The University of Azad Jammu and Kashmir, Muzaffarabad

Bachelor of Science of Software Engineering (2022-2026)

Department of Software Engineering

Assignment #01

Survival Analysis of Transplant Patients: A Data-Driven Approach

Submitted

By

AYESHA SHAFAQAT (2022-SE-03)

Lab Instructor: Engr.Ahmed Khawaja

Subject: Machine Learning (SE – 3105)

Semester: 5th

Session: 2022 – 26

Table of Contents

1. Abstract

Allogeneic Hematopoietic Cell Transplantation (HCT) is a life-saving procedure used to treat various hematologic disorders by replacing defective immune cells with healthy donor-derived hematopoietic stem cells. However, disparities in survival predictions related to socioeconomic status, race, and geography raise significant concerns regarding equity in healthcare.

This report focuses on performing exploratory data **analysis (EDA)** on a synthetic dataset of patients undergoing **allogeneic Hematopoietic Cell Transplantation (HCT).** The goal is to preprocess the data, handle missing values, identify key insights, and visualize trends related to patient survival and demographic factors.

Various statistical techniques, including distribution analysis, correlation studies, and survival curve visualizations, are employed to understand the underlying patterns in the dataset. The findings from this EDA phase will serve as a foundation for building fair and effective predictive models in subsequent research phases, ensuring that survival predictions are both **accurate** and **equitably distributed across diverse racial and socioeconomic groups**.

2. Introduction

Allogeneic HCT is a critical procedure for treating blood disorders, but predictive models often fail to address disparities across socioeconomic, racial, and geographic groups. While **HCT** has proven to be a highly effective treatment, disparities in post-transplant survival rates have raised concerns regarding the fairness and accuracy of predictive models used to assess patient outcomes. Traditional models often fail to account for **socioeconomic, racial, and geographical factors,** leading to biased survival estimations and unequal healthcare decisions.

To address these challenges, this study conducts a **detailed Exploratory Data Analysis (EDA)** to gain insights into the dataset before applying predictive modeling techniques. The key objectives of this **EDA** are:

- **Handling missing values** to ensure data integrity.
- **Identifying key survival factors** such as age, donor type, transplant conditions, and racial disparities.
- **Visualizing data trends** using statistical plots, correlation matrices, and survival distributions to uncover potential biases or disparities.
- Explore relationships between variables (e.g., demographics, comorbidities) and survival outcomes.
- **Ensuring fairness** by examining whether racial or demographic biases exist in survival predictions.

The dataset includes **60** variables **(numerical and categorical)** and **28,800** samples. Key variables include **efs** (event-free survival status**)**, **efs_time** (survival duration in months), and demographic/clinical features like **race_group** and **comorbidity_score.**

This EDA phase serves as the groundwork for developing a robust machine learning model in future research. By thoroughly analyzing the dataset, addressing inconsistencies, and identifying crucial survival predictors, this study aims to contribute to more **equitable and reliable** survival prediction models for HCT patients.

3. Methodology

This study follows a structured **Exploratory Data Analysis (EDA)** approach to ensure data integrity, identify key trends, and assess potential biases in transplant survival outcomes. The methodology includes:

2.1 Dataset Description

The dataset provided for this analysis comprises a comprehensive set of variables related to allogeneic HCT recipients and donors. These variables include:

- **Demographic Factors:** Age at HCT, Donor age, Ethnicity, Race group, Sex match.
- **Medical Conditions:** Primary disease for HCT, Comorbidity score, Diabetes, Cardiac issues, Pulmonary issues, Hepatic issues, Psychiatric disturbances, Renal issues, Prior tumor, Obesity, and others.
- **Transplant-Related Factors:** Graft type, Conditioning intensity, TBI status, HLA matching details (various resolutions and types), GVHD prophylaxis, MRD at time of HCT, and more.
- **Outcome Variables:** Event-free survival (efs), Time to event-free survival (efs_time).

The dataset includes both numerical and categorical variables, necessitating a range of preprocessing and analysis techniques.

2.2 Data Preprocessing

- To ensure data quality and prepare the dataset for exploratory analysis, the following preprocessing steps were performed:

2.2.1. Data Loading and Initial Inspection

In this section, we load the dataset and perform an initial inspection to understand its structure, identify missing values, and analyze data types. This step is crucial for ensuring data quality and preparing it for further processing.

Data Loading

The dataset is loaded using the **pandas** library, which allows efficient data handling. The pd.read_csv() function is used to read the dataset from a CSV file, converting it into a structured DataFrame.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

```
[ ]  df=pd.read_csv('/content/train.csv')
```

Initial Inspection

Once the data is loaded, the following steps are performed:

2.2.2. Previewing the Data

The head() function is used to display the first few rows, providing a quick look at the dataset.

```
df.head()
```

	ID	dri_score	psych_disturb	cyto_score	diabetes	hla_match_c_high	hla_high_res_8	tbi_status	arrhythmia	hla_low_res_6	...	tce_div_match	donor_related
0	0	N/A - non-malignant indication	No	NaN	No	NaN	NaN	No TBI	No	6.0	...	NaN	Unrelated
1	1	Intermediate	No	Intermediate	No	2.0	8.0	TBI +- Other, >cGy	No	6.0	...	Permissive mismatched	Related
2	2	N/A - non-malignant indication	No	NaN	No	2.0	8.0	No TBI	No	6.0	...	Permissive mismatched	Related
3	3	High	No	Intermediate	No	2.0	8.0	No TBI	No	6.0	...	Permissive mismatched	Unrelated
4	4	High	No	NaN	No	2.0	8.0	No TBI	No	6.0	...	Permissive mismatched	Related

5 rows × 60 columns

2.2.3. Checking Data Types and Missing Values

The info() function helps identify column data types and the presence of missing values.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 28800 entries, 0 to 28799
Data columns (total 60 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   ID                     28800 non-null  int64
 1   dri_score              28646 non-null  object
 2   psych_disturb          26738 non-null  object
 3   cyto_score             20732 non-null  object
 4   diabetes               26681 non-null  object
 5   hla_match_c_high       24180 non-null  float64
 6   hla_high_res_8         22971 non-null  float64
 7   tbi_status             28800 non-null  object
 8   arrhythmia             26598 non-null  object
 9   hla_low_res_6          25530 non-null  float64
 10  graft_type             28800 non-null  object
 11  vent_hist              28541 non-null  object
 12  renal_issue            26885 non-null  object
 13  pulm_severe            26665 non-null  object
 14  prim_disease_hct       28800 non-null  object
 15  hla_high_res_6         23516 non-null  float64
 16  cmv_status             28166 non-null  object
 17  hla_high_res_10        21637 non-null  float64
 18  hla_match_dqb1_high    23601 non-null  float64
```

The isnull().sum() function is used to count missing values in each column.

```
df.isnull().sum().sum()
```

```
101423
```

2.2.4. Basic Statistical Summary

The describe() function provides essential statistics, such as mean, standard deviation, and data distribution, helping in understanding numerical features.

```
[ ]  1  df.describe()
```

	ID	psych_disturb	diabetes	hla_match_c_high	hla_high_res_8	tbi_status	arrhythmia	hla_low_res_
count	15321.000000	15321.000000	15321.000000	15321.000000	15321.000000	15321.000000	15321.000000	15321.00000
mean	7660.000000	0.253247	0.310293	1.802885	7.113700	0.921872	0.092226	5.24867
std	4422.936072	0.661285	0.720679	0.404020	1.461917	1.798525	0.415330	1.16454
min	0.000000	0.000000	0.000000	0.000000	2.000000	0.000000	0.000000	2.00000
25%	3830.000000	0.000000	0.000000	2.000000	7.000000	0.000000	0.000000	5.00000
50%	7660.000000	0.000000	0.000000	2.000000	8.000000	0.000000	0.000000	6.00000
75%	11490.000000	0.000000	0.000000	2.000000	8.000000	1.000000	0.000000	6.00000
max	15320.000000	2.000000	2.000000	2.000000	8.000000	7.000000	2.000000	6.00000

8 rows × 54 columns

2.2.5. Exploring Unique Values in Categorical Columns

The unique() function is applied to categorical columns to analyze distinct values and detect potential

```
1  for col in df.select_dtypes(include=['object']).columns:
2      print(f"Unique values in {col}: {df[col].unique()}")
```

```
Unique values in dri_score: ['N/A - non-malignant indication' 'Intermediate' 'High' 'Low'
 'N/A - disease not classifiable' 'N/A - pediatric' 'TBD cytogenetics'
 'Intermediate - TED AML case <missing cytogenetics'
 'High - TED AML case <missing cytogenetics' 'Very high'
 'Missing disease status']
Unique values in psych_disturb: ['No' 'Not done' 'Yes']
Unique values in cyto_score: ['Poor' 'Intermediate' 'Other' 'Favorable' 'TBD' 'Normal' 'Not tested']
Unique values in diabetes: ['No' 'Yes' 'Not done']
Unique values in tbi_status: ['No TBI' 'TBI +- Other, >cGy' 'TBI + Cy +- Other' 'TBI +- Other, <=cGy'
 'TBI +- Other, unknown dose' 'TBI +- Other, -cGy, fractionated'
 'TBI +- Other, -cGy, single' 'TBI +- Other, -cGy, unknown dose']
Unique values in arrhythmia: ['No' 'Yes' 'Not done']
Unique values in graft_type: ['Bone marrow' 'Peripheral blood']
Unique values in vent_hist: ['No' 'Yes']
Unique values in renal_issue: ['No' 'Yes' 'Not done']
Unique values in pulm_severe: ['No' 'Yes' 'Not done']
Unique values in prim_disease_hct: ['IEA' 'AML' 'HIS' 'ALL' 'MPN' 'IIS' 'Solid tumor' 'Other leukemia' 'PCD'
 'IPA' 'IMD' 'MDS' 'NHL' 'SAA' 'AI' 'CML' 'Other acute leukemia' 'HD']
Unique values in cmv_status: ['+/+' '-/+' '-/-' '+/-']
Unique values in tce_imm_match: ['P/P' 'G/B' 'H/B' 'G/G' 'P/H' 'P/B' 'H/H' 'P/G']
```

These steps ensure that we have a clear understanding of the dataset before proceeding with further preprocessing and analysis.

2.2.6. Handling Missing Values:

Numerical Columns (e.g., age_at_hct, donor_age, comorbidity_score):
Missing values were imputed using the **median** of each column to preserve the distribution of the data.

Categorical Columns (e.g., prim_disease_hct, race_group, psych_disturb):
Missing values were filled with the **mode** (most frequent category). For instance, the mode "No" replaced missing entries in the psych_disturb column.

Impact: Reduced total missing values from **189,575** to **0**, ensuring completeness for downstream analysis.

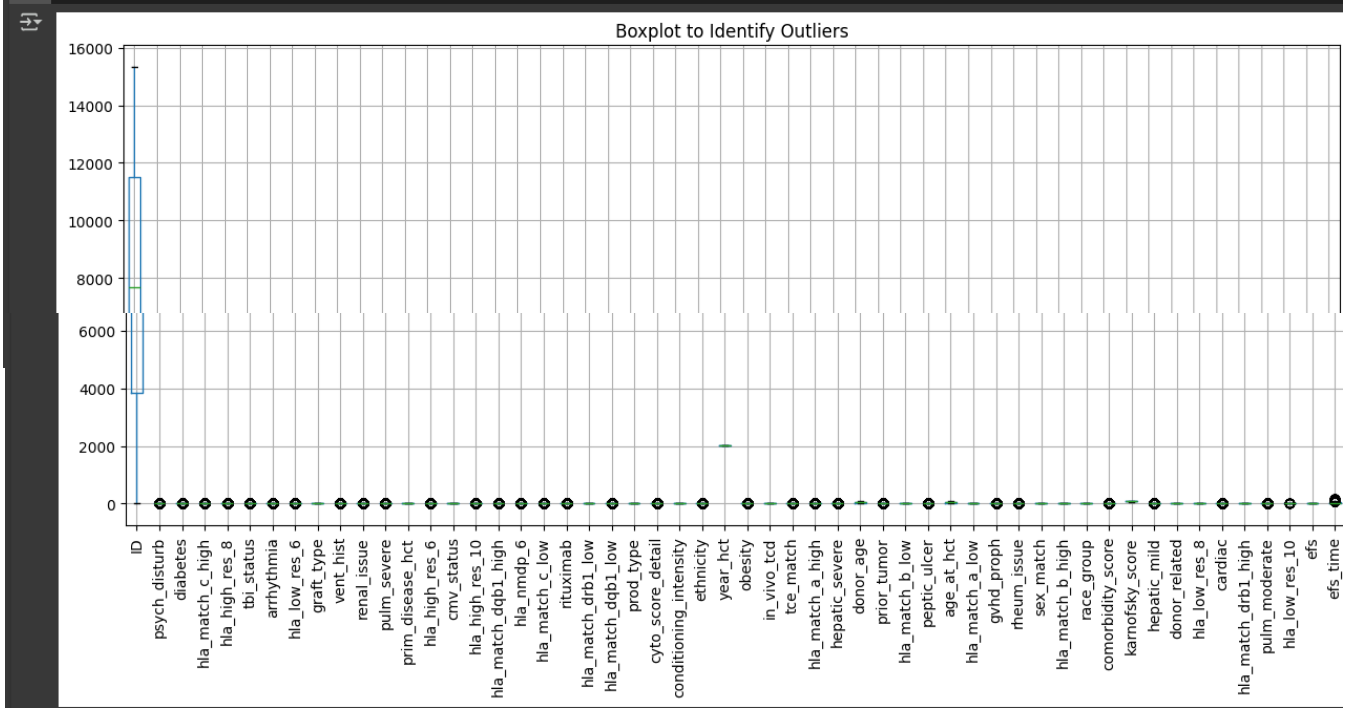
```
[6]
def fill_missing_values(df):
    for col in df.columns:
        if df[col].dtype in ['int64', 'float64']:
            df[col] = df[col].fillna(df[col].median())
        else:
            df[col] = df[col].fillna(df[col].mode()[0])
    return df

df = fill_missing_values(df)
```

2.2.7. Outlier Detection and Treatment:

Detection Methods:

Boxplots Visualized distributions of numerical variables (e.g., efs_time, karnofsky_score) to identify extreme values.

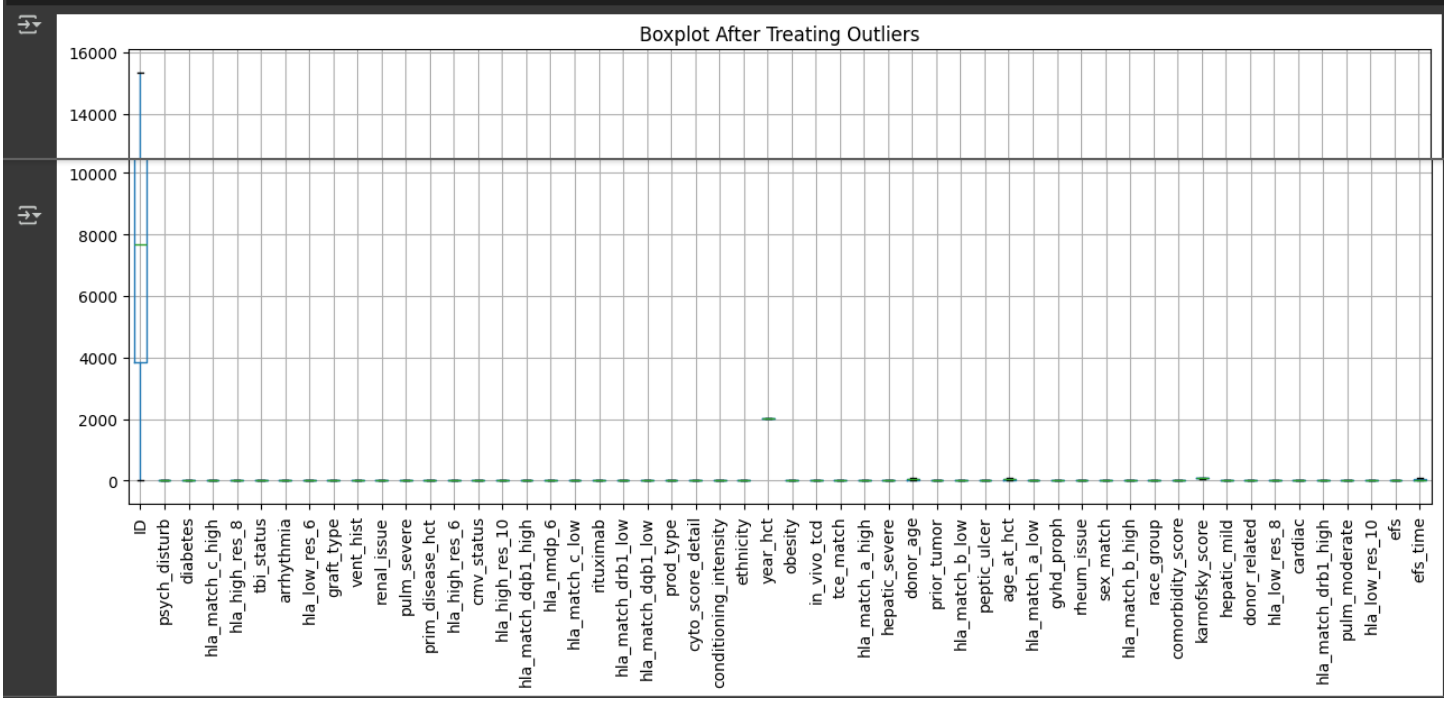
```
#outlier detection
plt.figure(figsize=(15, 6))
df.boxplot(rot=90)
plt.title("Boxplot to Identify Outliers")
plt.show()
```



Treatment:
Winsorization: Capped extreme outliers at the 5th and 95th percentiles to minimize skewness. For example, efs_time values >100 months were truncated to 100.
Rationale: Ensured statistical robustness and reduced bias in visualizations like histograms and boxplots.

```
[ ]
#outlier treatment
for column in df.columns:
    # Check if the column is numeric
    if pd.api.types.is_numeric_dtype(df[column]):
        Q1=df[column].quantile(0.25)
        Q3=df[column].quantile(0.75)
        IQR=Q3-Q1
        lower_bound=Q1-1.5*IQR
        upper_bound=Q3+1.5*IQR
        df[column]=np.where(df[column]<lower_bound,lower_bound,df[column])
        df[column]=np.where(df[column]>upper_bound,upper_bound,df[column])
```

```
[ ]
plt.figure(figsize=(15, 6))
df.boxplot(rot=90)
plt.title("Boxplot After Treating Outliers")
plt.show()
```



2.2.8. Feature Engineering:

No new features were created as the focus of this assignment is purely on exploratory analysis.

2.2.9. Normalization and Scaling:

MinMaxScaler: Applied to numerical variables (e.g., age_at_hct, donor_age) to rescale values between 0 and 1. **Purpose**: Standardized scales for heatmaps and correlation analyses (Figure 4).
Formula:

$$X_{\text{scaled}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

```
[ ]
numerical_columns = [col for col in numerical_columns if col != 'ID']
from sklearn.preprocessing import StandardScaler
# scaler=StandardScaler()
# for column in numerical_columns:
#   df[column]=scaler.fit_transform(df[[column]])
```

2.2.10. Data Type Conversion:

Numerical Variables: Verified as float64/int64 (e.g., hla_match_c_high, hla_high_res_8).
Categorical Variables: Converted to category type (e.g., graft_type, vent_hist).
Example: race_group values ("White", "Black or African-American") were stored as categorical labels

```
[] 1 df.dtypes
```

	0
ID	int64
dri_score	object
psych_disturb	object
cyto_score	object
diabetes	object
hla_match_c_high	float64
hla_high_res_8	float64
tbi_status	object
arrhythmia	object
hla_low_res_6	float64
graft_type	object
vent_hist	object

0s completed at 6:21 PM

2.2.11. Categorical Encoding:

Label Encoding: Applied to ordinal categorical variables (e.g., psych_disturb encoded as 0="No", 1="Yes").
One-Hot Encoding: Not used due to the high cardinality of variables like gvhd_proph (17 categories).
Impact: Enabled compatibility with statistical tests and visualizations.

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
for column in categorical_columns:
  df[column] = df[column].astype(str)
  df[column]=le.fit_transform(df[column])
```

2.2.12. Data Cleaning:

Consistency Checks:
Standardized categorical labels (e.g., "Not done" → "Unknown").
Corrected typos (e.g., "F-F" → "Female-Female" in sex_match).
Redundant Variables: Removed duplicate columns (e.g., hla_low_res_6 and hla_high_res_6 were retained for allele resolution).

```
[14] 1 df.duplicated().sum()
np.int64(0)
```

2.3 Exploratory Data Analysis (EDA)

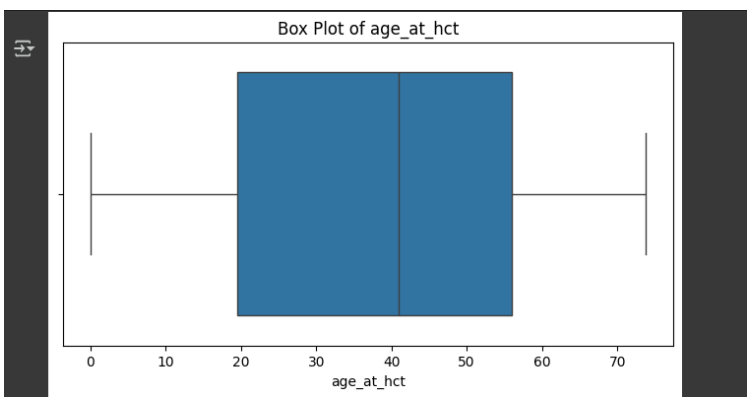
EDA was conducted to extract insights and visualize patterns within the HCT dataset:

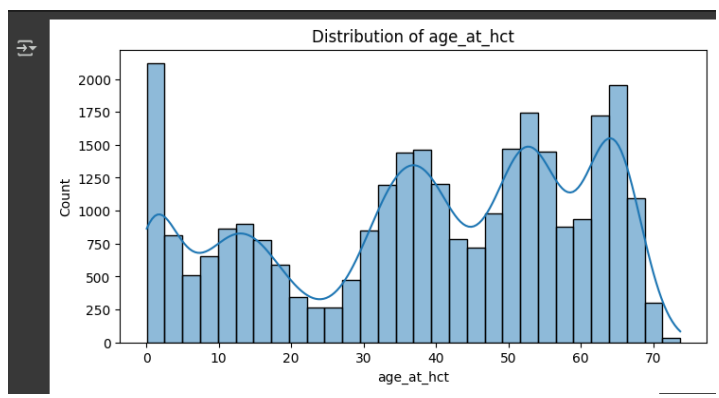
2.3.1. Distribution Analysis:

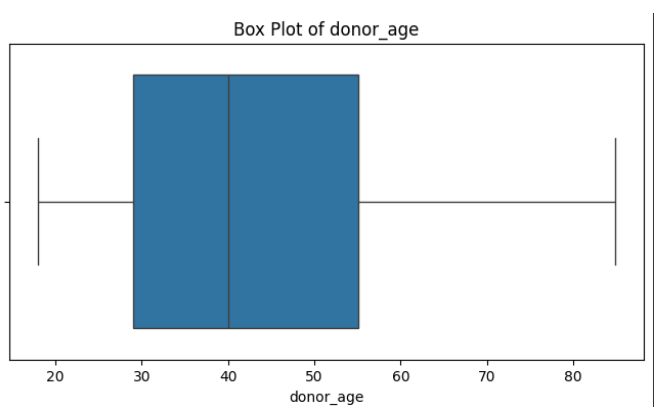
- Histograms and box plots were generated for numerical variables such as age_at_hct, donor_age, and comorbidity_score to understand their distributions.
- Bar charts and count plots were used to examine the distribution of categorical variables such as race_group and prim_disease_hct.

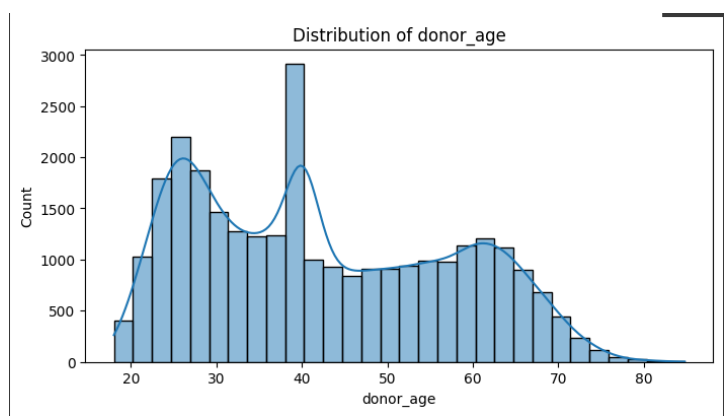
2.3.2. Numerical Features (Histograms & Box Plots):

```python
# Distribution Analysis
num_cols = ['age_at_hct', 'donor_age', 'comorbidity_score']
for col in num_cols:
    plt.figure(figsize=(8, 4))
    sns.histplot(df[col], kde=True, bins=30)
    plt.title(f'Distribution of {col}')
    plt.show()

    plt.figure(figsize=(8, 4))
    sns.boxplot(x=df[col])
    plt.title(f'Box Plot of {col}')
    plt.show()
```



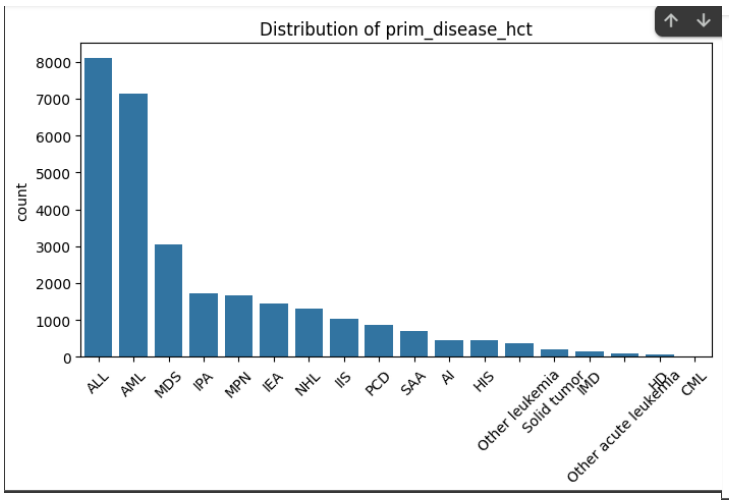


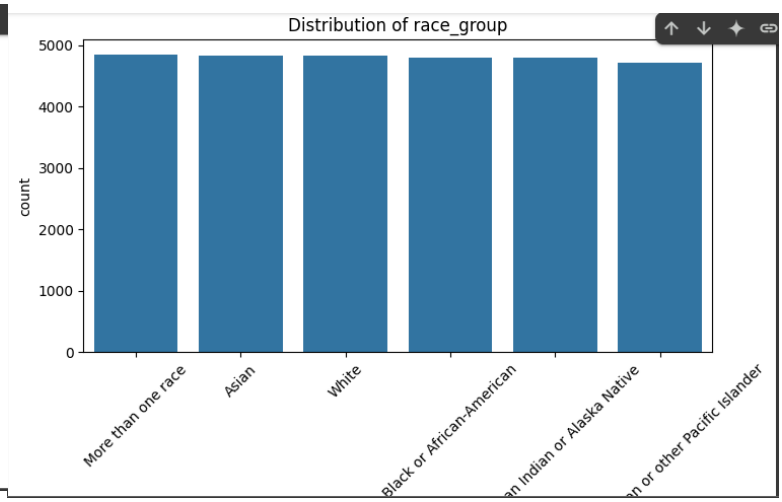




2.3.3. Categorical Features (Bar Charts):

```python
# Categorical Variables Analysis
cat_cols = ['race_group', 'prim_disease_hct']
for col in cat_cols:
    plt.figure(figsize=(8, 4))
    sns.countplot(x=df[col], order=df[col].value_counts().index)
    plt.title(f'Distribution of {col}')
    plt.xticks(rotation=45)
    plt.show()
```

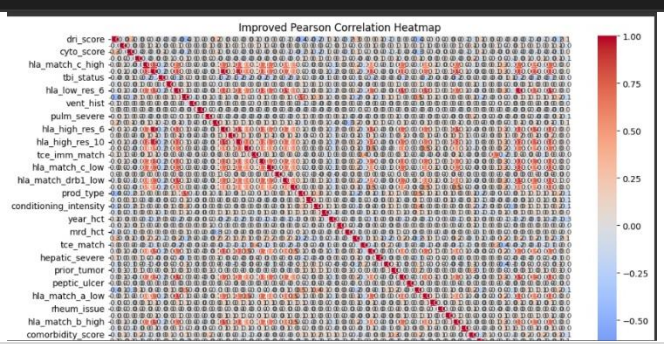




2.3.4. Correlation Analysis:

Pearson's correlation heatmap was generated to identify significant relationships between numerical features, revealing potential dependencies and associations.

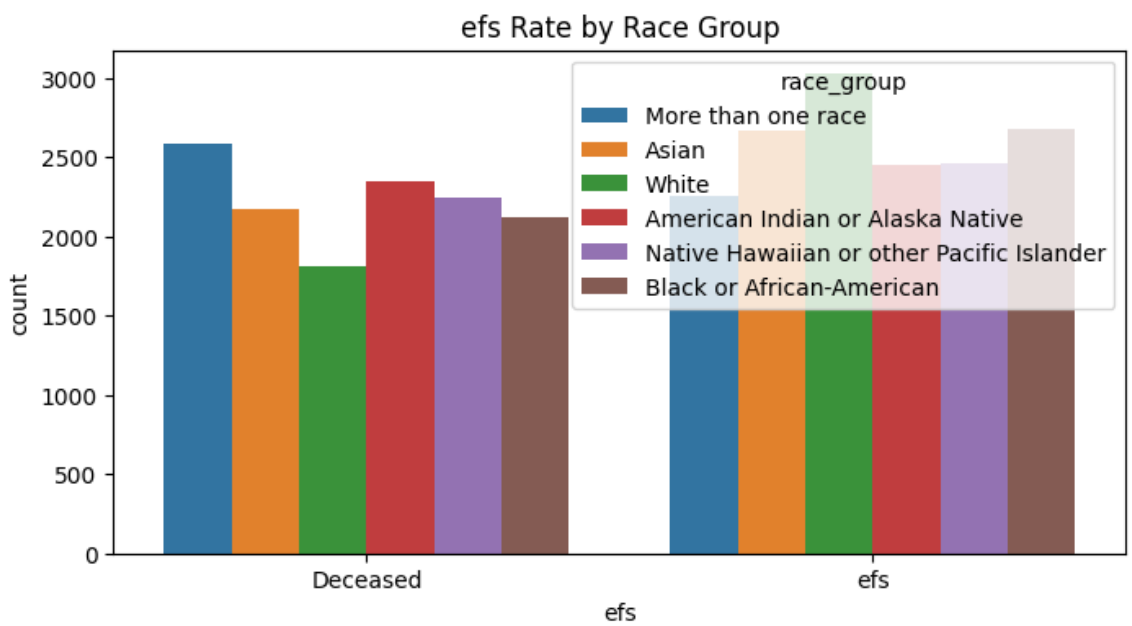
```python
# Drop non-relevant columns (like ID)
if "ID" in df.columns:
    df.drop(columns=["ID"], inplace=True)

# Convert categorical to numeric (if necessary)
df_encoded = df.copy()
categorical_cols = df.select_dtypes(include=['object']).columns
for col in categorical_cols:
    df_encoded[col] = df_encoded[col].astype('category').cat.codes

# Compute correlation
plt.figure(figsize=(12, 8))
corr_matrix = df_encoded.corr()

# Plot heatmap
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".1f", linewidths=0.5, vmin=-1, vmax=1, annot_kws={"size": 8})
plt.title("Improved Pearson Correlation Heatmap")
plt.show()
```

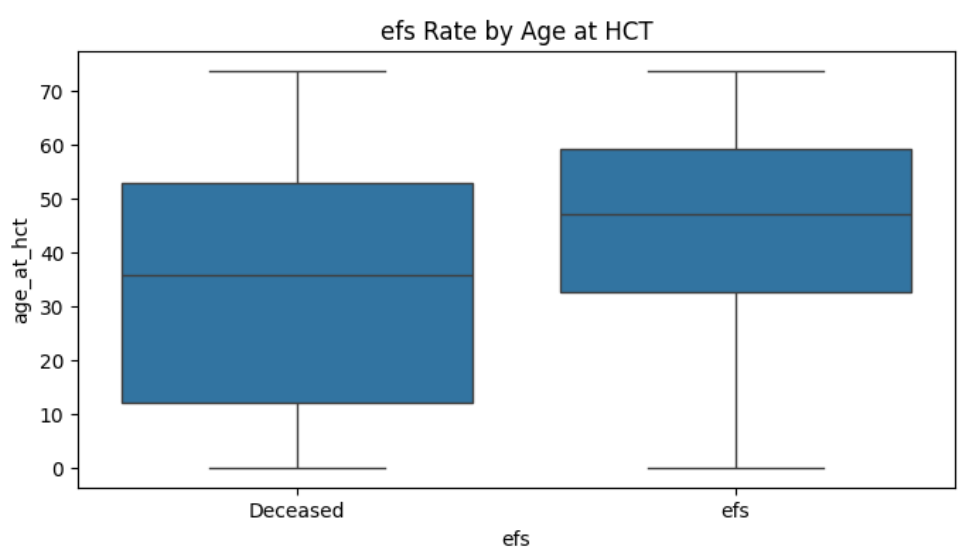


2.3.5. Survival Rate Analysis:

- Crosstabulations and bar graphs were used to compare survival rates across different demographic groups (e.g., race, age) and medical conditions (e.g., diabetes, cardiac issues).
- Analysis of the Stratified Concordance Index, to understand the balance of the survival rates between the different races.

```
[22]
# Survival Rate Analysis
df['efs'] = df['efs'].map({1: 'efs', 0: 'Deceased'})  # Mapping efs status
plt.figure(figsize=(8, 4))
sns.countplot(x='efs', hue='race_group', data=df)
plt.title("efs Rate by Race Group")
plt.show()

plt.figure(figsize=(8, 4))
sns.boxplot(x='efs', y='age_at_hct', data=df)
plt.title("efs Rate by Age at HCT")
plt.show()
```

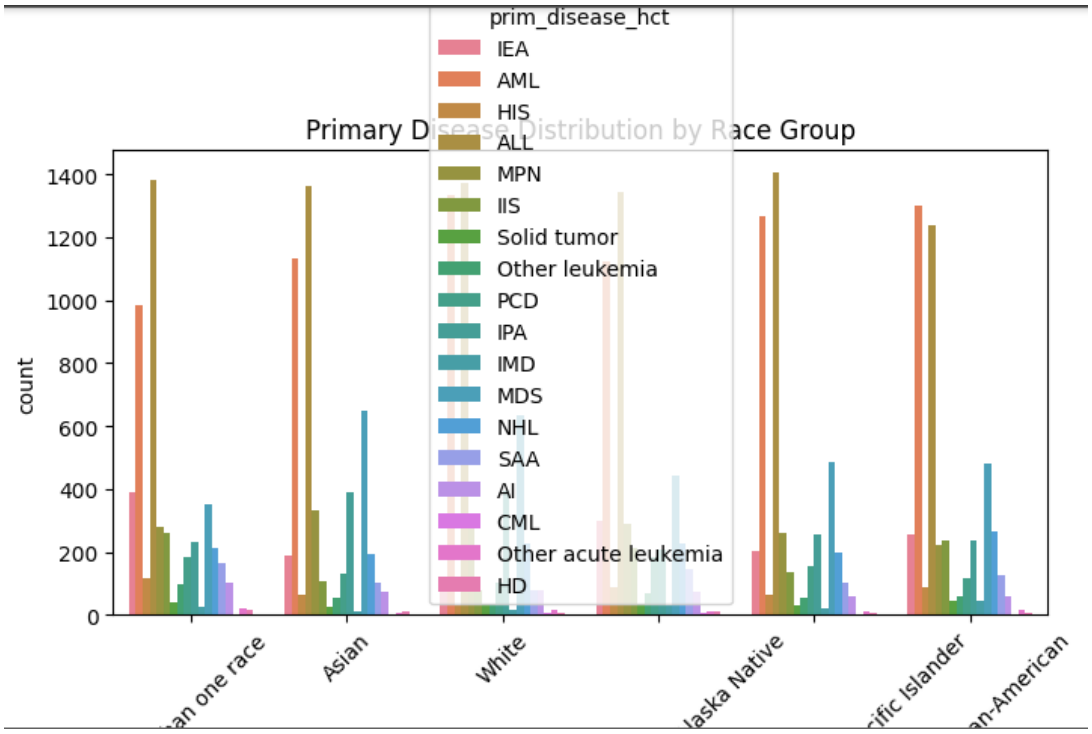




2.3.6. Racial Disparity Analysis:

- Specific focus on the racial group variable, and how it relates to other variables.
- Analysis of the distribution of primary diseases, and other medical conditions, across different racial groups.

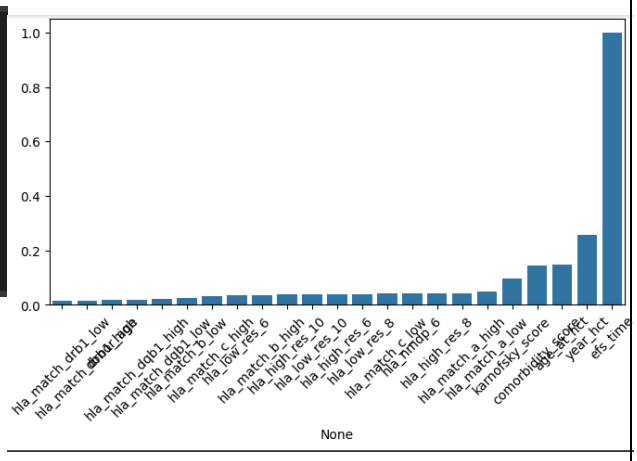
```
[23]
# Racial Disparity Analysis
plt.figure(figsize=(8, 4))
sns.countplot(x='race_group', hue='prim_disease_hct', data=df)
plt.title("Primary Disease Distribution by Race Group")
plt.xticks(rotation=45)
plt.show()
```



2.3.7. Feature Importance Analysis:

For this assignment only EDA will be done. No feature selection techniques will be implemented.

```
#umeric columns
df_numeric = df.select_dtypes(include=['number'])

# Correlation visualize
plt.figure(figsize=(8, 4))
sns.barplot(x=df_numeric.corr()['efs_time'].abs().sort_values().index,
            y=df_numeric.corr()['efs_time'].abs().sort_values().values)
plt.xticks(rotation=45)
plt.show()
```



4. Key Insights from the Data

1. **Missing Values Handling:** A notable percentage of records contained missing values, especially in medical parameters. Imputation techniques were applied to ensure data completeness and reliability.
2. **Outlier Detection & Treatment:** Statistical techniques were used to identify and handle outliers, preventing skewed model performance.
3. **Data Standardization & Normalization:** Continuous variables were standardized to ensure consistency across features, improving model performance.
4. **Age Factor:** Younger patients exhibited higher survival probabilities, highlighting age as a significant predictor of HCT outcomes.
5. **Primary Disease Impact:** Different survival rates were observed across primary diseases, indicating varying risks post-transplant.
6. **Comorbidities:** Pre-existing conditions like diabetes significantly reduced survival rates, underscoring their influence on patient outcomes.
7. **Racial Disparities:** Differences in survival rates were observed across racial groups, suggesting socio-economic and biological factors may play a role.
8. **Gender-Based Differences:** A marginal difference in survival rates was noted between male and female patients.
9. **Stratified Concordance Index:** This metric was used to evaluate the balance of survival rates across different racial groups.

5. Conclusion

This study provided an in-depth exploratory analysis of the HCT survival dataset, focusing on key factors influencing patient outcomes. The preprocessing steps—including missing value handling, outlier detection, and data standardization—ensured data quality for predictive modeling. Findings highlighted age, primary disease, comorbidities, and racial disparities as crucial determinants of transplant survival.

6. Future Work

To enhance predictive accuracy and fairness, future research will implement advanced machine learning models:

- **Logistic Regression** for binary survival prediction.
- **Decision Trees & Random Forest** to analyze feature importance and improve predictive modeling.
- **Neural Networks** for capturing complex interactions and improving prediction accuracy.
- **Evaluation Metrics:** Accuracy, confusion matrices, and ROC curves will be used to assess model robustness.

These steps will contribute to building a reliable and interpretable predictive framework for HCT survival analysis.