ML@NOVA: PRS

Survival Time in Multiple Myeloma Patients

Team identification

Name 1: Pedro Peralta

Number 1: 70070

Name 2: Rodrigo Maravilhas

Number 2: 60619

Name 3: Simão Carrasco

Number 3: 59208

Final score:3.44494

Leaderboard private ranking: 47º

Task [1]

Setting the baseline model

Task [1.1]

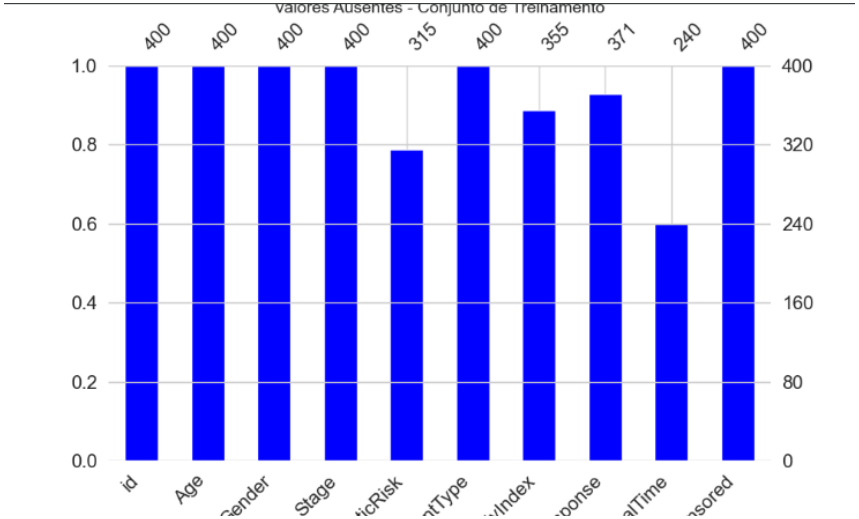
Data Preparation and Validation Pipeline

What was done in task [1.1]

- In this initial task, we focused on analyzing and preparing the dataset to ensure it could be effectively used in the machine learning models developed in subsequent steps. The dataset comprises 400 entries, but some columns contained null values that required thorough investigation. We performed a detailed analysis of these nulls to decide on appropriate handling methods, such as imputation or removal, depending on their impact on the overall dataset.
- Additionally, we employed various visualization tools to better understand the dataset's behavior and distribution patterns. These visualizations helped identify trends, detect potential outliers, and uncover correlations between variables, providing valuable insights for the modeling phase. The preparation process ensured the dataset was clean, consistent, and ready for further analysis.

What was done in task [1.1]

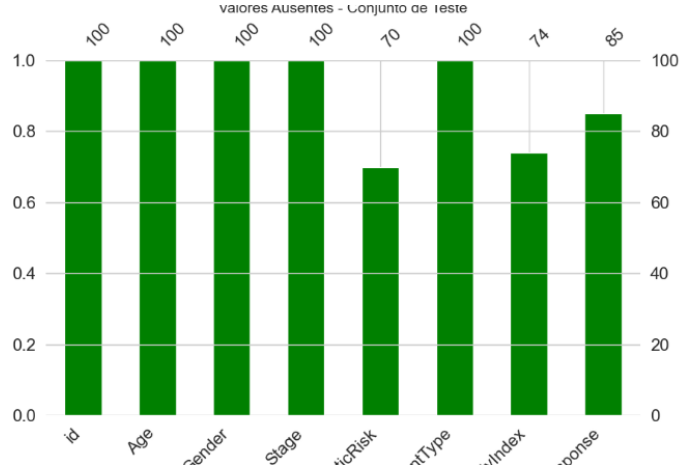
- The bar chart provides a clear visualization of the entries within the training dataset, helping us identify patterns in the data. To gain a more precise understanding of the number of null values present in each feature, we also analyzed the dataset using a detailed table.
- This approach allowed us to quantify the missing data accurately and informed our decisions on how to handle these null values, ensuring the dataset's quality and reliability for subsequent modeling steps.



Feature	Null Qnt
GeneticRisk	85
ComorbidityIndex	45
TreatmentResponse	29
SurvivalTime	160

What was done in task [1.1]

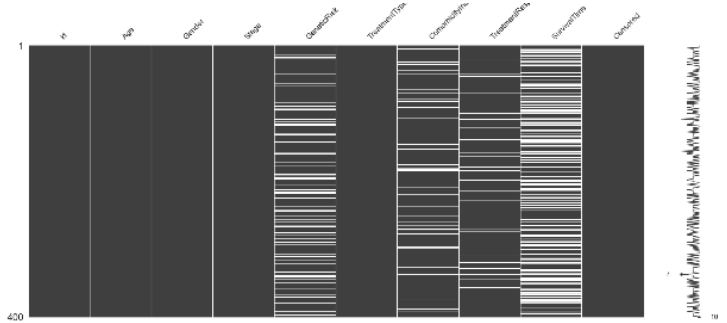
- The features SurvivalTime and Censored were not included in the training data. Other features with null values kept their missing entries.
- We reviewed these features to understand the missing data and plan how to handle it for better model performance.



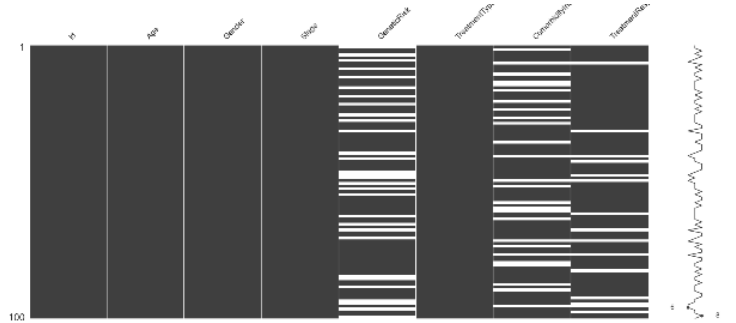
Feature	Null Qnt
GeneticRisk	30
ComorbidityIndex	26
TreatmentResponse	15

What was done in task [1.1]

- Using the msno.matrix() chart, we analyzed the distribution of null values in the dataset. Excluding SurvivalTime, we observed that the GeneticRisk feature contains many missing values.
- Due to the high percentage of nulls, we may consider removing GeneticRisk from the training dataset to improve model performance.



Training Dataset



Test Dataset

What was done in task [1.1]

Features with High Positive Correlation to SurvivalTime:

- TreatmentResponse (0.48):
  - This indicates that SurvivalTime increases as TreatmentResponse improves. It suggests a strong relationship between how well a patient responds to treatment and their survival duration.
- Age (0.42):
  - Older patients tend to have slightly higher survival times, as seen from the moderate positive correlation.
- ComorbidityIndex (0.27):
  - Patients with higher comorbidities seem to have longer survival times, though this relationship is weaker compared to the first two features.

Features with Low or Negligible Correlation to SurvivalTime:

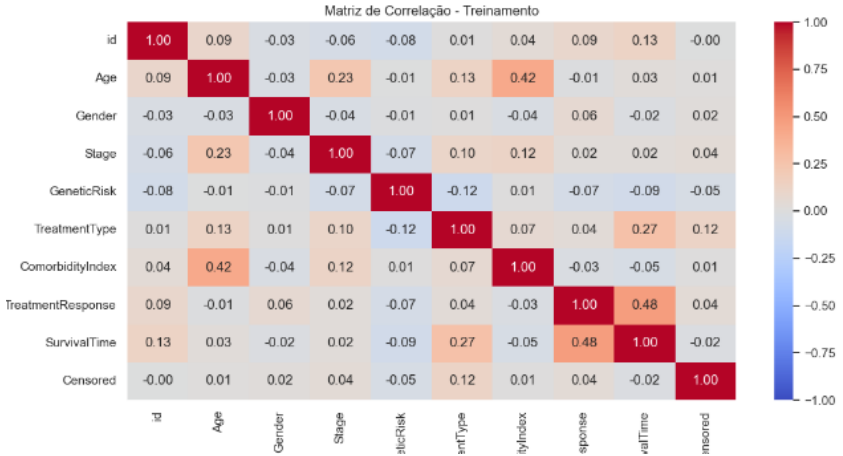
- Gender (-0.09):
  - Gender appears to have almost no impact on SurvivalTime, indicating that survival is not influenced by gender differences in this dataset.
- Stage (-0.12):
  - The correlation is slightly negative but weak, suggesting that the stage of the disease might not strongly determine survival time.

Other Relationships:

- Stage and ComorbidityIndex (0.42):
  - A moderate positive correlation suggests that higher stages of disease are associated with higher comorbidity indices.
- TreatmentResponse and ComorbidityIndex (0.27):
  - Indicates that treatment response slightly depends on the comorbidity index.

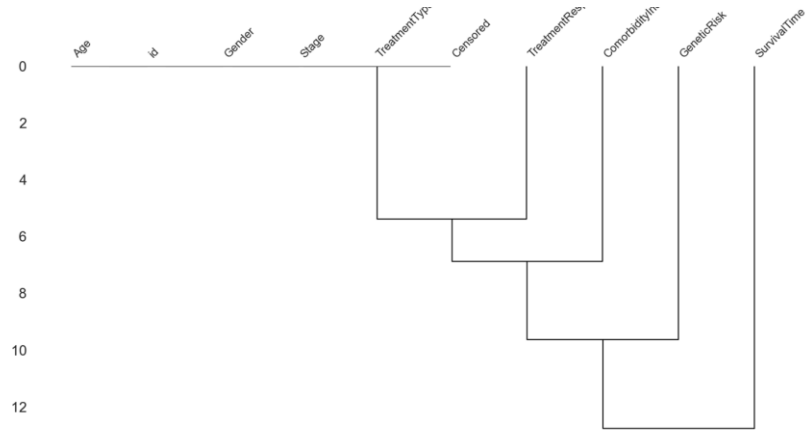
Weak Correlations with Censored:

- Most features show negligible correlation with the Censored column, which means the censoring status does not strongly relate to other features in the dataset.



What was done in task [1.1]

- TreatmentType and Censored:
  - These two features are clustered together at a very low linkage distance. This suggests that they share a similar pattern or statistical relationship, potentially due to the way treatments might influence censoring status.
- ComorbidityIndex, GeneticRisk, and SurvivalTime:
  - These features form a cluster, which implies a relationship between these factors and survival time. ComorbidityIndex and GeneticRisk might jointly contribute to predicting survival outcomes.
- Age and id:
  - These features are grouped but at a slightly higher linkage distance, indicating a weaker relationship. The id is unlikely to carry meaningful predictive power but may have coincidental relationships.
- Gender and Stage:
  - These features remain separate from the other clusters until a much higher linkage distance. This indicates they may not share strong statistical similarities with the other features or with each other.
- Survival-Related Features:
  - SurvivalTime is closely clustered with ComorbidityIndex and GeneticRisk, confirming the insight from the correlation heatmap that these features are key predictors of survival time.



Task [1.2]

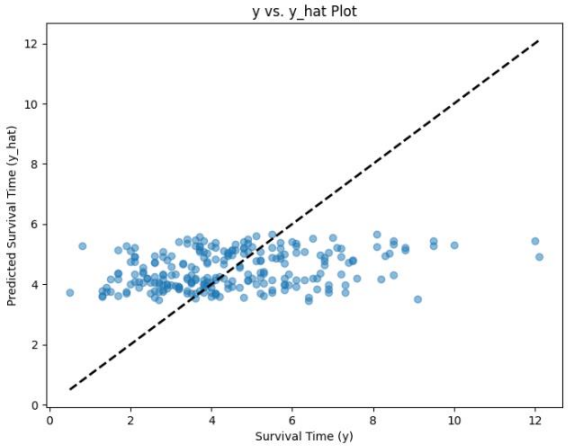
Learn the baseline model

What was done in task [1.2]

- The first step in this task was to remove all features with null values that would not be used to train our baseline model. We chose Linear Regression for the baseline, which does not support null values.
- After identifying the features with missing data during our initial analysis, this process was straightforward. The performance of our baseline model was evaluated using the Mean Squared Error (MSE) as the key metric.

[1.2] Y-Y hat plot analysis

- The Y-Y hat plot shows that most Survival Time points are centered along the line, though some dispersion is present.
- Comparing the Kaggle score with the local score revealed minimal discrepancy, as shown in the table. The difference, approximately 0.5, remained consistent throughout the project.



Location	MSE Score
Local code run	3.682
Kaggle submission	4.26352

Task [1.3]

Learn with the cMSE

What was done in task [1.3]

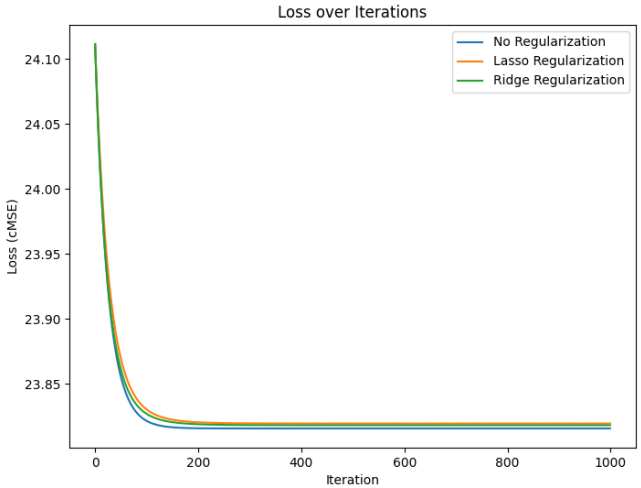
- In this task, we created a function to calculate the derivative of the cMSE loss. To achieve this, we first outlined the process and determined the necessary calculations for the function to work correctly.
- The cMSE formula was provided by the professor in the task description. Our focus was to implement a derivative function that could be used effectively in the Gradient Descent optimization process.

What was done in task [1.3]

- Escrever aqui quais os cálculos matemáticos que utilizamos….

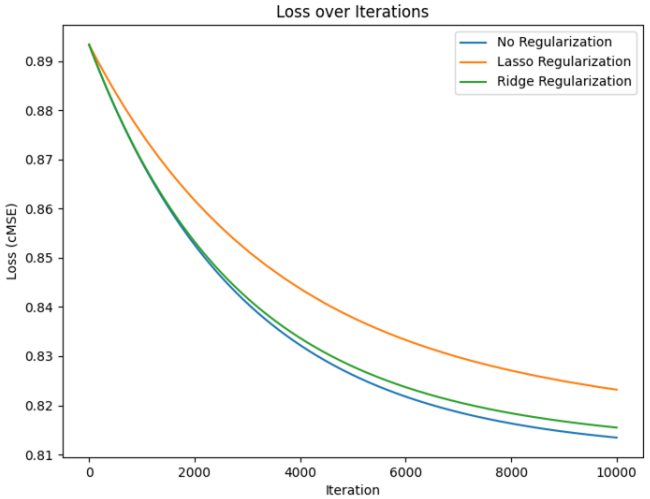
What was done in task [1.3]

- After completing the functions, we tested gradient descent with regularization using Lasso and Ridge to determine which would provide the best performance for our model.
- We encountered an issue initially, as the predicted values were inconsistent. Upon analyzing the graph, we noticed something unusual, which helped us identify the problem in our approach.



What was done in task [1.3]

- We only noticed something was wrong when we submitted the model to Kaggle and received a score of 29. After investigating, we discovered that the issue was due to our target variable Y not being on the same scale or normalized.
- Once we normalized Y and applied the necessary scaling, the graph improved, showing correct predictions. As seen in the graph, neither Lasso nor Ridge regularization improved the model's performance. Therefore, we decided to remove the regularizations and proceed without them.



What was done in task [1.3]

- We analyzed the scores with different types of regularization to complement our previous analysis, in addition to the graphs. We found that both Lasso and Ridge regularizations did not provide any benefit to the model's performance.
- As a result, we submitted the model without regularization to Kaggle. The score difference remained consistent, with a 0.5 difference, which is the same result we've been getting throughout the project.

Regularization	MSE Score
No Regularization	3,290
Lasso	3,330
Ridge	3,298

Best Local Score	Kaggle Score
3,290	3,71431

Task [2]

Nonlinear model

Task [2.1]

Development

What was done in task [2.1]

- In this task, we aimed to improve our model by using non-linear models. We tested two models: Polynomial Regression and K-Nearest Neighbors (KNN). Two pipelines were created to test each model with different degrees for the polynomial and different values for K in KNN, evaluating their performance using MSE.
- After implementing and analyzing the performance of these models, we decided to implement cross-validation to further improve the model's training and ensure more reliable results.

Performance Analysis [2.2]

- As shown in the table below, the non-linear models did not improve the performance of the model. Therefore, we decided to use the linear model for future analyses, as it provided the most consistent and reliable results.

Model	Local Score	Kaggle Score
Linear	3.816	-
Polinomial	4.262	4.50813
Knn	4.301	4.68458

Task [3]

Handling missing data

Task [3.1]

Missing data imputation

What was done in task [3.1]

- At this stage of the project, we decided to reintroduce the features that were initially removed due to missing values. To do this, we applied data imputation techniques to fill in the missing values with approximate values, aiming to improve the model's training.
- Before starting the imputation process, we analyzed the distribution of values in the features TreatmentResponse, ComorbidityIndex, and GeneticRisk. There are several ways to impute missing data, ranging from simple methods like using the mean, median, or a constant value.
- Since the features with missing values are all discrete, using the mean is not suitable. We considered alternative approaches, which we will now analyze further.

Features Analysis - TreatmentResponse [3.1]

- In the TreatmentResponse feature, there is a balanced distribution between the two response types, with only a few missing values. It wouldn’t be fair to assign all the null values to a single type. Instead, we decided to balance the missing values by imputing 9 records with a value of 1 and 20 records with a value of 0, based on the existing distribution. This approach helps maintain the feature's balance while filling in the missing data.

Value	Quantity
0	180
1	191
NA	29

Features Analysis - ComorbidityIndex [3.1]

- The ComorbidityIndex feature contains discrete values with a range, making the imputation of missing values more complex. The chosen imputed value can have a significant impact on the model's training, either positively or negatively.
- For basic imputation approaches, we could use the mode or median to fill in the missing values. However, we need to carefully consider the potential effects of these methods on the overall performance of the model, given the feature's variability.

Value	Quantity
0	96
1	153
2	83
3	23
NA	45

Features Analysis - GeneticRisk [3.1]

- The GeneticRisk feature has the same number of possible values as TreatmentResponse, but with a distribution more heavily skewed towards the value 0. This means that the imputation technique we use could impact the model's performance, depending on which value we assign to the missing data.
- Given this distribution, we must carefully consider the imputation strategy to maintain the feature's overall balance and avoid introducing bias into the model.

Value	Quantity
0	225
1	90
NA	85

Advanced Data Imputation [3.1]

- In addition to simpler approaches like mean, median, and mode, there are more advanced imputation techniques. One such technique is the KNN Imputer, which fills in missing values based on the closest neighbors. We believe this is a great approach as it helps normalize the data further by considering the relationships between entries.
- Another technique we explored is Iterative Imputation, which uses round-robin linear regression to model the missing values of a feature based on other features. This is also a promising approach, as it takes into account the correlations between features.
- Both of these techniques were mentioned in the resources provided in the task description, and they are the ones we will focus on for this task.

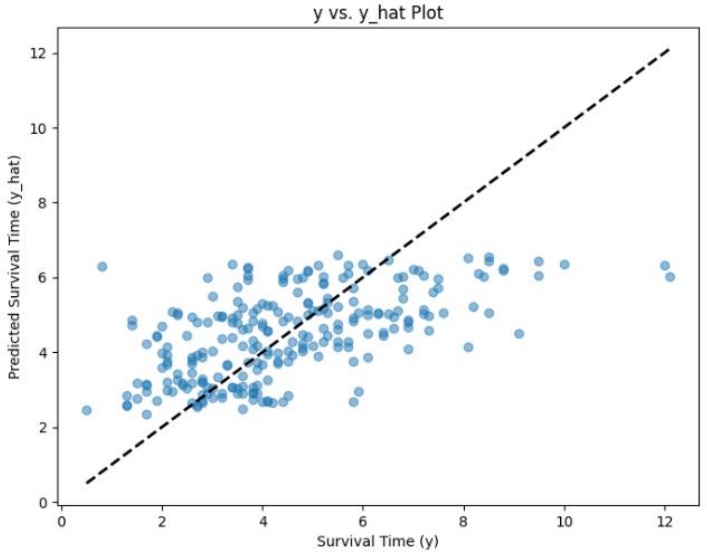
Knn Imputation [3.1]

- By implementing these imputation techniques, we achieved very positive results. We tested various values of K to find the optimal one for the best performance.
- After evaluating the results, we decided to submit the model to Kaggle using K = 2, as this gave us the best performance with the chosen imputation method.

K Value	Score
1	2,760
2	2,710
3	2,720
5	2,732
Kaggle K = 2	3,44623

Knn Imputation – Y-Y hat plot [3.1]

- When visualizing the Y-Y hat plot, we observed a greater concentration of points along the diagonal. This indicates that the model is better fitted to the dataset, suggesting it will be able to make more accurate predictions moving forward.



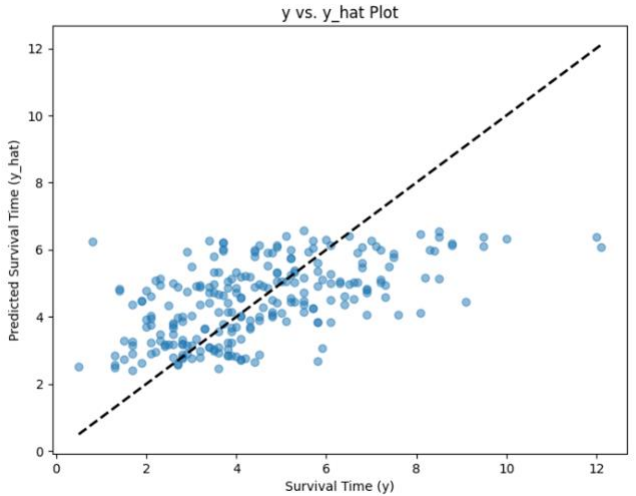
Iterative Imputation [3.1]

- Using the Iterative Imputation technique, we observed that although we achieved a better score than the model without the features, we could not surpass the performance of the KNN Imputation. We experimented with various numbers of iterations, even going to extremes to see how it would affect the model's performance, but no improvements were observed.

Iteration Number	Score
30	2,763
50	2,763
Kaggle	3,52075

Knn Imputation – Y-Y hat plot [3.1]

- When visualizing the Y-Y hat plot, we continued to see a greater concentration of points along the diagonal, indicating an improvement compared to the model without the features. The plot looks very similar to the one from KNN Imputation. However, upon analyzing the scores, we found that KNN Imputation still produces better predictions overall.



Task [3.2]

Train models that do not require imputation

What was done in task [3.2]

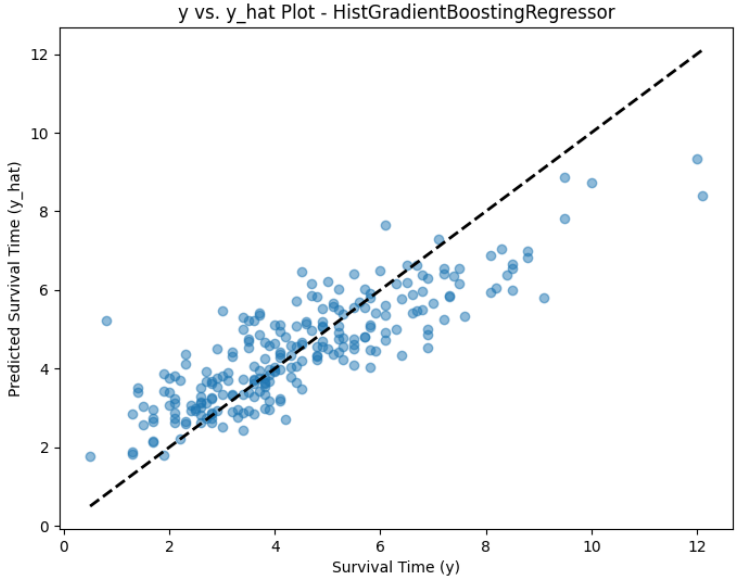
- In this task, we implemented and analyzed models and techniques capable of handling missing data directly, as suggested by the teacher. The objective was to explore approaches that inherently manage incomplete datasets without the need for preprocessing steps such as imputation.
- The teacher recommended two key approaches for this task: the HistGradientBoostingRegressor from Scikit-Learn and the CatBoostRegressor, which provides specialized support for survival analysis. Additionally, we developed a strategy using decision trees to adapt linear, polynomial, and k-NN models to work without explicit imputation, allowing us to extend traditional models to incomplete datasets.

HistGradientBoostingRegressor [3.2]

- The HistGradientBoostingRegressor was one of the models recommended by the teacher. This tree-based ensemble method is inherently capable of handling missing data by treating missing values as a distinct category during the tree construction process.

HistGradientBoostingRegressor Y-Y hat plot [3.2]

- The scatter plot for the HistGradientBoostingRegressor shows a noticeable variance around the diagonal, particularly for higher survival times, suggesting prediction inaccuracies for longer survival times.
- A significant number of points fall far from the diagonal, especially above the 6-8 survival time range, indicating over-predictions for certain instances.



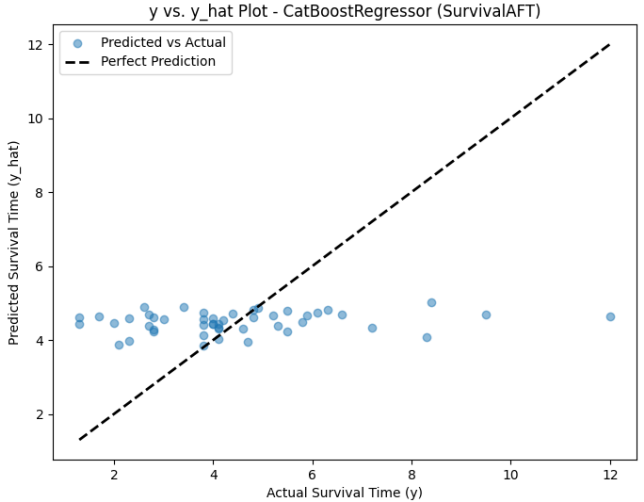
Local Score
3.557

CatBoostRegressor [3.2]

- The second model provided for this task was the CatBoostRegressor. CatBoost natively handles missing values by treating them as a separate state during training. Additionally, its support for the Accelerated Failure Time (AFT) loss function made it particularly suitable for survival analysis.
- Key steps:
  - The target variable was transformed into an interval representation to handle censored data effectively.
  - The SurvivalAft loss function was applied, and experiments were conducted with various distributions (Normal, Logistic, and Extreme) to determine the best fit.

CatBoostRegressor Y-Y hat plot [3.2]

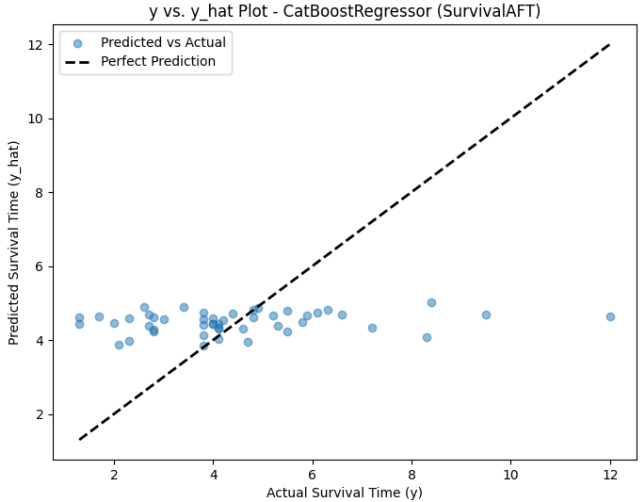
- The y-y hat plot shows that most predictions for survival times below 6 years align reasonably well with the diagonal, indicating good accuracy in this range.
- Predictions for actual survival times beyond 6 years exhibit significant underestimation, as points cluster below the diagonal.



Local Score
4.239

CatBoostRegressor Y-Y hat plot [3.2]

- This was the best rmse we could find after tinkering with the specifications of the model:
- Iterations: 2000, allowed the model to stabilize and converge effectively.
- Learning Rate: 0.001, ensured incremental updates during training, but may have limited the model's flexibility.
- Depth: 4, constrained the model’s ability to capture complex interactions, particularly for longer survival times.
- Loss Function: SurvivalAft:dist=Extreme, optimized predictions based on the Extreme Value distribution, suitable for skewed data but possibly contributing to the bias observed.



Local Score
4.239

Decision Tree Models [3.2]

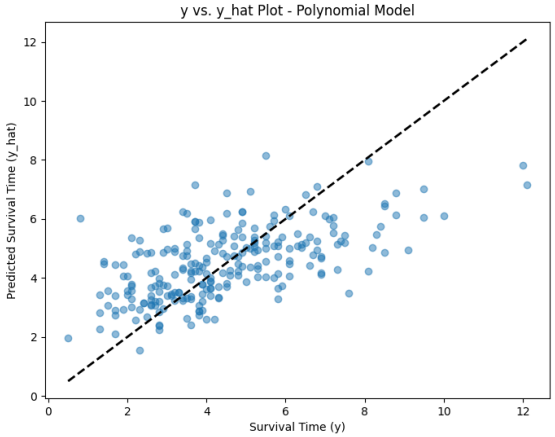
- To make models such as Linear Regression, Polynomial Regression, and k-NN work without imputation, a novel approach using Decision Trees was implemented. The decision tree imputation technique leverages the predictive power of DecisionTreeRegressor to estimate missing feature values directly from the relationships observed in the rest of the data.
- A DecisionTreeRegressor was trained for each feature containing missing values, using the remaining features as predictors.
- Missing values were then filled using predictions from the corresponding trained decision tree.
- This process was applied independently to both the training and test datasets.

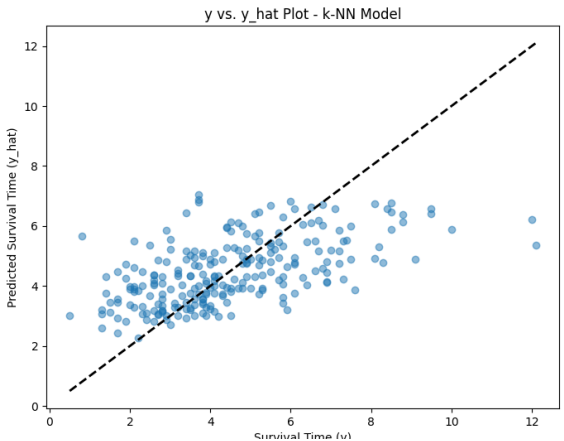
Decision Tree Models Analysis [3.2]

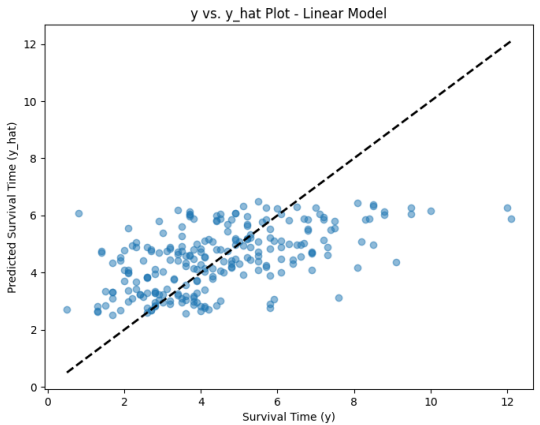
- The Kaggle scores for both Linear Regression and k-NN are marginally worse than their local counterparts. This discrepancy is expected due to potential differences in data characteristics, such as unseen patterns or noise in the Kaggle test set.
- The polynomial model's poor local performance (RMSE 4.173) suggested overfitting to the training data, making it an unsuitable candidate for submission.

Model	Local Score	Kaggle Score
Linear	3.023	3.555
Polynomial	4.173	N/A
k-NN	3.286	3.483

Decision Tree Models Y-Y hat plot [3.2]







Decision Tree Models Y-Y hat plot [3.2]

- Accuracy:
  - The k-NN model shows the best alignment with the "perfect prediction" line, indicating higher accuracy.
  - The linear model also performs well but exhibits slightly more scatter compared to k-NN.
  - The polynomial model demonstrates the weakest performance, with significant deviations, particularly for higher survival times.
- Generalization:
  - The k-NN and linear models generalize better compared to the polynomial model, which overfits for small survival times and fails to predict larger values accurately.
- Complexity:
  - The polynomial model is the most complex but underperforms due to overfitting.
  - The linear model, being the simplest, provides stable but less flexible predictions.
  - The k-NN model strikes a balance between complexity and accuracy.

Task [4]

Semi-supervised learning for unlabeled data

Task [4.1]

Imputation with labeled and unlabeled data

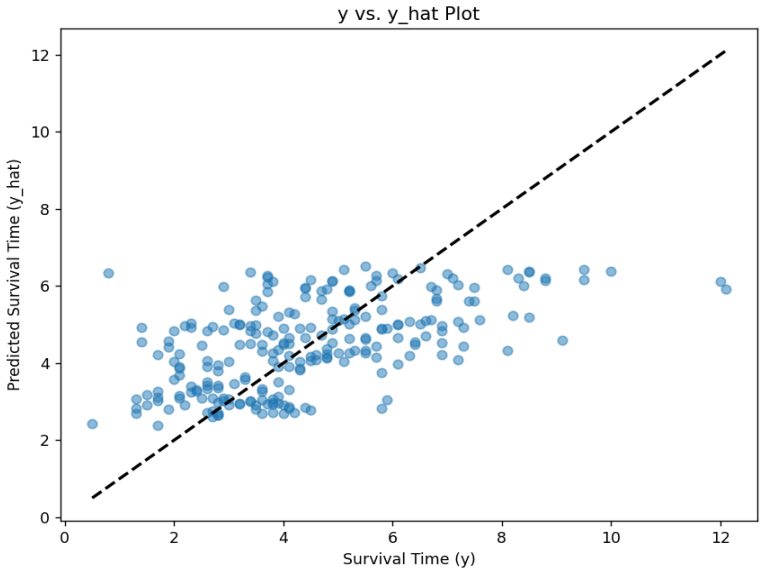
What was done in task [4.1]

- The best inputer from task 3.1 (knn inputer, n=2) was now fitted using both the data with values of the SurvivalTime feature, and with missing values (labeled and unlabeled data).
- After inputting missing features, only the labeled rows were used to train a linear regression model
- After that, again the combined labeled and unlabeled data was used but this time to train an isomap lower dimensional model of the data.
- Using an isomap we tested reducing the data to various smaller dimensions.

Analysis [4.2]

Using the knn inputer fitted with labeled and unlabeled data resulted in a slight improvement in terms of RMSE.

Observing the y-y hat plot, we can also observe a small improvement in the sense that the values are more concentrated near the diagonal



Analysis [4.2]

3.2	4.1 knn inputer using all data	4.1 isomap
3.557	2.749	3.228

We weren’t able to get good results when using an isomap, no matter the number of components tested.

Overall assessment

What went wrong

We didn't manage to get as much improvement in our model evaluation scores as we'd hoped, we just had a big jump in the score at one point but then we couldn't get it down from 3. Our goal was to get a score below 3, which with all the methods we tried we couldn't achieve.

What went great

In the beginning, I think we got off to a good start with the whole development of the project, starting with the analysis of the features and checking what their values and distribution were. Before we started developing the algorithms, we did a lot of analysis to understand the problem from the outset, which made the process easier.