ML@NOVA
DFJ

Predicting Survival Time in Multiple Myeloma Patients

Team identification

Name 1: José Costa

Number 1: 72899

Name 2: Dinis Santos

Number 2: 72815

Name 3: Francisco Jorge

Number 3: 70293

Final score on the private leaderboard: 2.58777

Leaderboard private ranking: 30

Task [1.1] - Data preparation and validation pipeline

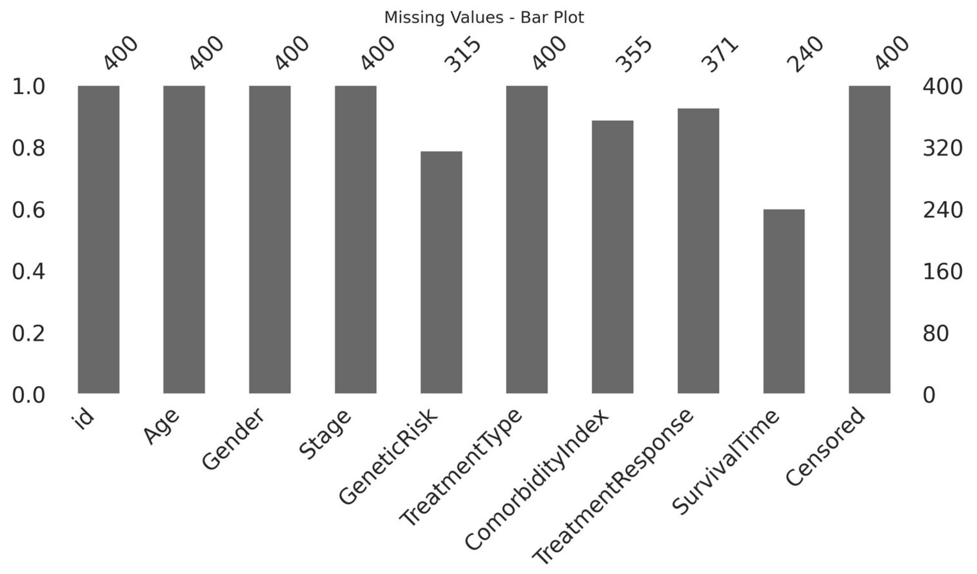
What was done in task [1.1]

1. Missing Values Analysis	Visualized missing values using multiple methods (bar plot, heatmap, matrix, dendrogram)
	Created comprehensive overview of data completeness
	Identified patterns in missing data
2. Data Cleaning	Dropped rows with missing SurvivalTime values
	Removed columns containing missing data (baseline approach)
	Excluded censored cases (where Censored == 1)
	Retained only complete, uncensored observations
3. Feature Exploration	Visualized feature relationships using pairplot
	Analyzed correlations between Age, Gender, Stage, TreatmentType, and SurvivalTime
	Examined distribution patterns across features

What was done in task [1.1]

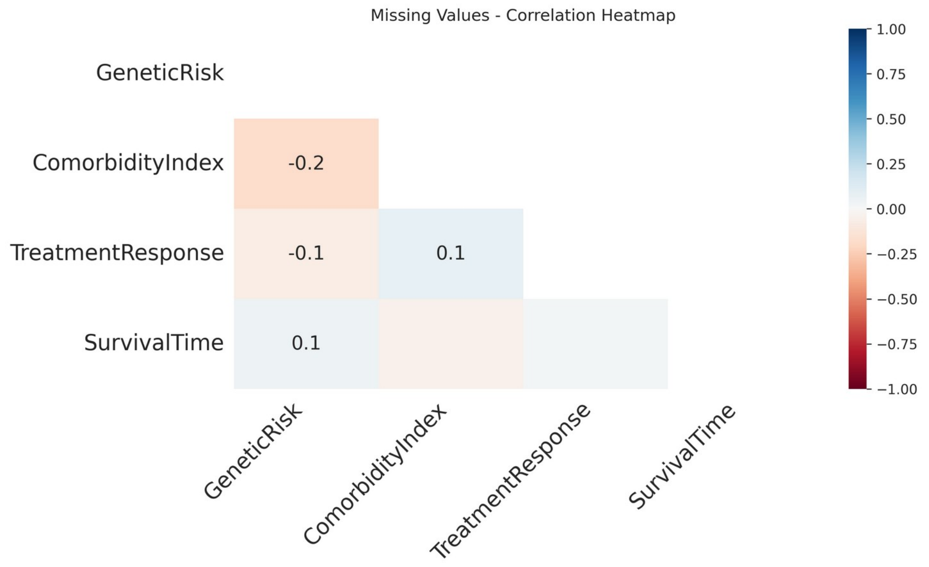
4. Data Preparation	Defined feature matrix (X) by dropping target and identifier columns
	Isolated target variable (y) as SurvivalTime
	Preserved censoring indicator for potential future use
5. Validation Strategy Development	Implemented train/validation/test split (64%/16%/20%)
	Tested simple split approach with Linear Regression
	Implemented 5-fold cross-validation for more robust evaluation
	Compared both validation strategies (simple split vs. cross-validation)
6. Performance Evaluation	Calculated MSE (Mean Squared Error) and cMSE (Censored MSE)
	Evaluated model performance on validation and test sets
	Compared cross-validation results to simple split results

Results and Analysis from task [1.1]



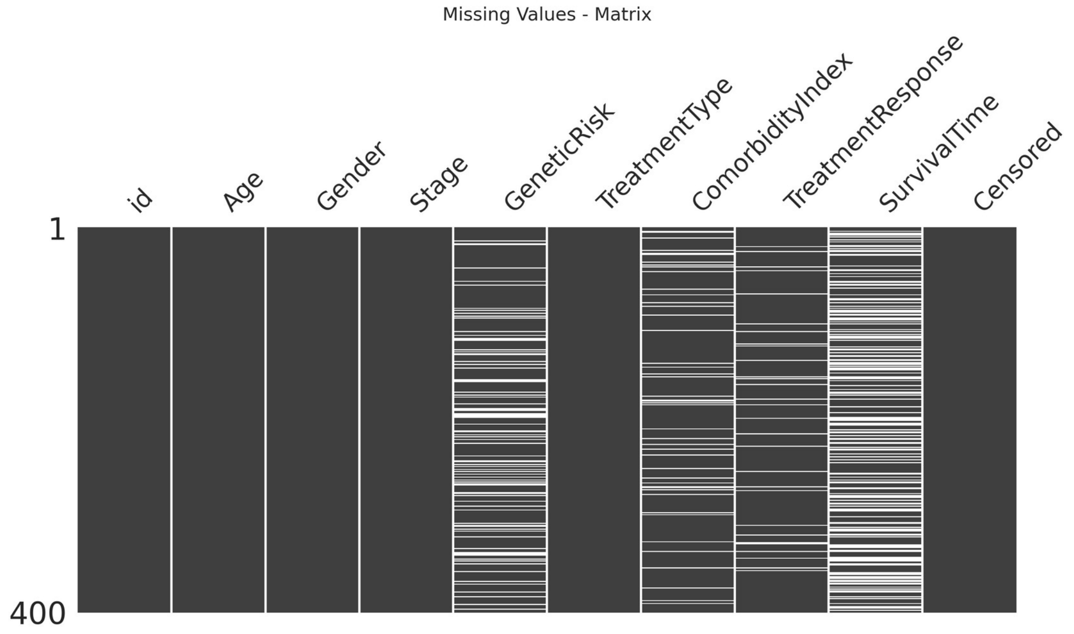
Bar chart showing missing-value counts per variable, with labels indicating how many entries are missing in each column.

Results and Analysis from task [1.1]



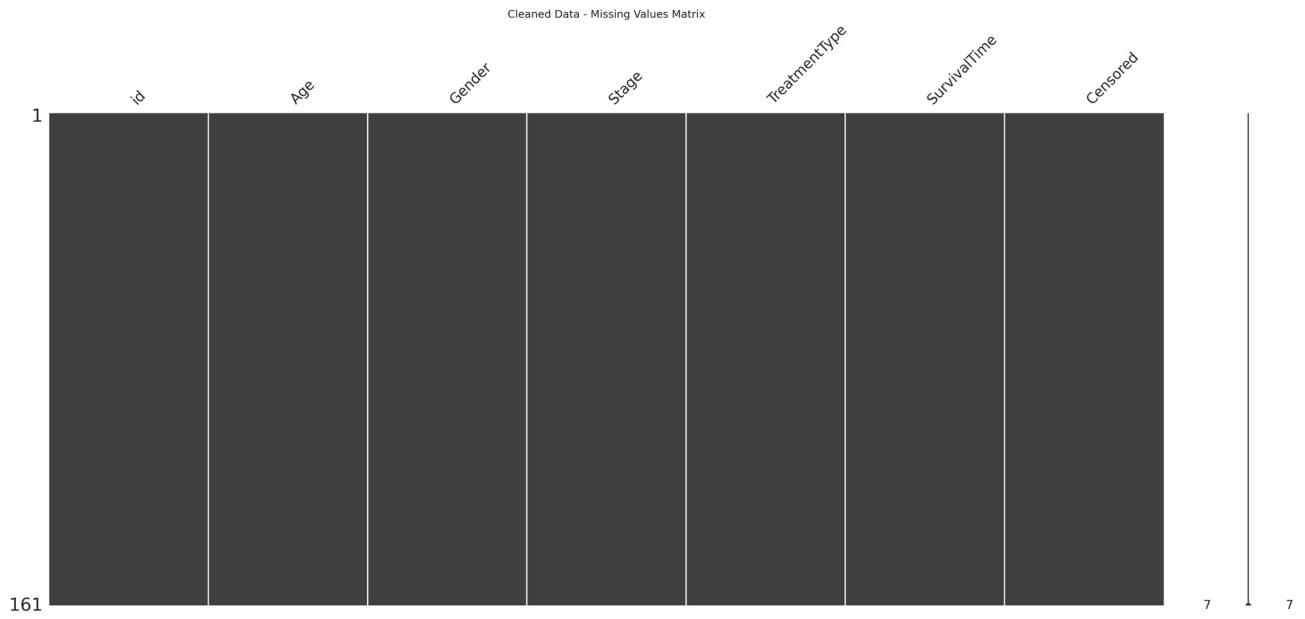
Heatmap showing correlations between missing-value patterns across selected variables, with mostly weak relationships.

Results and Analysis from task [1.1]



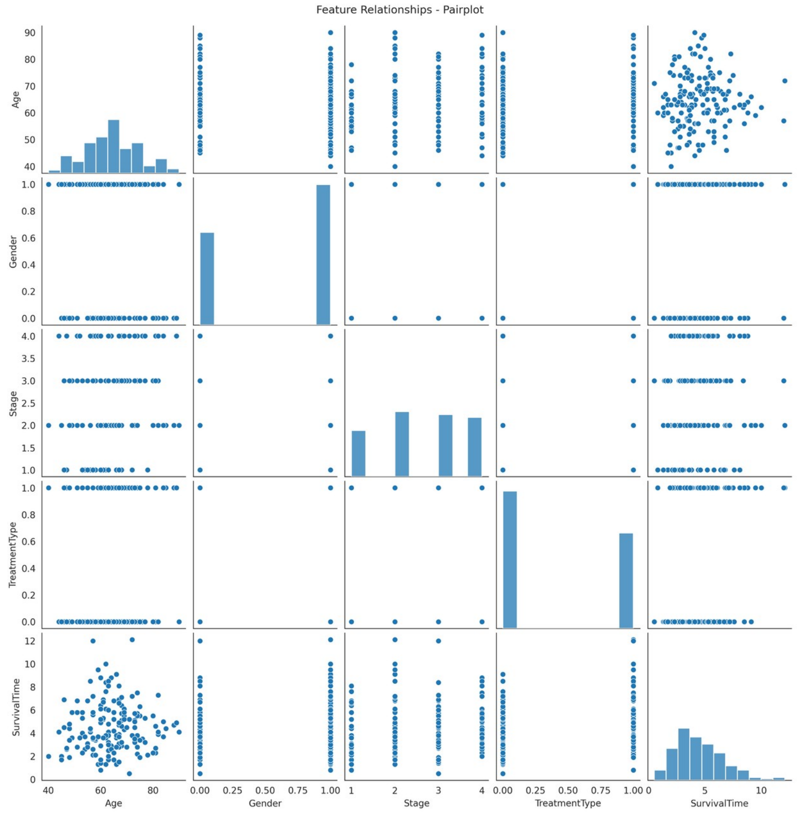
Matrix plot highlighting which entries are missing across variables, showing the distribution and density of gaps in the dataset.

Results and Analysis from task [1.1]



Matrix plot showing the cleaned dataset, confirming that all variables have no remaining missing values.

Results and Analysis from task [1.1]



Age
• Roughly uniform distribution between 40 and 85.
• No clear relationship with Stage, Gender, or TreatmentType.
• Slight indication that lower ages may correspond to higher SurvivalTime, but with large variability.

Gender
• Binary variable with no meaningful pattern across other features.
• SurvivalTime similarly distributed between genders.

Stage
• Balanced distribution across stages 1–4.
• Clear trend: higher stages correspond to lower SurvivalTime.
• No noticeable relationship with Age or Gender.

TreatmentType
• Most individuals fall into one or two treatment categories.
• No strong association with Age or Gender.
• SurvivalTime varies within each treatment group without strong separation.

SurvivalTime
• Right-skewed distribution, with most values between 1 and 5 months.
• Strongest visible relationship is with Stage (more advanced stages → shorter survival).
• No clear patterns with Gender or TreatmentType.

Results and Analysis from task [1.1]

	MSE	cMSE
Simple Split	5.0473	5.0473
Cross-Validation	4.0873	4.0873
Pipeline	4.4112	4.4112

The results show that cross-validation improves performance compared to a simple train–test split, reducing the MSE by almost one unit.
The pipeline approach performs slightly worse than cross-validation but still better than the simple split.

Task [1.2] - Learn the baseline model

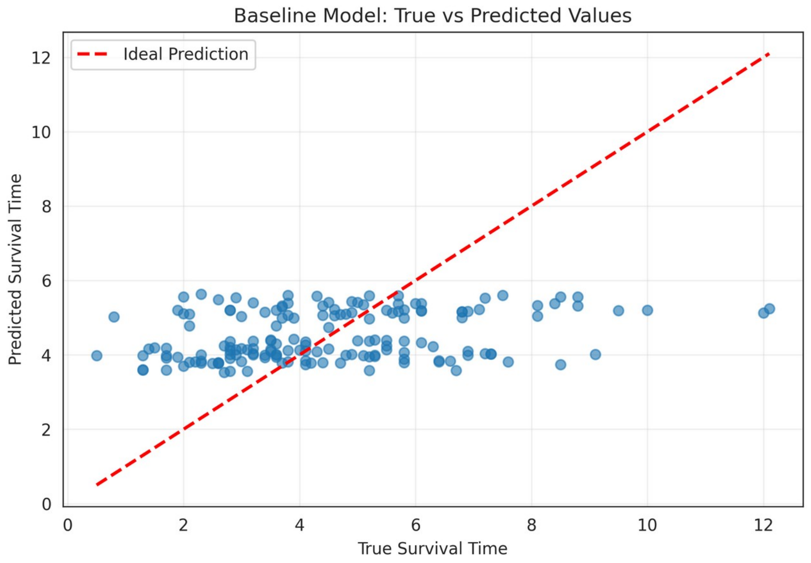
What was done in task [1.2]

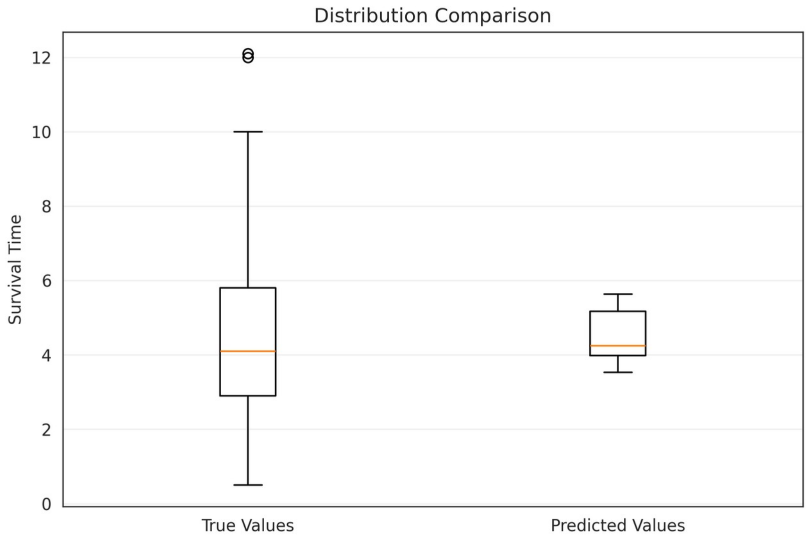
1. Pipeline Construction	- Built baseline pipeline combining StandardScaler and Linear Regression
	- Ensured feature scaling for improved model performance
	- Created modular, reusable pipeline structure
2. Cross-Validation Training	- Performed 5-fold cross-validation for robust model evaluation
	- Calculated CV MSE scores across all folds
	- Computed mean and standard deviation of cross-validation performance
3. Final Model Training	- Fitted baseline pipeline on entire training dataset
	- Generated predictions on training data
	- Maximized use of available data for final model

What was done in task [1.2]

4. Performance Metrics	- Calculated Training MSE (Mean Squared Error)
	- Calculated Training cMSE (Censored Mean Squared Error)
	- Established baseline performance benchmarks
5. Test Predictions & Submission	- Loaded test dataset and prepared features
	- Generated predictions for test samples
	- Created submission file for competition/evaluation
6. Model Visualization	- Created scatter plot comparing true vs predicted survival times
	- Generated boxplot for distribution comparison
	- Visualized model fit quality and prediction patterns
	- Saved individual plots for documentation

Results and Analysis from task [1.2]





The scatter plot shows that the model's predictions are squeezed into a narrow band, mostly around 4–6 years and rarely follow the true values across their full range.
The boxplots make this even clearer: the true survival times vary widely, while the predicted ones stay tightly grouped.
Overall, the model isn't capturing the real variability in the data and is underfitting.

Results and Analysis from task [1.2]

- Baseline Model Training with Cross-Validation

MSE	cMSE	Best Kaggle Score
4.0873	4.0873	3.87347

Task [2.1] - Development

What was done in task [2.1]

1. Polynomial Regression Function Development	- Created `train_polynomial_regression()` function with hyperparameter search
	- Implemented cross-validation for degree selection (testing degrees 1 to max_degree)
	- Added early stopping mechanism (stops after 2 consecutive iterations without improvement)
	- Returned best degree, trained model, and complete CV results dictionary
2. k-Nearest Neighbors Function Development	- Created `train_knn()` function with hyperparameter search
	- Implemented cross-validation for k selection (testing k from 1 to max_k)
	- Added early stopping mechanism for efficiency
	- Returned best k value, trained model, and complete CV results dictionary
3. Hyperparameter Selection	- Used 5-fold cross-validation for both models
	- Searched polynomial degrees from 1 to 10
	- Searched k values from 1 to 20
	- Tracked MSE scores with standard deviations for each hyperparameter

What was done in task [2.1]

4. Model Training	- Trained Polynomial Regression with optimal degree on full dataset
	- Trained k-NN Regression with optimal k on full dataset
	- Generated predictions on training data for both models
5. Performance Evaluation	- Calculated training MSE for both models
	- Calculated training cMSE for both models
	- Compared performance against baseline expectations
	- Documented hyperparameter selection results

Results and Analysis from task [2.1]

- Model performance for polynomial and k-NN Regression

	MSE	cMSE
Polynomial	4.0873	4.0873
k-NN	3.6571	3.6571

Task [2.2] - Evaluation

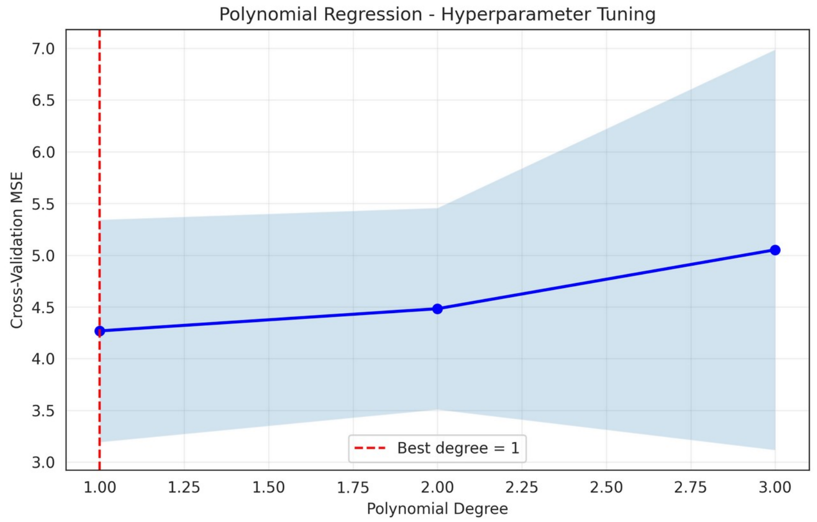
What was done in task [2.2]

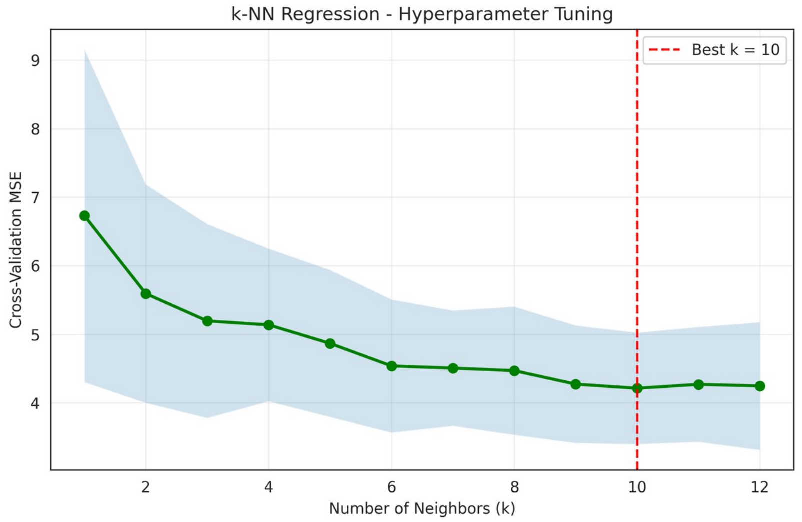
1. Comprehensive Model Comparison	- Created comparison table with baseline, polynomial regression, and k-NN models
	- Included hyperparameter configurations for each model
	- Displayed min, max, mean, and standard deviation of errors
	- Identified best performing model based on mean cross-validation error
2. Hyperparameter Tuning Visualization	- Plotted polynomial degree vs MSE with confidence intervals
	- Plotted k-value vs MSE with confidence intervals
	- Marked optimal hyperparameters with vertical lines
	- Showed performance trends across hyperparameter ranges
3. Model Predictions Comparison	- Created scatter plots of true vs predicted values for all three models
	- Displayed MSE on each plot for direct comparison
	- Included ideal prediction line (y=x) as reference
	- Generated combined and individual visualization plots

What was done in task [2.2]

4. Statistical Analysis	- Computed cross-validation statistics for each model
	- Analyzed variance in predictions across folds
	- Compared model stability through standard deviation metrics
	- Evaluated improvement over baseline model
5. Test Set Predictions	- Selected best performing model based on CV results
	- Generated predictions for test dataset
	- Created submission file for evaluation
	- Documented model selection rationale
6. Results Documentation	- Saved all comparison plots with task-specific naming
	- Generated separate plots for polynomial and k-NN tuning
	- Created individual prediction visualizations for each model
	- Documented complete evaluation workflow

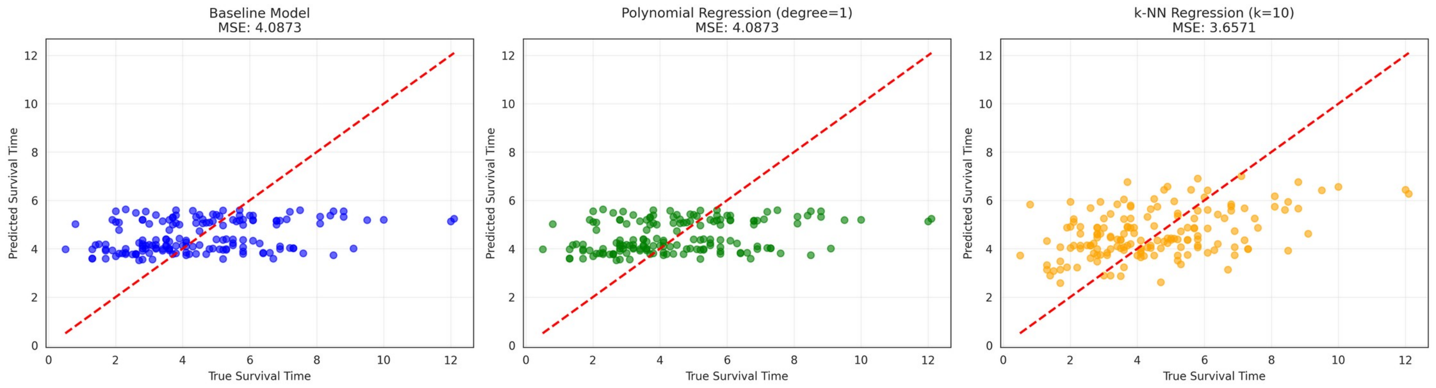
Results and Analysis from task [2.2]





The tuning results show that polynomial regression performs best at degree 1, meaning that adding polynomial terms does not improve the model and only increases error.
In contrast, k-NN regression benefits from increasing the number of neighbors: the error decreases steadily and reaches its minimum around k = 10.
This indicates that a smoother, more averaged prediction strategy fits the data better than more complex polynomial relationships.

Results and Analysis from task [2.2]



The baseline and polynomial regression models behave almost identically: both compress predictions into a narrow band and fail to follow the true survival times across their full range.
The k-NN model improves this pattern. It spreads predictions more realistically and gets closer to the ideal diagonal, which is reflected in its lower MSE.
Even so, the improvement is modest, and the model still underestimates higher survival times.

Results and Analysis from task [2.2]

- Best submission for non-linear in Kaggle: **3.29914**
- Best submission model : K-NN Regressor

Task [3.1] - Missing data imputation

What was done in task [3.1]

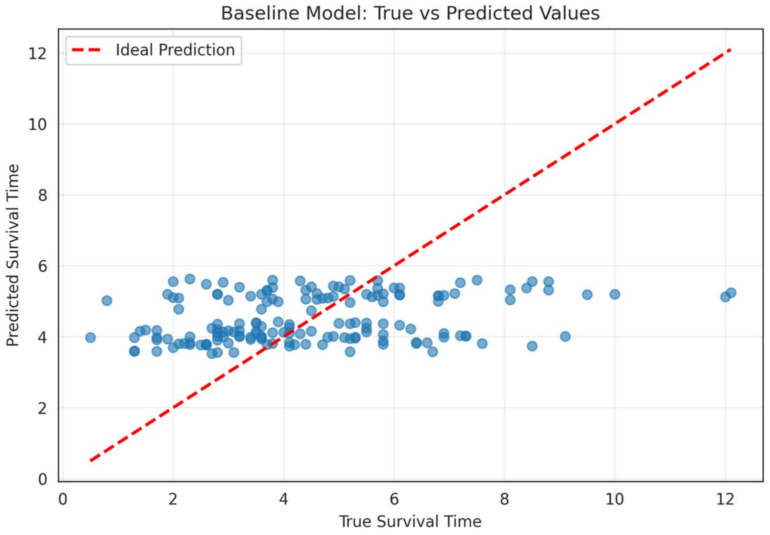
1. Data Preparation	- Load original dataset with missing values
	- Analyze missing value patterns
	- Prepare feature matrix (X) and target variable (y) with missing data intact
2. Imputation Strategies	- Mean Imputation: Replace missing values with column means
	- KNN Imputation: Use k-nearest neighbors to estimate missing values
	- Iterative Imputation: Use Bayesian Ridge regression for multivariate imputation
3. Model Evaluation	- Train baseline Linear Regression model on each imputed dataset
	- Evaluate using both train/test split and cross-validation approaches
	- Compare performance using cMSE (Censored Mean Squared Error)
	- Test with KNN Regression model for comparison

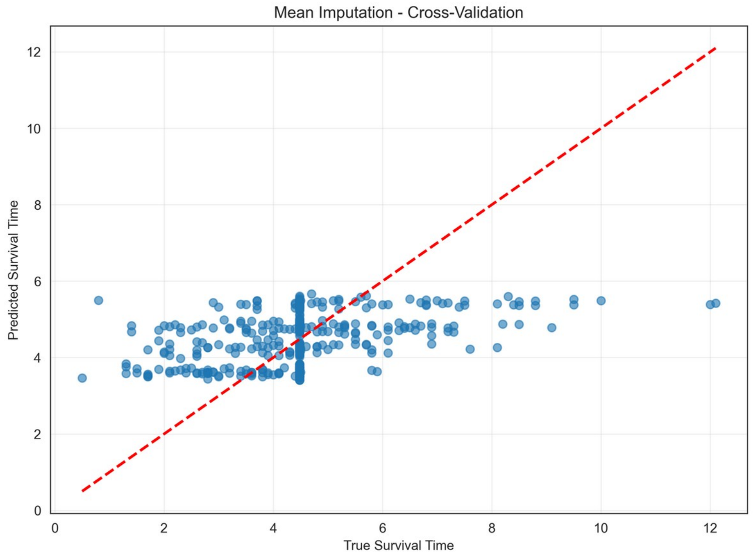
Results and Analysis from task [3.1]

Imputation Strategy	cMSE (Train/Test)	cMSE (Cross-Val)	cMSE (KNN Model)	Best k (KNN Model)
Mean	1.478845	1.764469	1.747625	20
kNN	1.489639	1.772562	1.756381	20
Iterative	1.489639	1.767101	1.739146	20

Best Imputation Strategy	Best Cross-Validation cMSE
Mean	1.7645

Results and Analysis from task [3.1]





- The right plot, corresponding to the mean imputation with cross-validation, shows a wider spread of predicted values and slightly better alignment with the diagonal for medium-range survival times.
- Overall, the cross-validated mean imputation model provides a modest improvement over the baseline, but both models still show bias toward the center of the survival time range.

Task [3.2] - Train models that do not require imputation

What was done in task [3.2]

1. Data Preparation	- Loaded dataset with missing values (no imputation required)
	- Removed rows with missing `SurvivalTime` values only
	- Defined feature matrix, target variable, and censoring indicator
2. Decision Tree Model	- Replaced missing values with placeholder (-999) for tree compatibility
	- Performed GridSearchCV with 5-fold cross-validation on hyperparameters
	- Visualized predictions, residuals, and feature importance
3. HistGradientBoosting Model	- Utilized native missing value support (no imputation needed)
	- Performed GridSearchCV with 5-fold cross-validation
	- Generated training and CV predictions with performance visualizations

What was done in task [3.2]

4. CatBoost Standard Regression	- Leveraged CatBoost's native missing value handling
	- Performed GridSearchCV with 5-fold cross-validation
	- Evaluated performance with scatter plots, residuals, and feature importance
5. CatBoost AFT (Survival Analysis)	- Implemented Accelerated Failure Time model for censored data handling
	- Created interval targets (y_lower, y_upper) and trained with SurvivalAft loss
	- Generated predictions and aligned with Task 3.2 dataset for comparison

Results and Analysis from task [3.2]

Strategy	Model	MSE	cMSE
Baseline (Drop Missing)	Linear Regression	4.087307	4.087307
Mean Imputation	Linear Regression	N/A	1.764469
KNN Imputation	Linear Regression	N/A	1.771368
Iterative Imputation	Linear Regression	N/A	1.767101
Missing as -999	Decision Tree	2.231226	1.959689
Native Missing Support	HistGradientBoosting	2.628646	2.271743
Native Missing Support	CatBoost Standard	2.668006	2.306718
Native Missing Support	CatBoost AFT	2.792279	1.733899

Task [3.3] - Evaluation

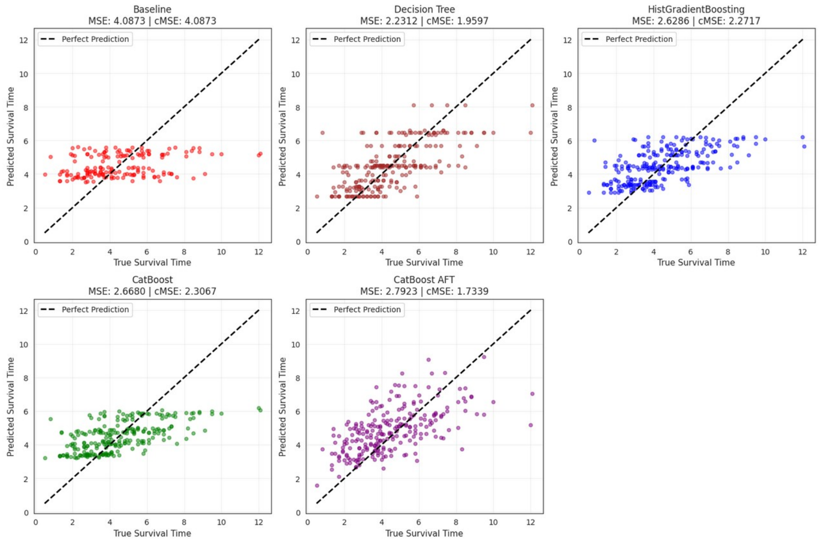
What was done in task [3.3]

1. Comparison Analysis	- Built comparison table with all strategies: baseline, imputation methods (Mean, KNN, Iterative), and models handling missing data (Decision Tree, HistGradientBoosting, CatBoost AFT)
	- Displayed MSE and cMSE metrics for all approaches
	- Created y vs y-hat scatter plots for visual comparison of model performance
2. Combined Approach Testing	- Selected best imputation strategy from Task 3.1: Mean Imputation (cMSE: 1.7645)
	- Combined with best model from Task 3.2: CatBoost AFT
	- Trained CatBoost AFT on mean-imputed data
	- Evaluated performance: cMSE of 3.1236 (worse than native missing handling)

What was done in task [3.3]

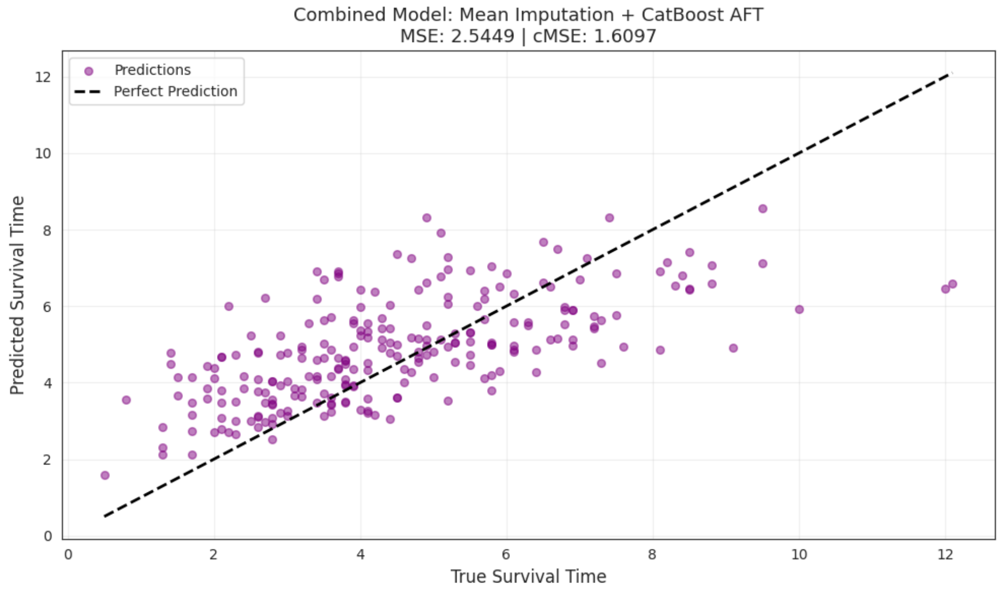
3. Best Model Selection	- Compared all strategies including combined approach
	- Identified CatBoost AFT with native missing support as best performer (cMSE: 1.7339)
4. Test Predictions & Submission	- Generated predictions on test data using best model (CatBoost AFT)
	- Created Kaggle submission file: `handle-missing-submission-xx.csv`

Results and Analysis from task [3.3]



Across all tested models, the CatBoost AFT approach delivered the most accurate predictions. It achieved the lowest cMSE and showed the strongest alignment with the diagonal in the regression plot.

Results and Analysis from task [3.3]



Comparing this plot that combines Mean Imputation + CatBoost AFT we can clearly see an improvement

Results and Analysis from task [3.3]

Strategy	Model	MSE	cMSE
Baseline (Drop Missing)	Linear Regression	4.087307	4.087307
Mean Imputation	Linear Regression	N/A	1.764469
KNN Imputation	Linear Regression	N/A	1.771368
Iterative Imputation	Linear Regression	N/A	1.767101
Missing as -999	Decision Tree	2.231226	1.959689
Native Missing Support	HistGradientBoosting	2.628646	2.271743
Native Missing Support	CatBoost Standard	2.668006	2.306718
Native Missing Support	CatBoost AFT	2.792279	1.733899
Mean Imputation + AFT	CatBoost AFT on Imputed Data	2.544942	1.609733

Combining Mean Imputation with AFT clearly shows the best results

Results and Analysis from task [3.3]

- Using Mean Imputation + CatBoost AFT model:
    - Best Local cMSE: 1.6097
    - Best Kaggle score: 2.73477
- Despite achieving the lowest local cMSE , the Mean Imputation + CatBoost AFT model obtained a substantially higher error on the Kaggle test set. This gap indicates overfitting to the local validation data and limited generalization to unseen samples, likely due to sensitivity to distributional differences and missing data patterns.

Task [4.1] - Imputation with labeled and unlabeled data

What was done in task [4.1]

1. Data Preparation and Split	- Loaded full dataset and separated labeled (with SurvivalTime) and unlabeled (missing SurvivalTime) samples
	- Combined labeled and unlabeled feature matrices for semi-supervised learning
	- Preserved censoring indicators and target values for labeled data only
2. Semi-Supervised Imputation Testing	- Fitted imputers (Mean, KNN, Iterative) on combined labeled+unlabeled data
	- Extracted imputed labeled features for model training and evaluation
	- Compared imputation strategies using 5-fold cross-validation with Linear Regression
3. Isomap with Semi-Supervised Learning	- Selected best imputation method from comparison results
	- Fitted Isomap on combined scaled data (labeled + unlabeled) for dimensionality reduction
	- Tested multiple n_components values (2-8) using cross-validation to find optimal dimensionality

Results and Analysis from task [4.1]

Imputation Comparison (with Semi-Supervised Data)

Imputation	cMSE	Train MSE	Train cMSE
Mean	3.001359	2.817465	2.393221
kNN	3.013526	2.840490	2.422803
Iterative	3.005358	2.823820	2.398354

Best Imputation Model: **Mean**

Results and Analysis from task [4.1]

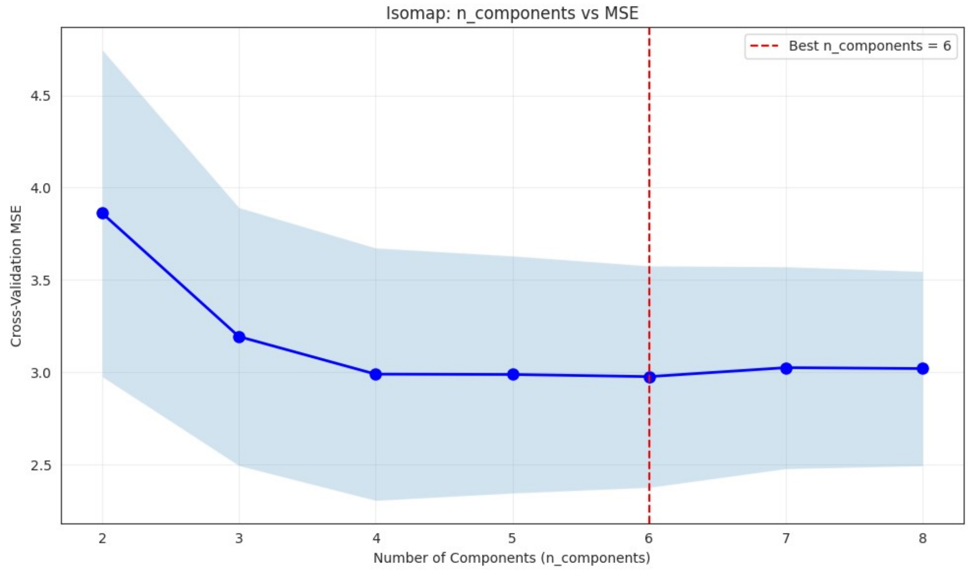
- Final Best Isomap Model:
    - n_componentes: 6
    - Training MSE: 2.8258
    - Training cMSE: 2.3901

Task [4.2] - Evaluation

What was done in task [4.2]

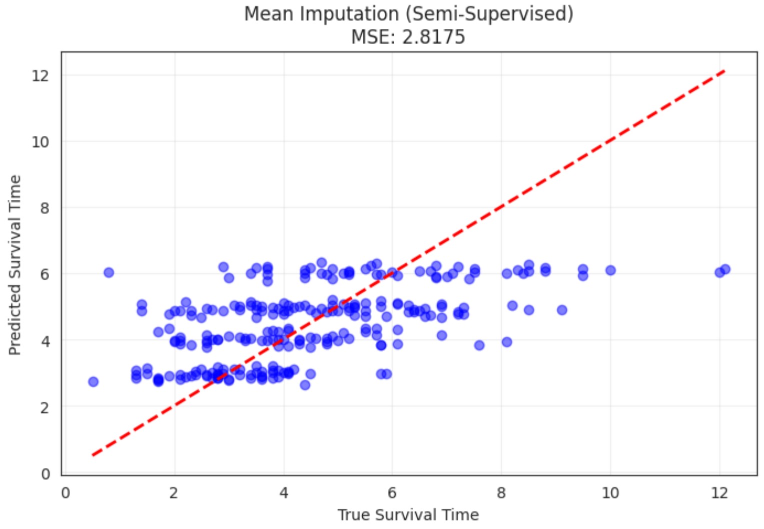
1. Isomap Performance Visualization	- Ploted n_components vs CV MSE with confidence intervals to identify optimal dimensionality
	- Marked best n_components with vertical line on the plot
	- Saved visualization showing performance trends across different component values
2. Model Comparison Plots	- Created side-by-side scatter plots comparing true vs predicted values
	- Visualized best imputation method (semi-supervised) vs best Isomap model
	- Included MSE metrics on each plot for direct performance comparison
3. Best Model Selection and Submission	- Compared cMSE between best imputation and Isomap models
	- Selected model with lowest cMSE for test predictions
	- Generated and saved Kaggle submission file with best model predictions

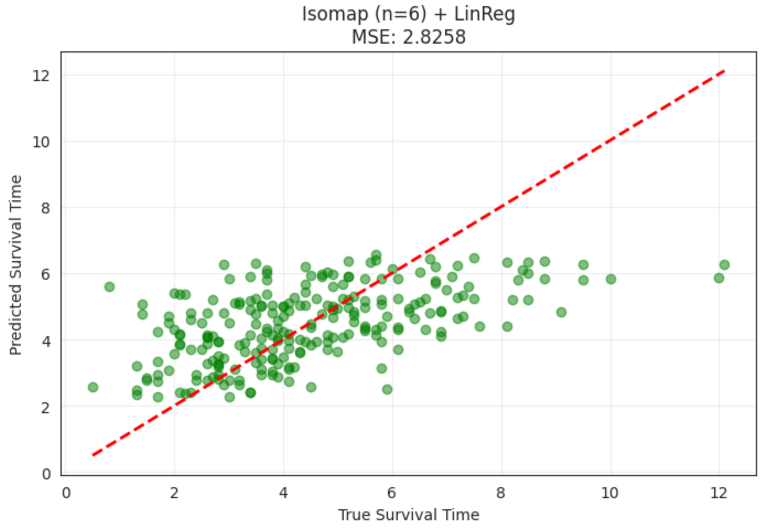
Results and Analysis from task [4.2]



The MSE decreases sharply from 2 to 4 components, indicating improved reconstruction.
From 4 to 8 components, the curve stabilizes, showing lessening returns.
Even though, the best MSE is 2.8258 with 6 componentes.

Results and Analysis from task [4.2]





Both plots show a very close performance in both techniques

Results and Analysis from task [4.2]

Strategy	Model	MSE	cMSE
Baseline (Drop Missing)	Linear Regression	4.087307	4.087307
Mean Imputation	Linear Regression	N/A	1.764469
KNN Imputation	Linear Regression	N/A	1.771368
Iterative Imputation	Linear Regression	N/A	1.767101
Missing as -999	Decision Tree	2.231226	1.959689
Native Missing Support	HistGradientBoosting	2.628646	2.271743
Native Missing Support	CatBoost Standard	2.668006	2.306718
Native Missing Support	CatBoost AFT	2.792279	1.733899
Mean Imputation + AFT	CatBoost AFT on Imputed Data	2.544942	1.609733
Isomap (with Semi-Supervised Data)	Linear Regression	2.8258	2.3901
Mean Imputation (with Semi-Supervised Data)	Linear Regression	2.8174	2.3932

Results and Analysis from task [4.2]

- Using Isomap model:
  - Best Local cMSE: 2.3901
  - Best Kaggle score: 2.58777

Code Demo

Overall assessment

What went wrong

Due to the heavy academic workload of all group members, we were not able to dedicate as much time to the project during the early stages as we had initially planned.

As a result, we started the Kaggle submission phase later than expected. This delay became a limitation because Kaggle only allows five submissions per day, which significantly restricted our ability to iterate, test different modeling strategies, and optimize our final model.

With more time and more allowed submissions, we would likely have explored additional approaches and achieved more refined results.

What went great

Even though our team faced time constraints, several aspects of the project went very well. We successfully implemented, tuned, and evaluated multiple machine learning models, applied appropriate handling for missing data, and correctly dealt with censored targets using suitable metrics.

Collaboration within the group was effective, with each member contributing to the modeling pipeline, validation strategy, and analysis of results.

Importantly, even though our public leaderboard results were not very strong, the model performed noticeably better on the private score.