
- Ensure the Docker image runs both steps of the task in an end-to-end manner.

4 Data Processing

4.1 Data Preparation

Data preparation involves loading the data, applying transformations, and preparing the features and target columns:

- We use a 1% sample of the data for initial analysis.
- The dataset is split into training and test sets (in proportion 80:20).
- Target variable `y` is transformed using Box-Cox (in our case `optimal_lambda` is close to 0, so the Box-Cox is almost a logarithm).
- We crop prefixes `'nr_'` and `'sh_'` from the beginning of entries of `feature_1` and `feature_2`.
- We drop colums with less than 2 non-null values and imput mean in missing values.
- We plot a Q-Q plot and apply a normality check to the target variable `y` (the null hypotesis of normality of data cannot be rejected) and remove the outliers (Figure 1).

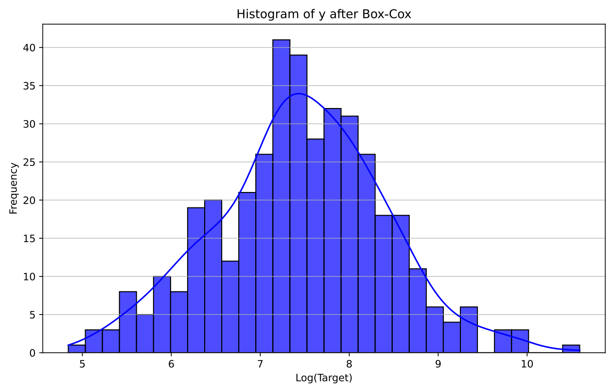
4.2 PCA for Dimensionality Reduction

Principal Component Analysis (PCA) is used for dimensionality reduction. We compute the first 45 principal components, which explain over 99% of the variance in the data. The results of the PCA transformation are used for training models (Figure 2).

4.3 Model Training and Evaluation

We trained several models on the transformed data:

- Models trained include XGBoost, Random Forest, Support Vector Regression (SVR), ElasticNet, and Neural Network.
- We evaluate the models on both PCA-reduced data and the full dataset without PCA.



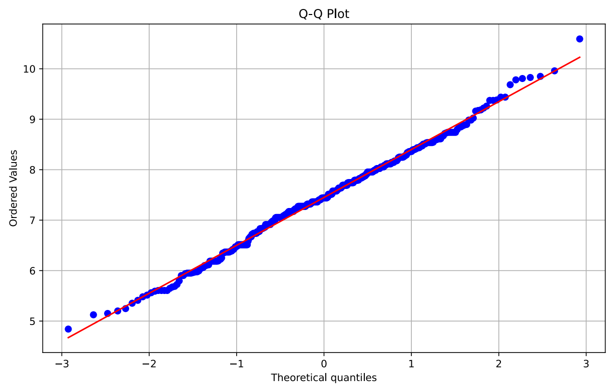


Figure 1: histogram and Q-Q plot of target after applying Box-Cox transformation and removing outliers.

- The evaluation results shows that XGBoost performs best, so we are going to fine-tune it on complete dataset (Table 1).

4.4 Full Data Processing

For the complete dataset (splitted in proportion 80% training set, 20% test set), we fine-tune the XGBoost model parameters ('learning_rate', 'max_depth', 'n_estimators', 'subsample') using GridSearchCV. We repeat the process for both PCA and non-PCA data:

The XGBoost model achieves RMSE (Root of Mean Square Error) of 0.3802 when using 50 principal components and an RMSE of **0.3156** when no

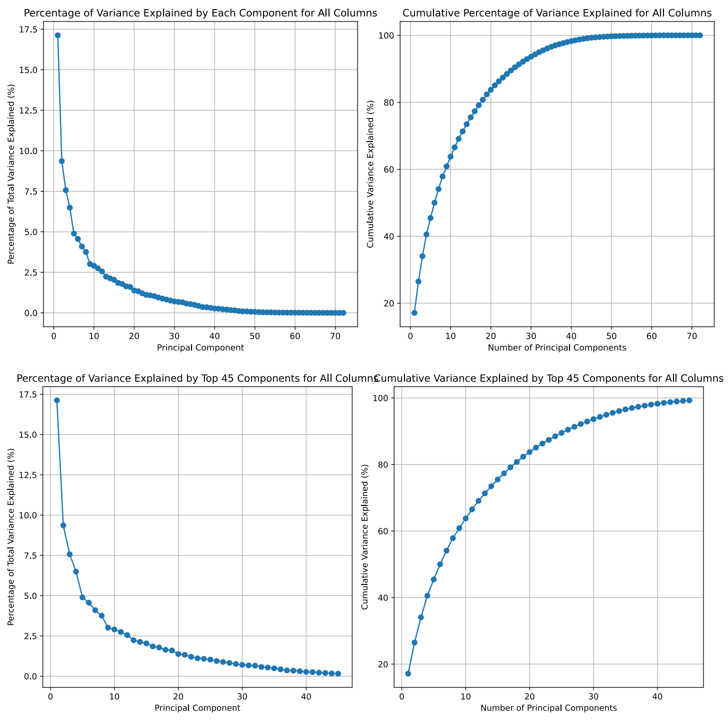


Figure 2: Principal component analysis of 1% of the dataset and (cumulative) percentage of variance explained by top components.

dimensionality reduction is applied (Figure 3). The best parameters obtained for the model are:

$$\{\texttt{learning\_rate} : 0.05, \texttt{max\_depth} : 9, \texttt{n\_estimators} : 400, \texttt{subsample} : 0.7\}.$$

5 Step 2: Description Embedding and PCA on Description Embeddings

5.1 BERT Embedding of Description Column

The description column is embedded using BERT, resulting in new columns labeled as `desc_*`. The model performance (XGBoost) with description embeddings showed an RMSE of 0.3387 for the full dataset (and 0.4288 for a 10% sample).

Table 1: Results for different data subsets and principal component restrictions (without using data from description column nor images). XGBoost perfoms best.

Model	**1% of Data**		**10% of Data**	
	Non-red.	**PCA Red.**	**Non-red.**	**PCA Red.**
XGBoost	0.478751	0.524714	0.441654	0.528197
Random Forest	0.496532	0.559146	0.459571	0.530138
SVR	0.558672	0.557516	0.504502	0.503616
ElasticNet	0.573751	0.577160	0.611285	0.618885
Neural Network	-	-	4.519947	3.957845

5.2 PCA for Description Embeddings

To reduce the dimensionality of the description embeddings, we apply PCA. We evaluate XGBoost with different numbers of principal components (50, 10, and 1). The performance of the models is evaluated for each case (Table 2).

5.3 One-Hot Encoded Descriptions

We check also the performance of One-Hot Embedding of a description column. It does not fullfil the condition of the challenge and do not scale well when number of unique solutions is more than just 50 as in out case. Still, we were curious how does it performs in comparison with BERT embedding. The RMSE achieved with one-hot encoded descriptions is 0.3389 for the full dataset (and XGBoost with the same parameters as before).

5.4 Feature importance plots

After training each of the models in this section, we visualized the top 30 most important features (Figures 4, 5, 6). The most important features which do not come from description column are these with indices 27, 52, and 46. Depending on the embedding and applying PCA reduction either they are the most important (BERT + PCA 1, BERT + PCA 10, One-Hot Encoding) or only the first of them is visible among 30 most important features (BERT + no PCA).

Table 2: Results of XGBoost trained on all rows: without description column, with BERT description embedding, after applying PCA on columns coming from description embedding (with 50,10,1 components), and using One-Hot-Embedding. Counterintuitively not using description gives the best results.

XGBoost on all rows	**RMSE**
without description column	0.3156
BERT desc. emb. and PCA reduction (1)	0.3224
BERT desc. emb. and PCA reduction (10)	0.3294
BERT desc. emb. and PCA reduction (50)	0.3338
with BERT desc. emb. and no PCA red.	0.3386
with One-Hot-Embedding	0.3389

5.5 Independent Predictions for Each Description

We also explore independent predictions for each unique value in the 'description' column (using XGBoost, SVR and ElasticNet). This involves producing histograms of the target for each description and evaluating the model on a per-description basis. The overall XGBoost RMSE is 1.093 for the full dataset. It is slightly better in case of SVR and ElasticNet (1.008 and 1.004 correspondingly). Thus, this idea gives much worse results than the previous approaches (and it do not scale well when there is more than 50 unique descriptions). One can find all the histograms and scatterplots for the XGBoost case in the folder `'pictures'`.

6 Spacecraft Image Processing

To incorporate image data, we process the spacecraft images by change resolution to 128x128 and removing transparency. Since we do not use DNN approach we flatten the tensor 3x128x128 to a vector. We apply PCA on both image and description data (coming from BERT embedding as before). After PCA transformation, the model is evaluated on the reduced data:

- With image columns reduced by PCA (46) and without `desc_` columns, the RMSE is 0.4318.
- With both image and description columns reduced by PCA (correspondingly, 46 and 50 most important principal components), the RMSE

is 0.4381.

7 Results and Summary

7.1 Execution Time Summary

The total execution time is approximately 25 minutes when running locally (CPU Model: Intel Core i7-10710U (6 cores, 12 threads) Max Speed: 4.7 GHz, RAM: 8 GB RAM, Swap: 8 GB) or 50 minutes when running using container (8 GB RAM, 4 GB Swap, no restriction on processor).

7.2 GPU and sound notification

We tried using a GPU, but the time savings were not significant. Our functions can be used with a GPU by setting `CPU=False`, but this feature is not available if you run the script using the container. We decided not to offer this feature due to the complexity of creating a container compatible with various GPUs.

There is also a function `play_sound()` that informs when a code fragment has finished running. However, we commented it out because it does not work when running in the container.

7.3 Conclusion

Counterintuitively, excluding both description and image embeddings yields the best results in our case. The optimal model is XGBoost with parameters { `learning_rate`: 0.05, `max_depth`: 9, `n_estimators`: 400, `subsample`: 0.7 }, trained on data without description or images. This model achieves an RMSE of 0.3156 on the prediction of a Box-Cox transformed target. In Table 3 you can find the comparison of different approaches with XGBoost. For models different than XGBoost see Table 1.

7.4 Files Included in the Main Directory and Attached to the Solution

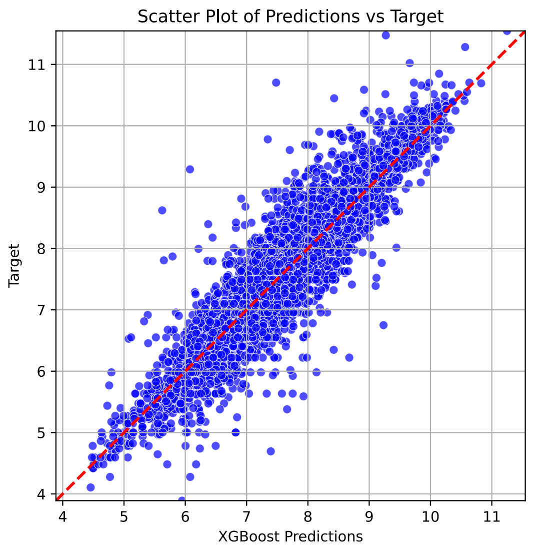
The main directory contains the following files and resources:

- `Dockerfile`: Defines the Docker container for the solution.

Table 3: XGBoost model performance summary. For models different than XGBoost see Table 1.

XGBoost	**RMSE**
without description column nor img_ nor desc_ cols	**0.3156**
BERT desc. emb. and PCA reduction (1)	0.3224
BERT desc. emb. and PCA reduction (10)	0.3294
BERT desc. emb. and PCA reduction (50)	0.3338
with BERT desc. emb. and no PCA reduction	0.3386
with One-Hot Encoding of description	0.3389
with PCA (46) on img_ cols without desc_ cols	0.4318
with PCA (46) on img_ cols and PCA (50) on BERT desc_ cols	0.4381

- `requirements.txt`: Lists the necessary dependencies for the docker.
- `optimization_challenge.tar`: Docker container (alternative to building the container from the `Dockerfile` and `requirements.txt`).
- `my_functions.py`: Python file containing all supporting functions.
- `solution_script.py`: Main Python script for executing the solution.
- `candidates_data.csv`: Dataset used in the analysis.
- `spacecraft_images`: Folder containing relevant image files for the project.
- This `report.pdf`: Document detailing the methodology, results, and conclusions.



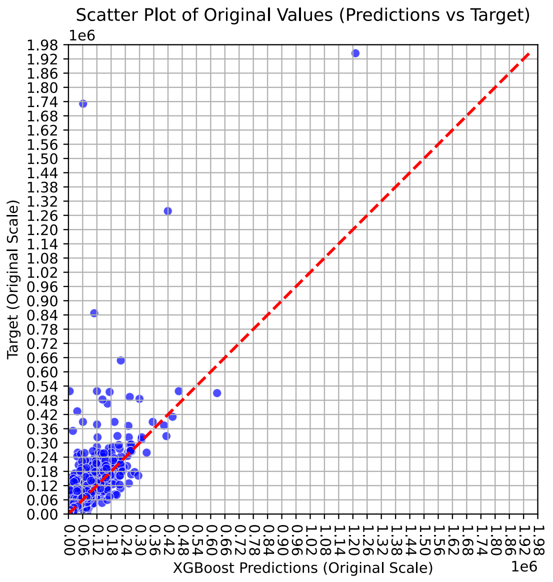
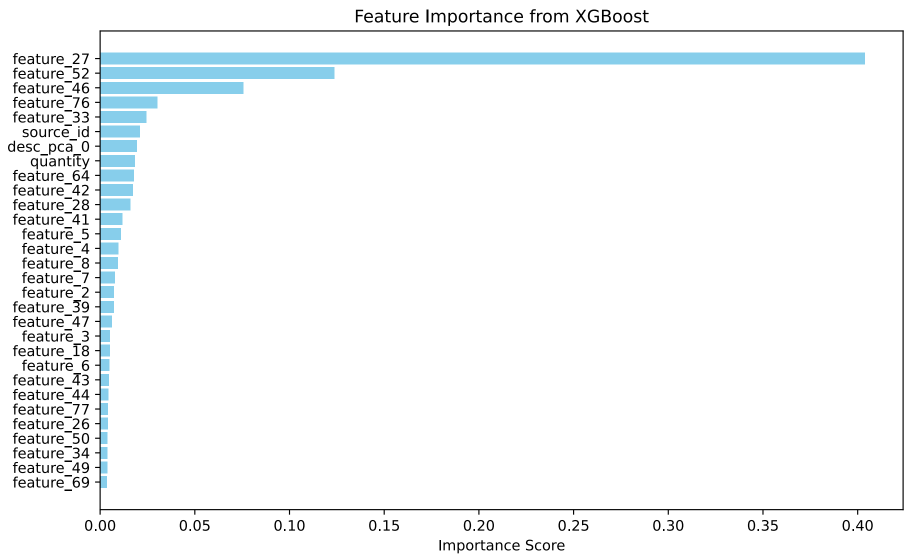


Figure 3: Scatter Plot of Prediction (using XGBoost trained without description nor images data) vs Target (before and after inverse Box-Cox)



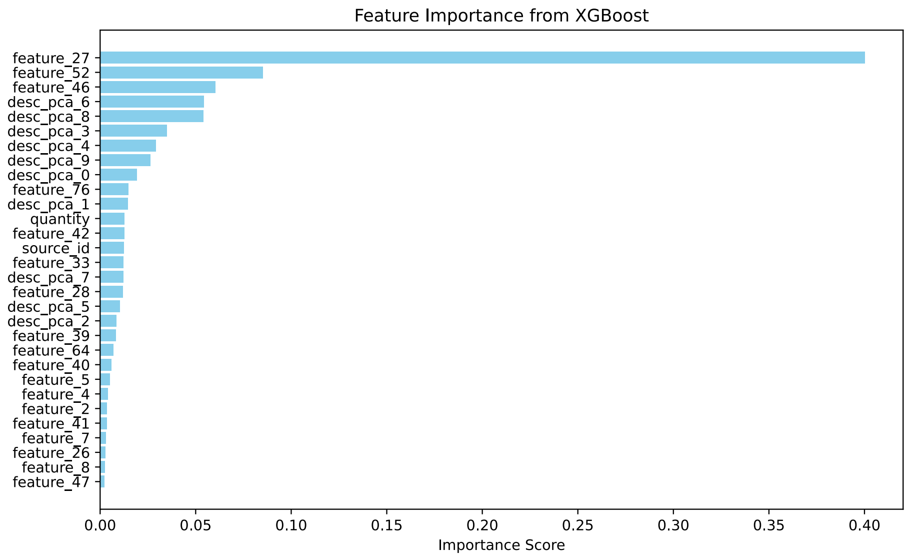
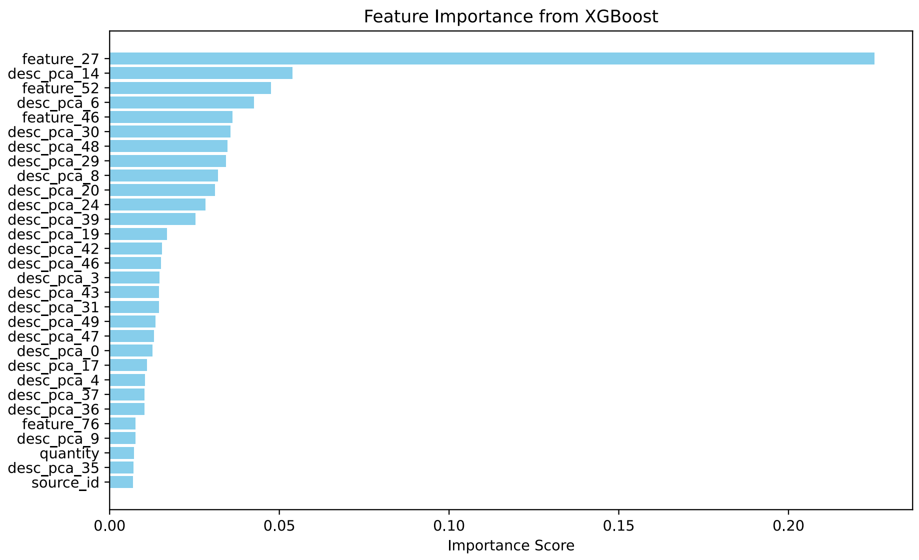


Figure 4: Feature importance plots for: BERT desc. emb. and PCA reduction (1) and BERT desc. emb. and PCA reduction (10).



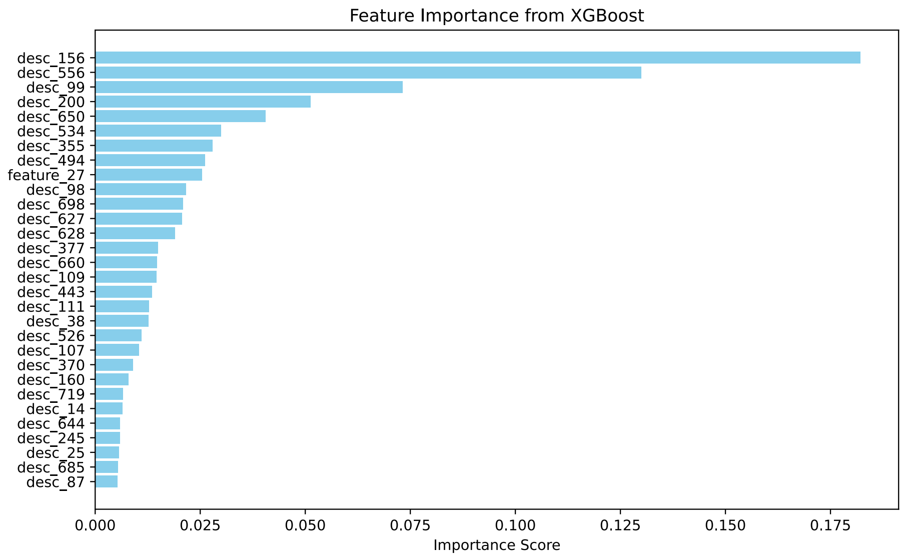


Figure 5: Feature importance plots for: BERT desc. emb. and PCA reduction (50) and BERT desc. emb. without PCA reduction.

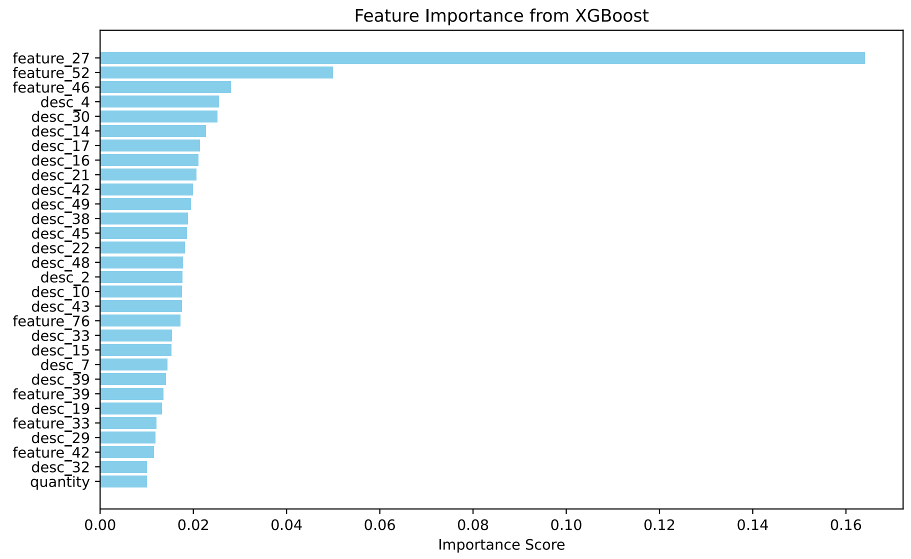


Figure 6: Feature importance plot for One-Hot Encoding of description.