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]

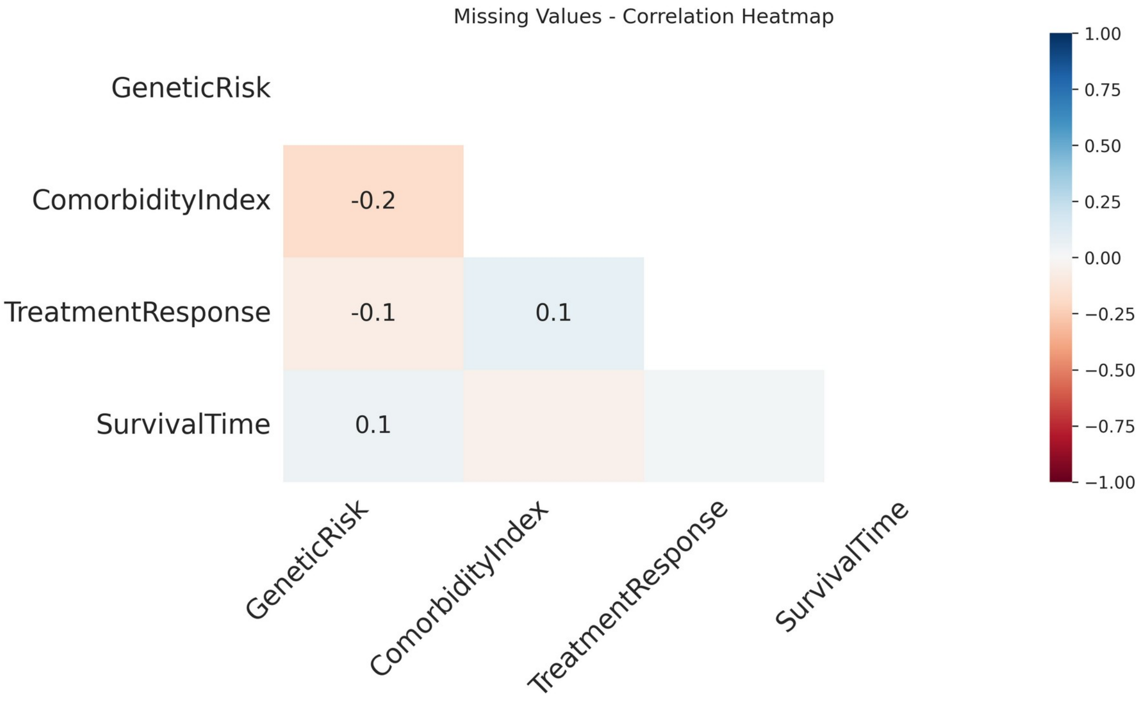
1. Missing Values Analysis	Visualized missing values using multiple methods (bar plot, heatmap, matrix, dendrogram)
	Created comprehensive overview of data completeness
	Identified patterns in missing data
2. Data Cleaning	Dropped rows with missing SurvivalTime values
	Removed columns containing missing 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]

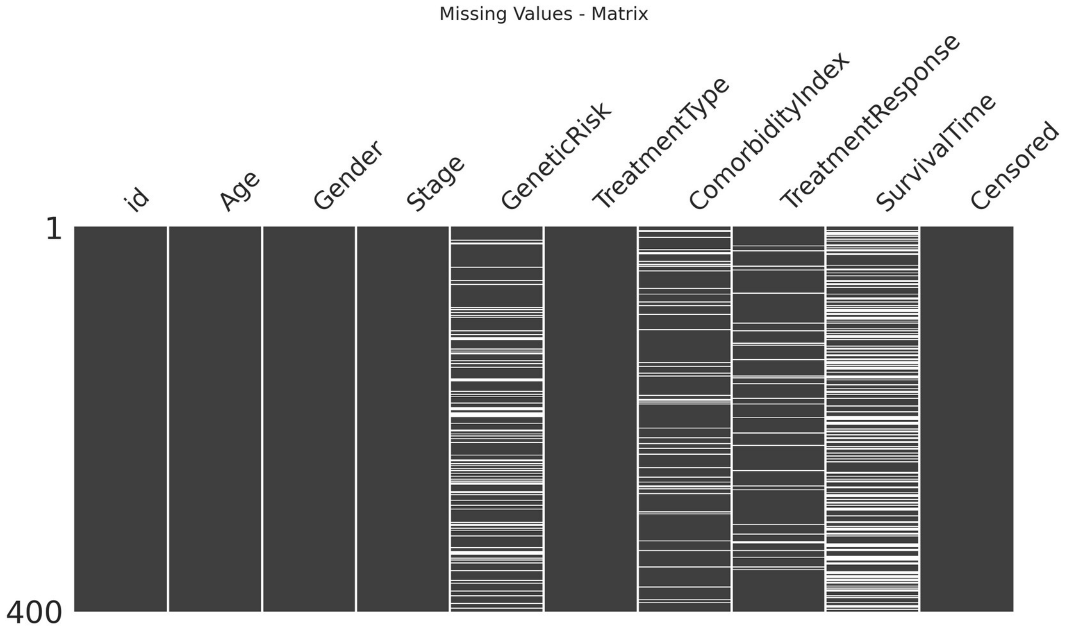
4. Data Preparation	Defined feature matrix (X) by dropping target and identifier columns
	Isolated target variable (y) as SurvivalTime
	Preserved censoring indicator for potential future use
5. Validation Strategy Development	Implemented train/validation/test split (64%/16%/20%)
	Tested simple split 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]

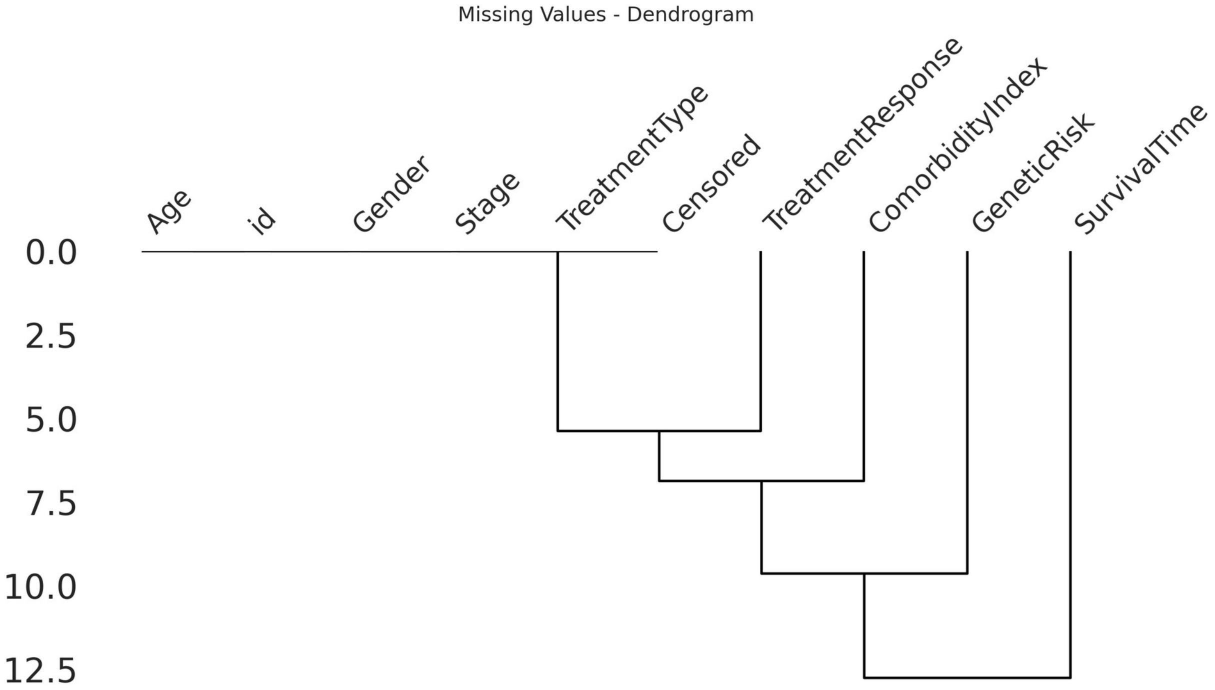
Results and Analysis from task [1.1]



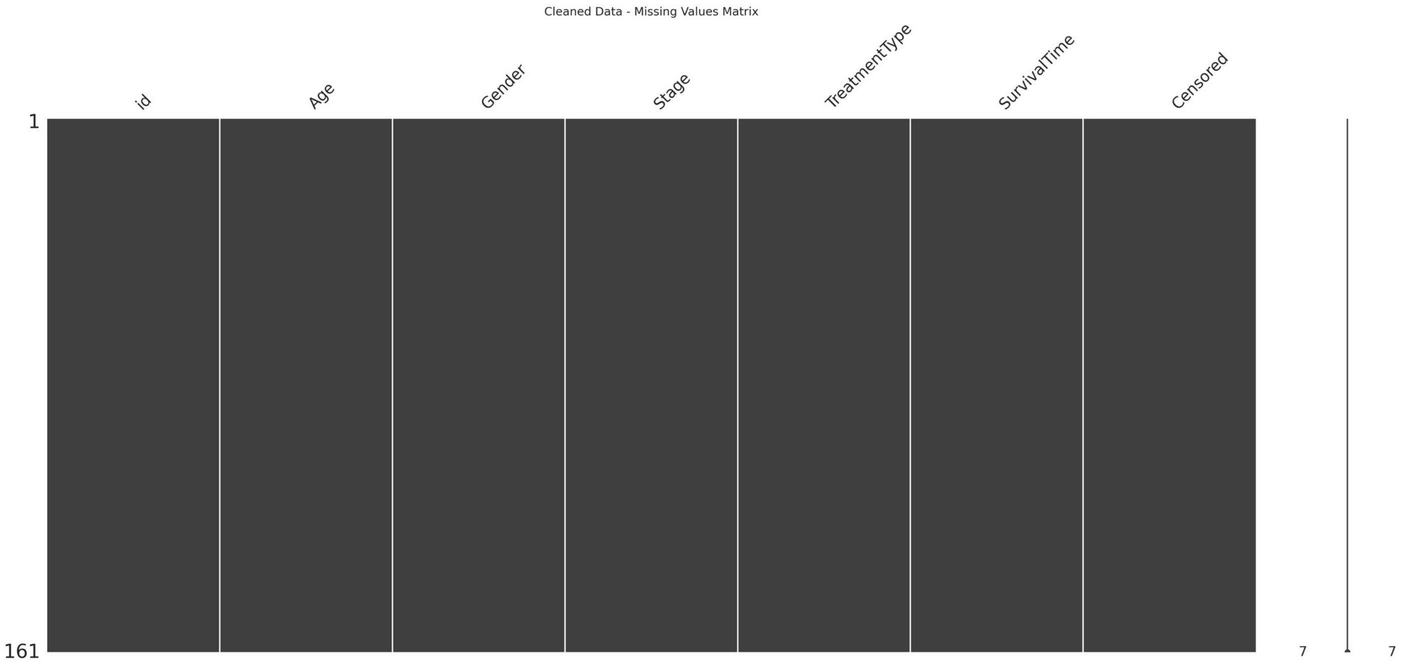
Results and Analysis from task [1.1]



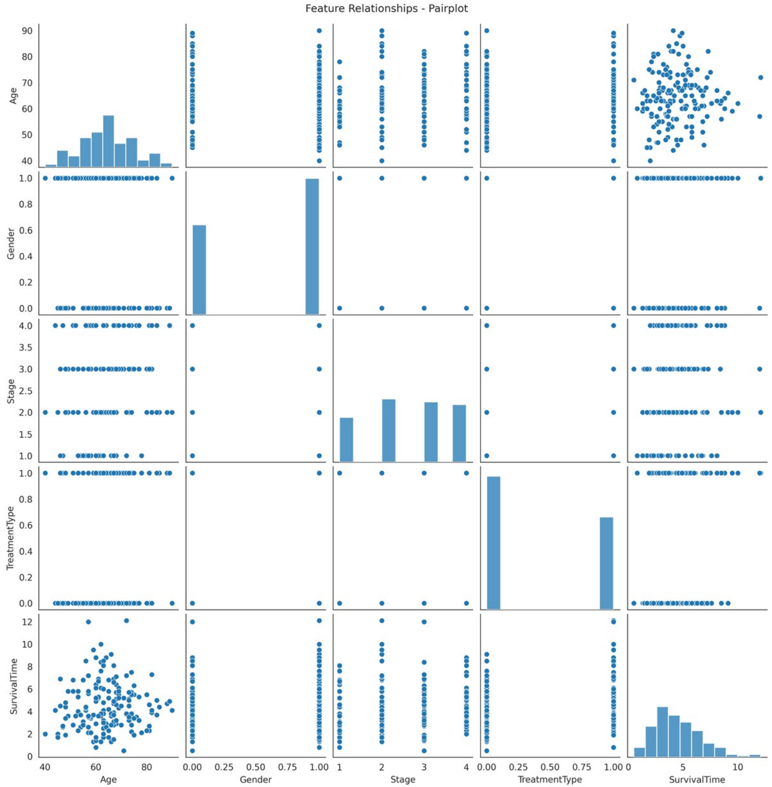
Results and Analysis from task [1.1]



Results and Analysis from task [1.1]



Results and Analysis from task [1.1]



Task [1.2] - Learn the baseline model

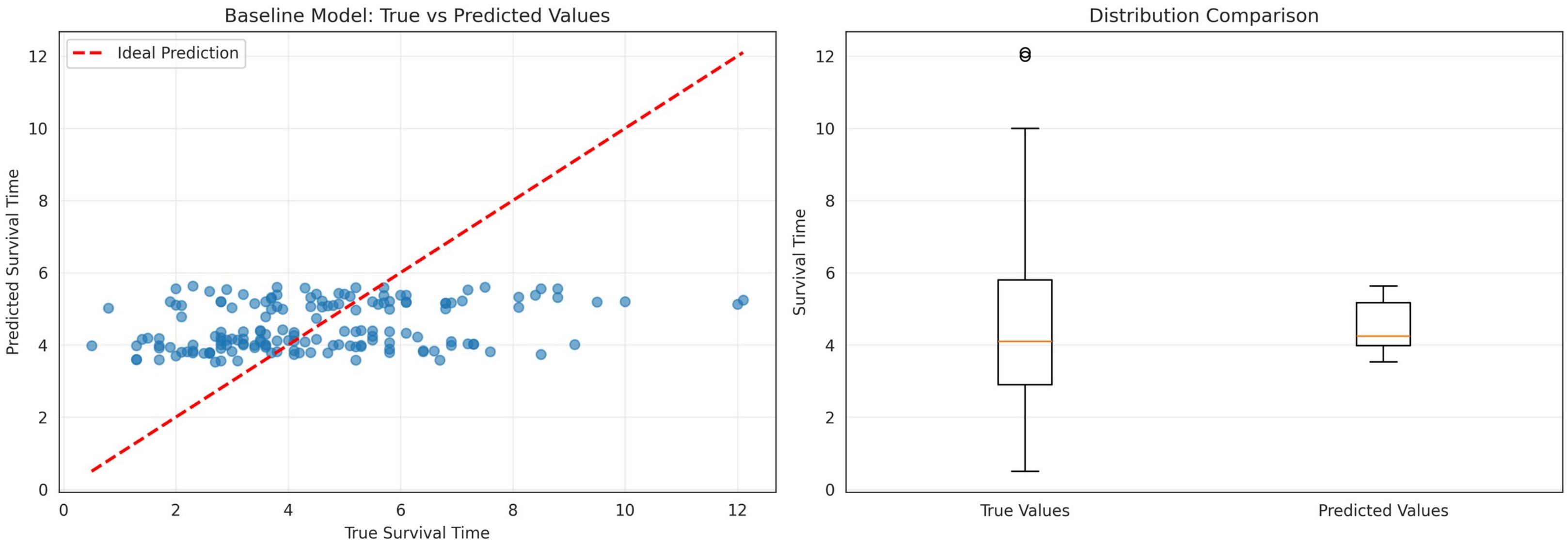
What was done in task [1.2]

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]



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]

Task [2.2] - Evaluation

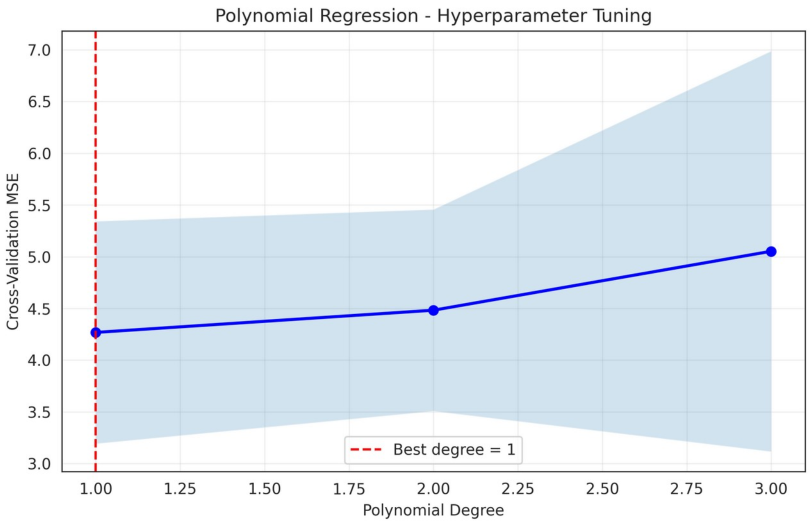
What was done in task [2.2]

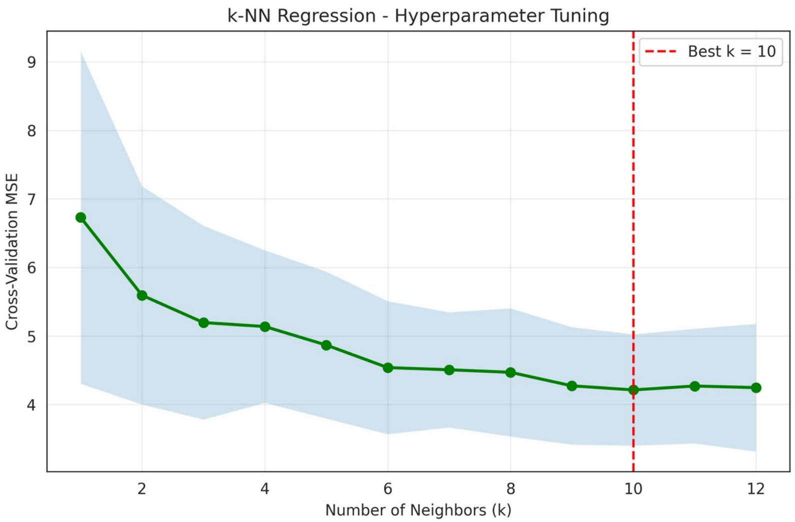
1. Comprehensive Model Comparison	- Created comparison table with baseline, polynomial regression, and k-NN models
	- Included hyperparameter configurations for each model
	- Displayed min, max, mean, and standard deviation of errors
	- Identified best performing model based on mean cross-validation 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]

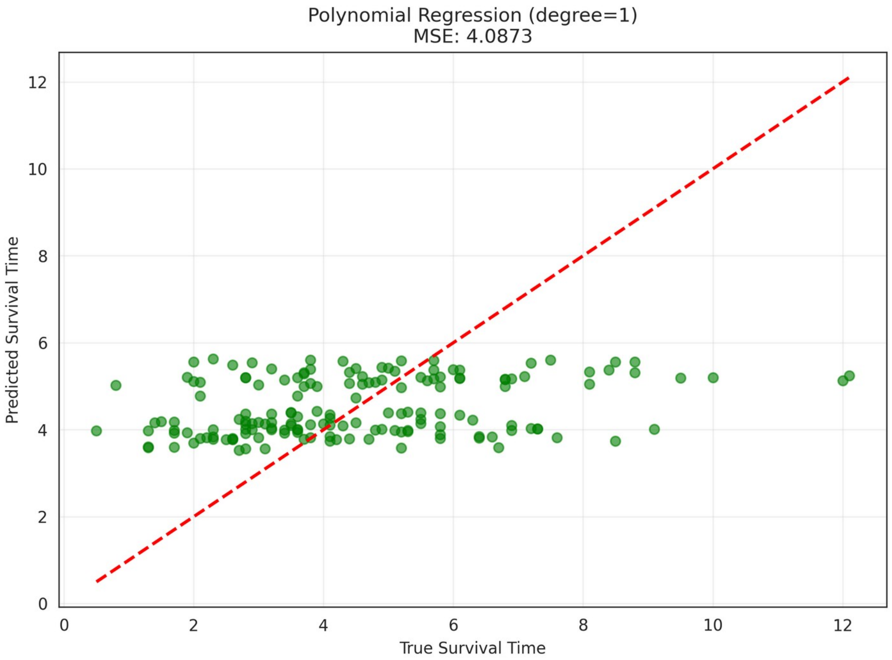
4. Statistical Analysis	- Computed cross-validation statistics for each model
	- Analyzed variance in predictions across folds
	- Compared model stability through standard deviation metrics
	- Evaluated improvement over baseline model
5. Test Set Predictions	- Selected best performing model based on 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]

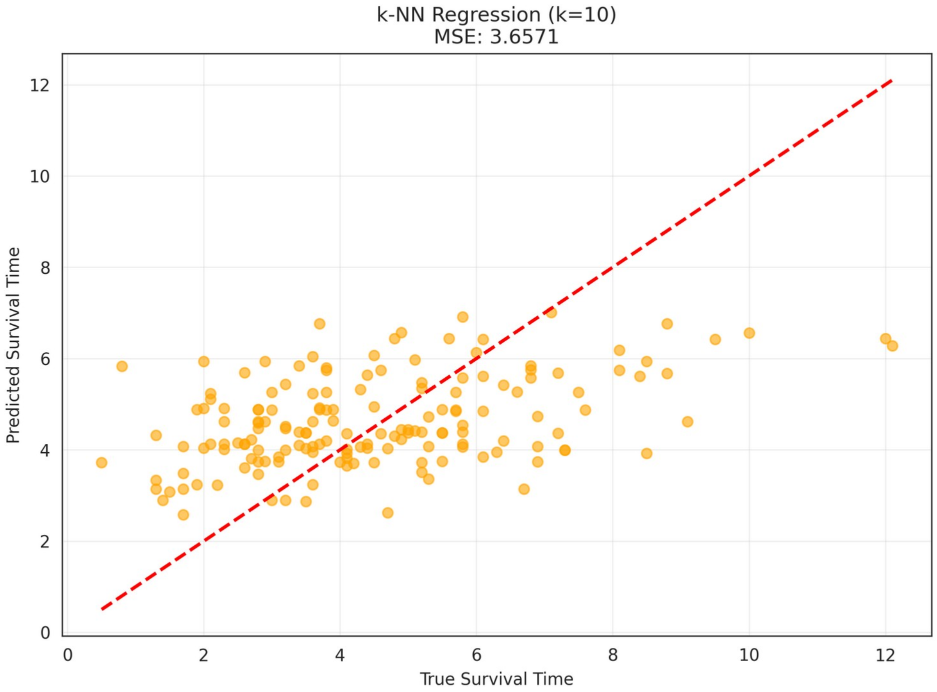




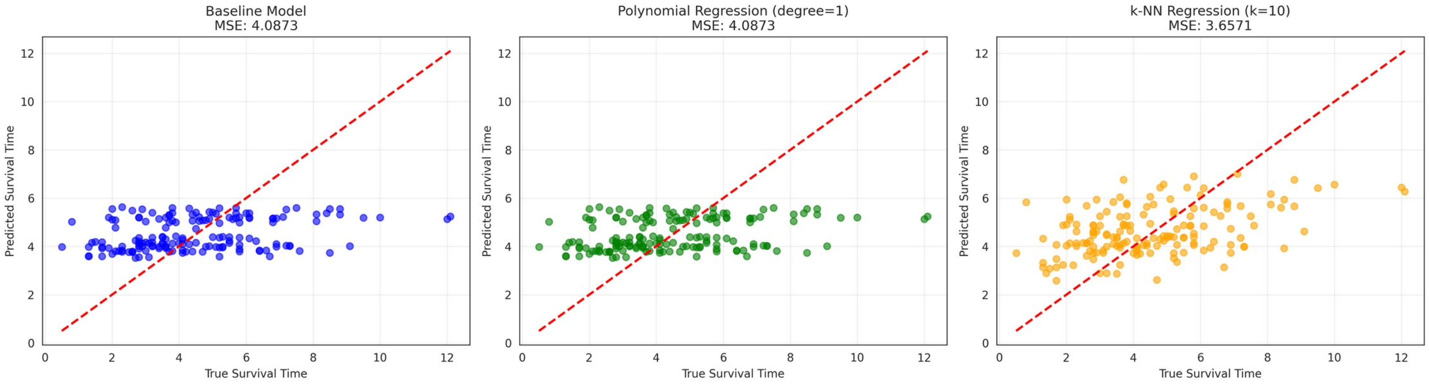
Results and Analysis from task [2.2]



Results and Analysis from task [2.2]



Results and Analysis from task [2.2]

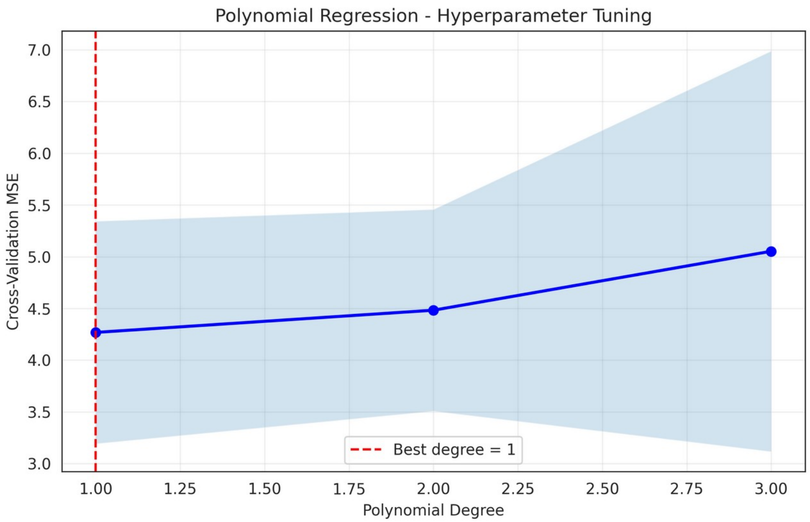


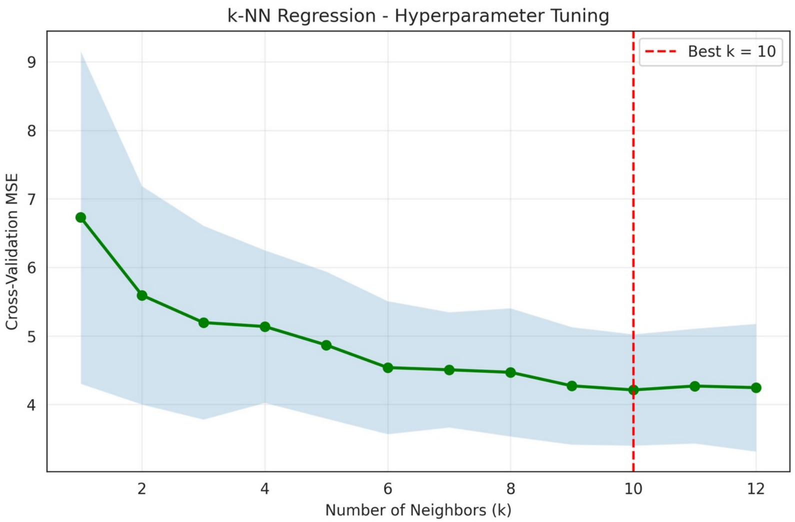
Task [3.1] - Missing data imputation

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]





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

What was done in task [3.2]

Results and Analysis from task [3.2]

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]

Code Demo

Overall assessment

What went wrong

What went great