

## Contents

<b>Exploratory Data Analysis:</b>	2
Importing libraries:	2
Reading data:	3
Descriptive Statistics:	3
Dropping Unnecessary Column:	4
Finding Unique Values:	4
Separating Numerical & Categorical column:	4
Elimination of null values:	5
Graphical Representation of Data:	6
Subplot:	6
<b>Count Plot:</b>	7
Correlation:	8
Seaborn Implot:	9
Distplot:	9
Violin plot:	10
<i>Scatter Plot:</i>	11
Scatter plot with trend line:	11
Scatter Matrix Plot	14
scatters:	15
Conclusion of EDA:	16
<b>Prediction:</b>	17
Feature Importance:	17
Ordered Rank Features & Datastore:	18
Test-Train Data:	19
Model Selection:	19
XGBoost classification:	19
Linear Regression:	20
Cross-Validation with other models:	21
<b>References</b>	22
Presentation Video Link:	22

## Introduction

The dataset we will be working on is the Bone marrow transplant: children dataset. This type of this dataset is multivariate and it has 187 instances where the number of attributes is 39. The associated tasks with this dataset are classification and regression. It also contains some missing values. The data set includes children patients with a variety of hematologic disorders who received allogeneic unrelated donor hematopoietic stem cell transplantation without any manipulation. (Sikora, 21)

We will perform EDA (Exploratory Data Analysis) in this dataset. To tell a story with this dataset we need to identify all aspects of this dataset. As we already know this dataset is multi-variant, so we have to plot data taking two or multiple variables and show them accordingly. I worked on google colab to analyze the dataset. From the data dictionary, we already had this idea about the columns and other important factors. In colab, I imported all necessary libraries which I will be needed to plot my dataset.

## Exploratory Data Analysis:

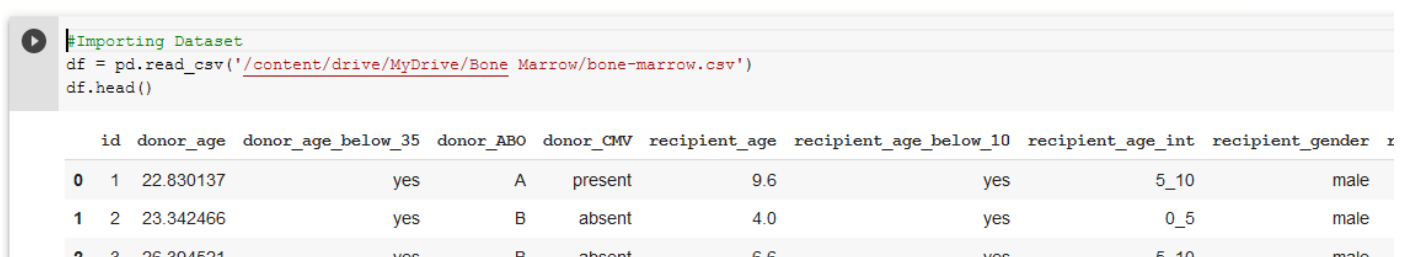
To perform EDA on this dataset we have to show some critical analysis to this dataset so that we can understand the pattern, discover inappropriate behavior of data, predict the summaries with the help of visualization. First, we will understand the data then we will show some insight.

### Importing libraries:

I started my EDA by importing important libraries which are NumPy, pandas, matplotlib, seaborn, and plotly. With the help of the pandas' library,

### Reading data:

I loaded the dataset into google colab with “`df = pd.read_csv()`”. After loading the dataset from the head of the dataset I can get a clear insight about the dataset. Similarly with “`.tail()`” will return the last portion of the dataset.



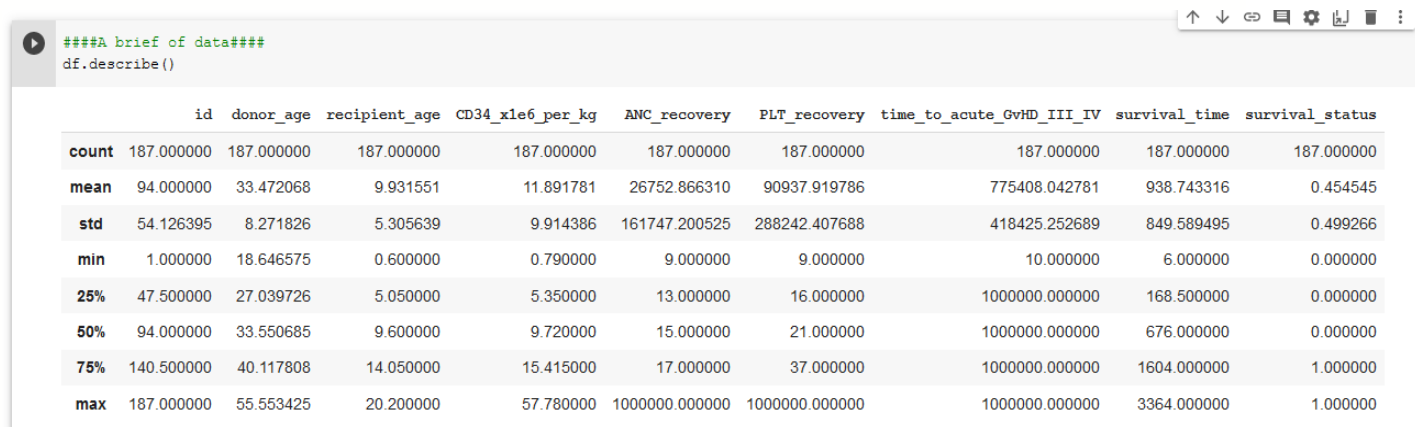
The screenshot shows a Google Colab interface. At the top, there is a code cell with the following text: `#Importing Dataset`, `df = pd.read_csv('/content/drive/MyDrive/Bone Marrow/bone-marrow.csv')`, and `df.head()`. Below the code cell, the first few rows of the dataset are displayed as a table. The table has 10 columns: `id`, `donor_age`, `donor_age_below_35`, `donor_ABO`, `donor_CMV`, `recipient_age`, `recipient_age_below_10`, `recipient_age_int`, and `recipient_gender`. The first three rows of data are visible, showing patient information and transplant details.

	id	donor_age	donor_age_below_35	donor_ABO	donor_CMV	recipient_age	recipient_age_below_10	recipient_age_int	recipient_gender
0	1	22.830137	yes	A	present	9.6	yes	5_10	male
1	2	23.342466	yes	B	absent	4.0	yes	0_5	male
2	3	26.304521	yes	B	absent	6.6	yes	5_10	male

Here I found the names of all columns using “df.columns”. Datatype is also a very important element to get the insight. “df.dtypes” will return the data types of all columns. Here we have int, float, and object type data.

### Descriptive Statistics:

To get a brief idea of this data I used “df.describe()”. It returns with count, mean, median, std, min, 25%, 50%, 75%, and max value of each column.



```
####A brief of data####
df.describe()
```

	id	donor_age	recipient_age	CD34_xle6_per_kg	ANC_recovery	PLT_recovery	time_to_acute_GvHD_III_IV	survival_time	survival_status
count	187.000000	187.000000	187.000000	187.000000	187.000000	187.000000	187.000000	187.000000	187.000000
mean	94.000000	33.472068	9.931551	11.891781	26752.866310	90937.919786	775408.042781	938.743316	0.454545
std	54.126395	8.271826	5.305639	9.914386	161747.200525	288242.407688	418425.252689	849.589495	0.499266
min	1.000000	18.646575	0.600000	0.790000	9.000000	9.000000	10.000000	6.000000	0.000000
25%	47.500000	27.039726	5.050000	5.350000	13.000000	16.000000	1000000.000000	168.500000	0.000000
50%	94.000000	33.550685	9.600000	9.720000	15.000000	21.000000	1000000.000000	676.000000	0.000000
75%	140.500000	40.117808	14.050000	15.415000	17.000000	37.000000	1000000.000000	1604.000000	1.000000
max	187.000000	55.553425	20.200000	57.780000	1000000.000000	1000000.000000	1000000.000000	3364.000000	1.000000

### 2. A brief of the whole dataset

If I explain survival time we have a total of 187 data, where the mean value is 938 days and the standard deviation is 849.58 days, minimum 6 days, 25% data has 168 days, 50% data has 676 days, 75% data has 1604 days, and max survival time is 3364 days. Other columns have the same explanation as well.

### Dropping Unnecessary Column:

At this point, I think I don't need the column “id” and I will drop this column with “df.drop('id',axis=1, inplace=True)”. There are more columns which are redundant. I will drop or remove those columns from the data frame to get more efficiency. The columns I am going to drop are:

- id
- recipient\_age\_below\_10
- recipient\_age\_int
- gender\_match
- HLA\_mismatch

antigen  
acute\_GvHD\_II\_III\_IV  
acute\_GvHD\_III\_IV

After dropping these columns new data frame shape is (187,30). Reduced data is more convenient to work with.

#### Finding Unique Values:

To understand how much unique value each column has I used “`df.nunique()`” and it returns the number of unique values each column has. For example `recipient_body_mass` has 131 different values. `stem_cell_source` has 2 different values. These values are attributes of these columns.

#### Separating Numerical & Categorical column:

This dataset has both numerical columns and categorical columns. I will separate them for a better insight. After separating them I have two types of columns. Here categorical column contains those columns those have non-numeric values.

There are 19 categorical columns. The categorical columns are:

donor_age_below_35	CMV_status
donor_ABO	HLA_match
donor_CMV	allele
recipient_gender	HLA_group_1
recipient_ABO	risk_group
recipient_rh	stem_cell_source
recipient_CMV	tx_post_relapse
disease	extensive_chronic_GvHD
disease_group	relapse
ABO_match	

I also extract numerical columns from the dataframe. Numerical columns are most often use for visualization. We get proper insight from numerical data. This dataset has 11 numerical column. They are:

donor_age	CD34_x1e6_per_kg
recipient_age	CD3_x1e8_per_kg
recipient_body_mass	CD3_to_CD34_ratio

ANC\_recovery  
PLT\_recovery  
time\_to\_acute\_GvHD\_III\_IV

survival\_time  
survival\_status

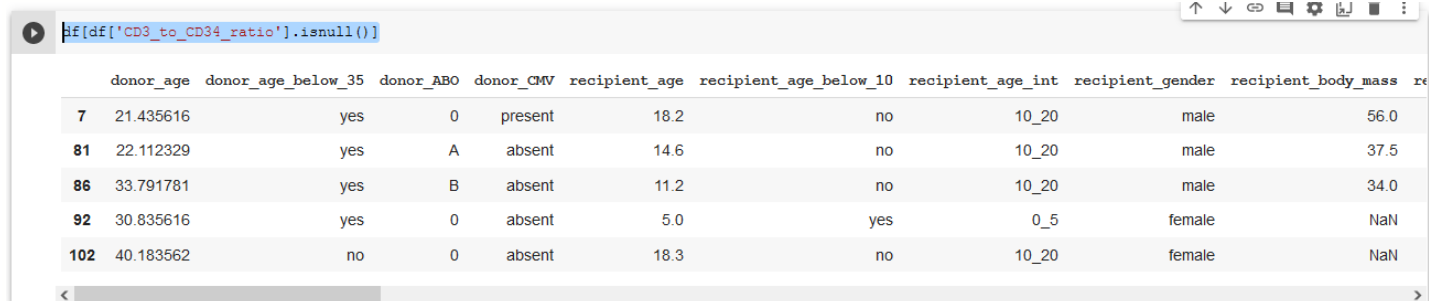
### Elimination of null values:

At this point, I will look for if there are any null values in the dataset. Null value creates a problem for visualization as well as for prediction. We will eliminate all null, nan values before our model prediction. To get higher accuracy, we have to eliminate them.

In this dataset there are 5 columns contains null values. Getting insight from numerical values I seen column “CD3\_to\_CD34\_ratio” contains null value. “df[df['CD3\_to\_CD34\_ratio'].isnull()]” returns all indexes that has null values.

```
[ ] df['CD3_xle8_per_kg'].isnull().sum()
```

5



	donor_age	donor_age_below_35	donor_ABO	donor_CMV	recipient_age	recipient_age_below_10	recipient_age_int	recipient_gender	recipient_body_mass	re
7	21.435616	yes	0	present	18.2	no	10_20	male	56.0	
81	22.112329	yes	A	absent	14.6	no	10_20	male	37.5	
86	33.791781	yes	B	absent	11.2	no	10_20	male	34.0	
92	30.835616	yes	0	absent	5.0	yes	0_5	female	NaN	
102	40.183562	no	0	absent	18.3	no	10_20	female	NaN	

### 3. Indexes contains NULL values

“df.dropna(inplace=True)” This will eliminate these null values. After using this command I use df[num\_col].isnull().sum() and found no null values in the data frame.

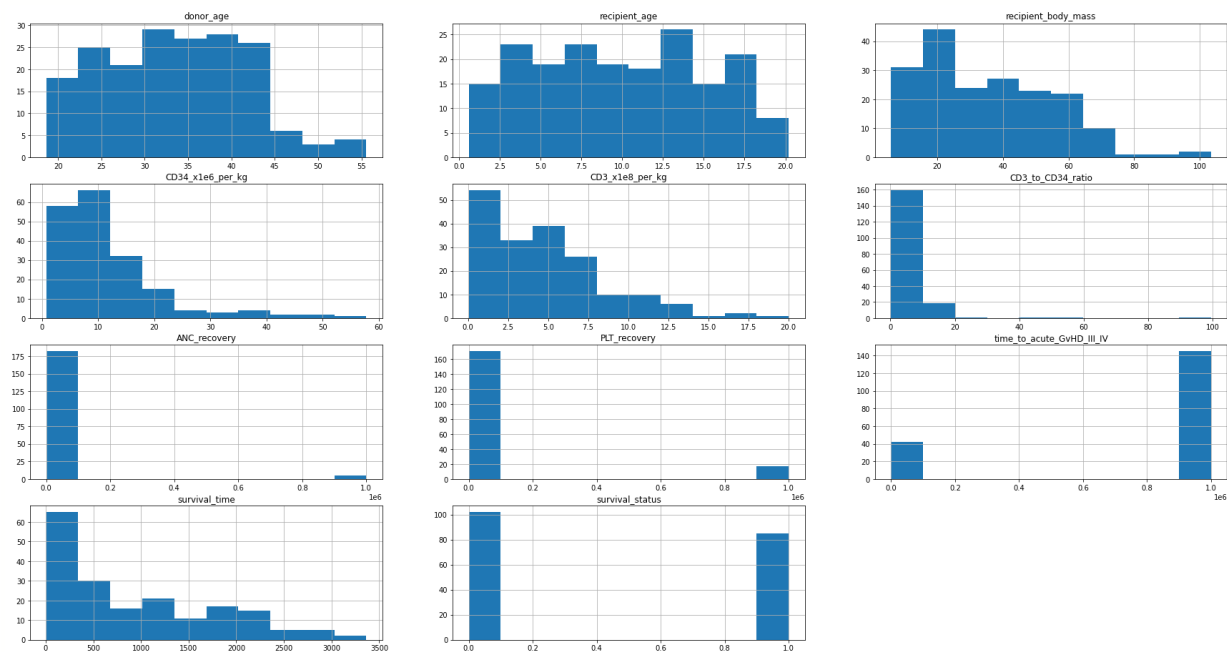
Now we have a fresh data frame without null values and it is ready for visualization. We will visualize from a different aspect.

## Graphical Representation of Data:

As we have two types of column numerical and categorical. First I will look into the numerical column to visualize the distribution. As it is multivariate the visualization will be consist of multiple variable.

### Subplot:

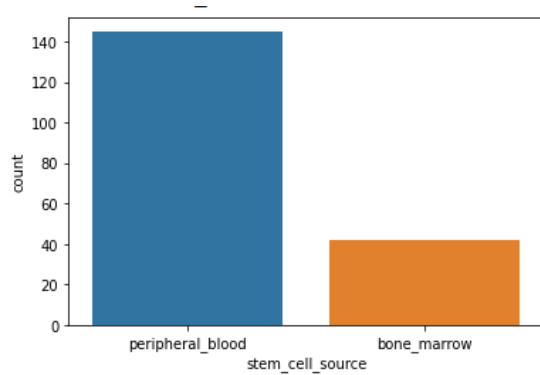
With this visualization, we can conclude that for the column donor age maximum donors are age20 -40. On the contrary recipient's age is from 2.5 to 20 and the maximum is 12.5-13. This is telling us about how all numerical data is distributed.



### 4. Subplot of all Numerical column

**Count Plot:** I used count plot to a categorical data “stem\_cell\_source” to understand the source of hematopoietic stem cells. Here the insight is telling that 140 is from

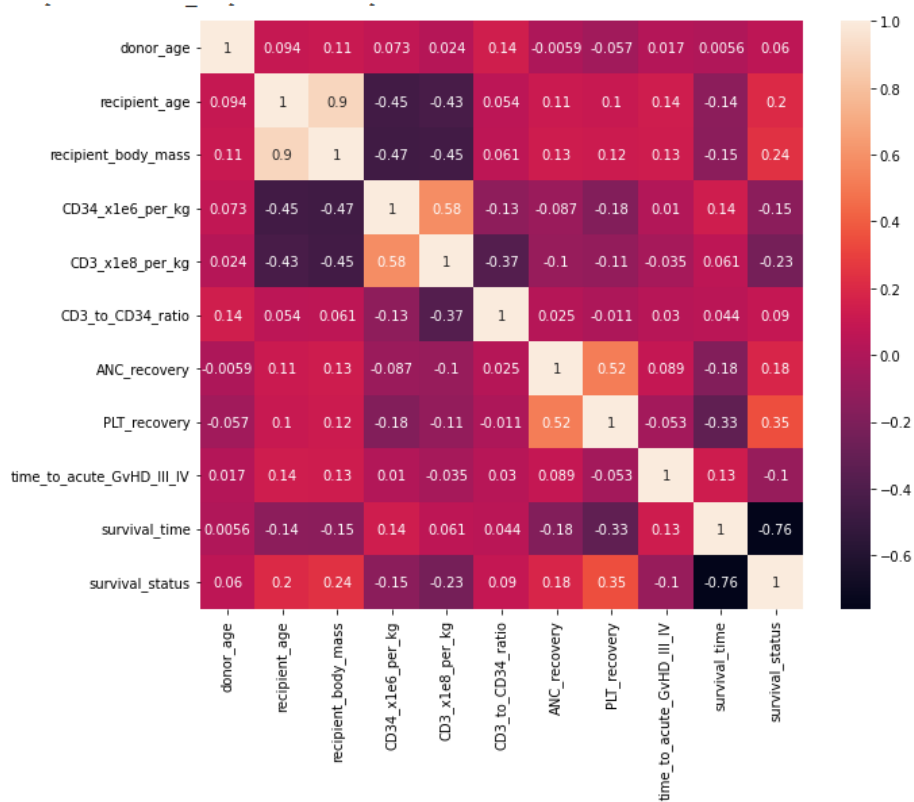
peripheral blood and 47 is from bone marrow. Here I will conclude the source of stem cell source differ because of other attributes like collection method, cellular content , and transplant outcomes. Here the popular or the major source is peripheral blood.



5. Countplot of stem cell source

## Correlation:

Getting insight into correlation will give us a clear vision about which data is strongly correlated with which data. It is a way to know more about the data. Using “df.corr()” we get the correlation table. After generating a correlation table, I showed that data into a heatmap. Heatmap gives us an easy way to understand the correlation. Observing the heatmap I can conclude that recipient age is negatively correlated with survival time. Recipient body mass is highly correlated with CD34\_x1e6\_per\_k column. Again survival status is highly correlated with survival time. Which columns meet in dark position in the heatmap are more correlated than others. With this method, we can describe every other column.

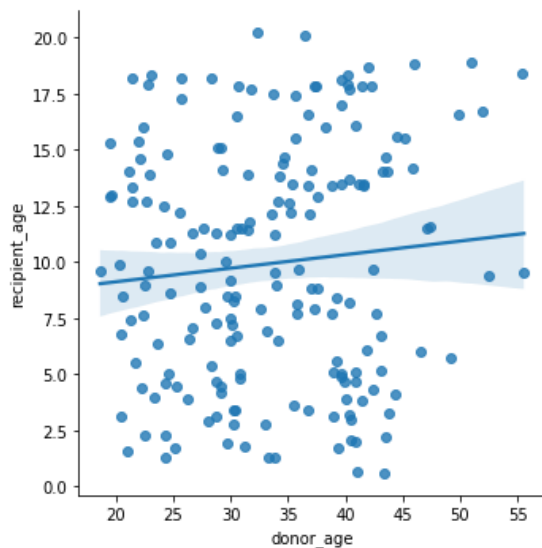


## 6. Correlation of the Data Frame



#### Seaborn Implot:

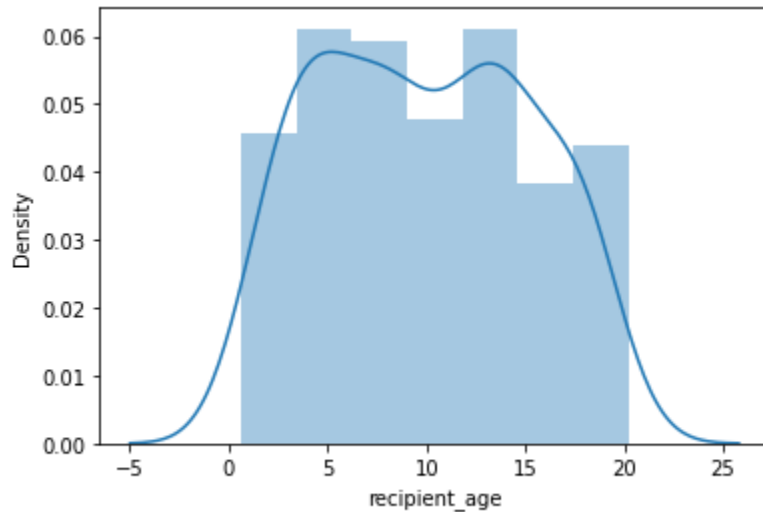
In this insight, it is clear that donor age and recipient age are hardly near. There is a huge distance between donor age and recipient age. Most of the recipients are children and they are not close to 20. On the other hand, donors are mostly adults and their age is between 20 to 45. This plot satisfies our previous plot which was the subplot of the numerical column. In both cases, we get the same result in different visuals.



7. Seaborn Implot of Donor Age and Recipient Age

#### Distplot:

Again distplot of recipient age gives us the same result as previous insight. Most recipients are between 2.5 to 15 years. Visualizing recipient age is important because other variables like survival status, survival time, plt recovery, and recovery, disease columns are related to this column.



8. Distplot of recipient age

#### Violin plot:

This violin plot is the distribution of recipient age with respect to disease group. The disease group has two attributes and they are malignant and nonmalignant.

In this visual for the malignant group, the max recipient age is 18.2 and minimum is 0.7, and the median is 7.75. Notable that here q1 is 5.55 and q3 is 14.675. For the recipient age, the 10 kernel density estimation is 0.57.

For the nonmalignant group, the max recipient age is 20.2 and minimum is 0.6, and the median is 11.3. Notable that here q1 is 4.45 and q3 is 10.675. For the recipient age, It got 10 as kernel density estimation is 0.78.

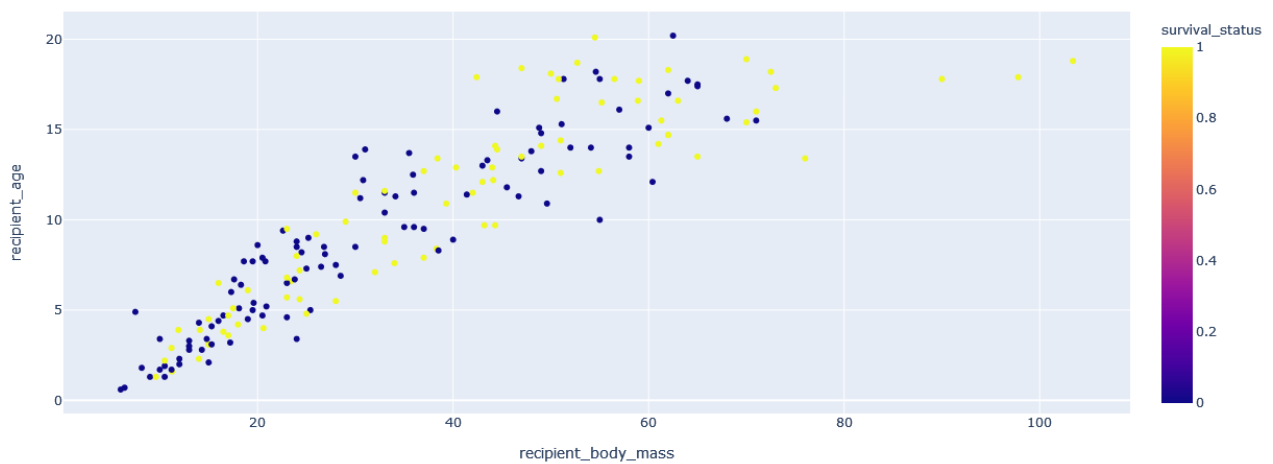
For different disease groups, we have different data. Overall kernel density estimation also changed for the two disease groups.



## 9. Violin plot for recipient age and disease type

### Scatter Plot:

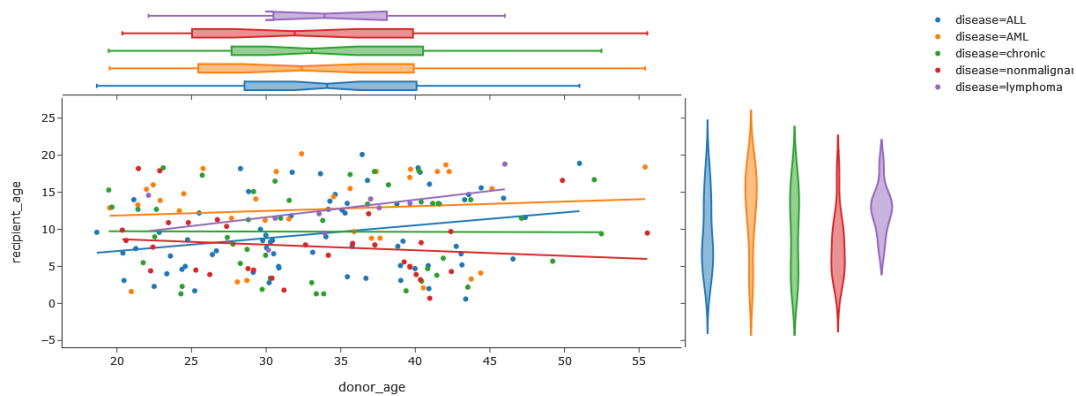
This plot will give us an understanding of how recipient body mass and recipient age are related. This shows us recipient with higher body mass has and higher age has survived but with lower mass and age unable to survive. So body mass is an important probability in terms of survival in bone marrow transplants. For example, a recipient with age 0.6 and mass 6 not survive while age 18.3 and mass 103.4 survived.



## 10. Scatter plot recipient body mass and recipient age

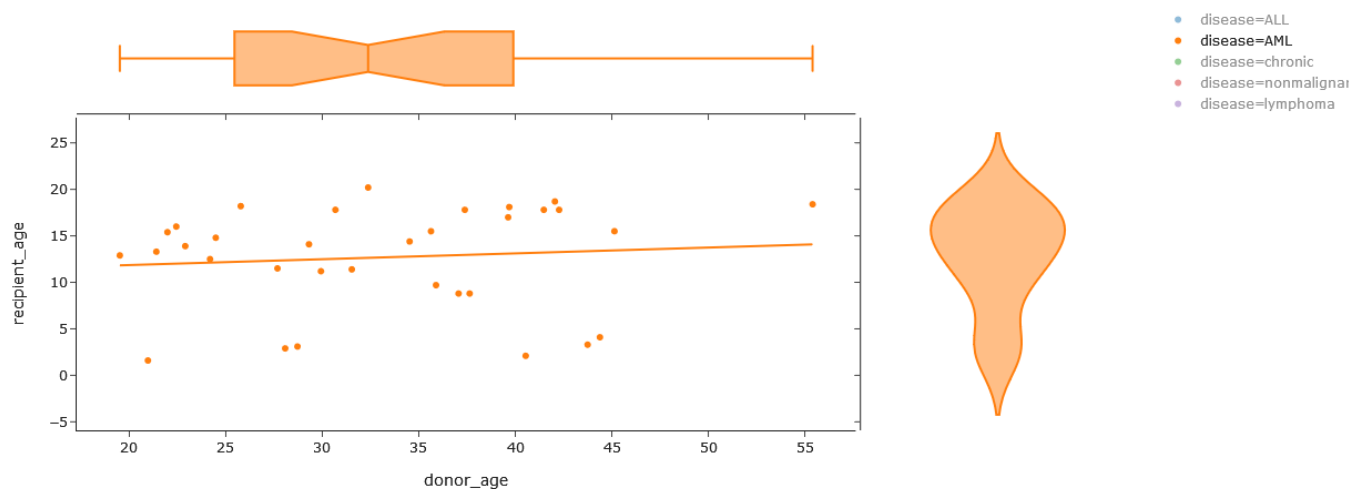
### Scatter plot with trend line:

I have extracted a lot of important insight using scatter plot with trend line also violin. Here we have both violin and box plots along with a trend line.



## 11. Scatter plot, Violin, box plot for donor age and recipient age

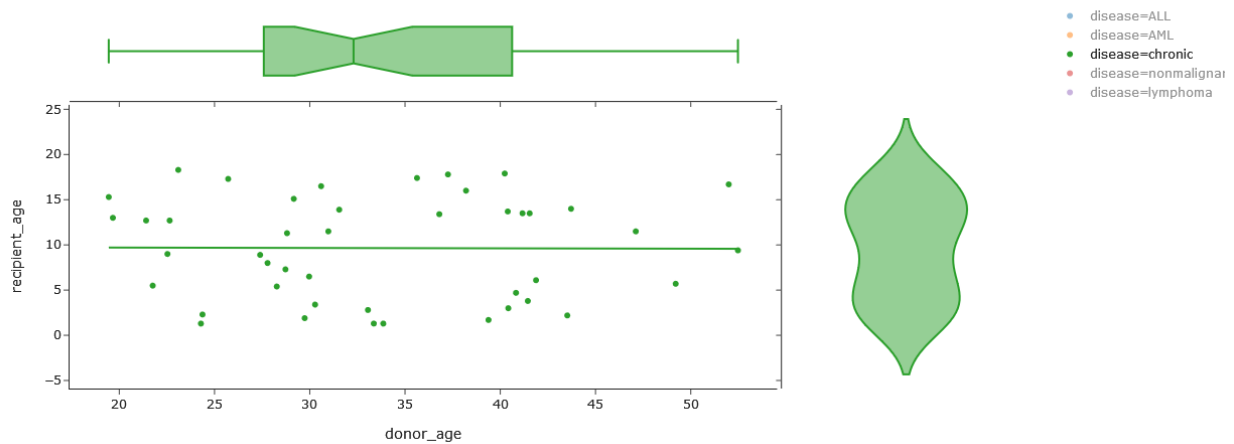
Here we plotted three important features which are donor age, recipient age concerning the disease column. Here the disease has 5 elements that are All, AML, Chorionic, nonmalignant. For a particular disease, we have a different scenario in donor age and recipient age.



## 12. Scatter plot with trend line showing AML

**AML:** Donor minimum age 19.50 Maximum age 55.40 median is 32.37.

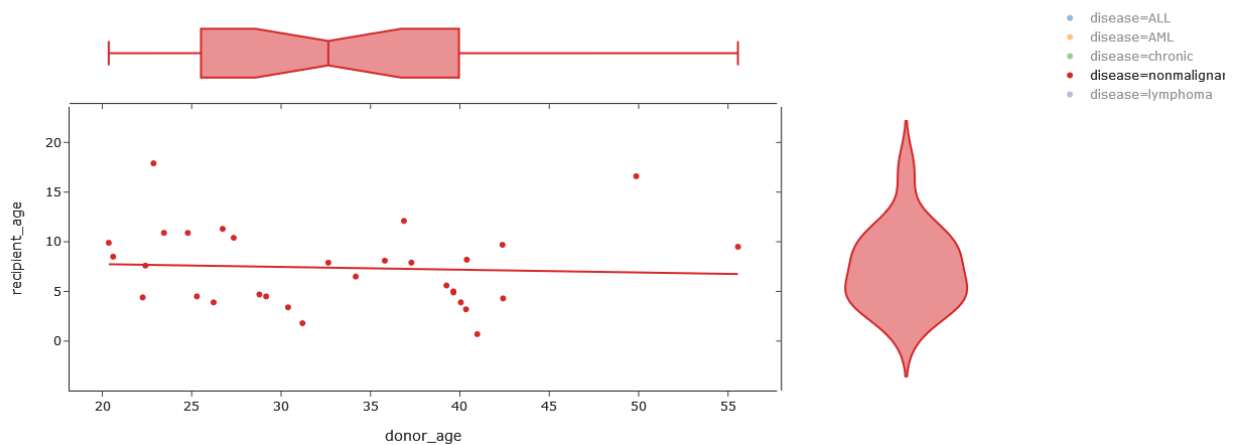
The recipient minimum age is 1.6, and the maximum is 20.2, and the median is 14.1. In the OLS trend line donor age 37.63 and recipient 13.43(trend). OLS return 0.0098 means these are rarely dependent.



### 13. Scatter plot with trend line showing chronic

**Chronic:** Donor minimum age 19.44 Maximum age 55.40 median is 32.29.

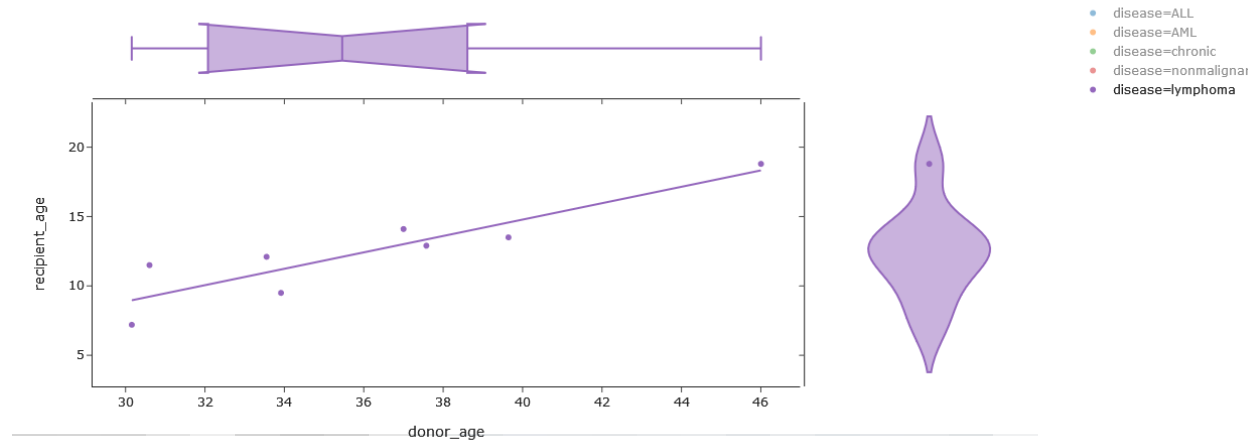
The recipient minimum age is 1.3, and the maximum is 18.3, and the median is 10.35. In the OLS trend line donor age 35.62 and recipient 9.64(trend). OLS return 0.000 means these are not even dependent.



### 14. Scatter plot with a trend line showing chronic Nonmalignant

**Nonmalignant:** Donor minimum age 20.35 Maximum age 55.55 median is 32.64.

The recipient minimum age is 0.7, and the maximum is 17.9, and the median is 7.6. In the OLS trend line donor age 49.86 and recipient 6.91(trend). OLS return 0.0039 means these are rarely dependent.

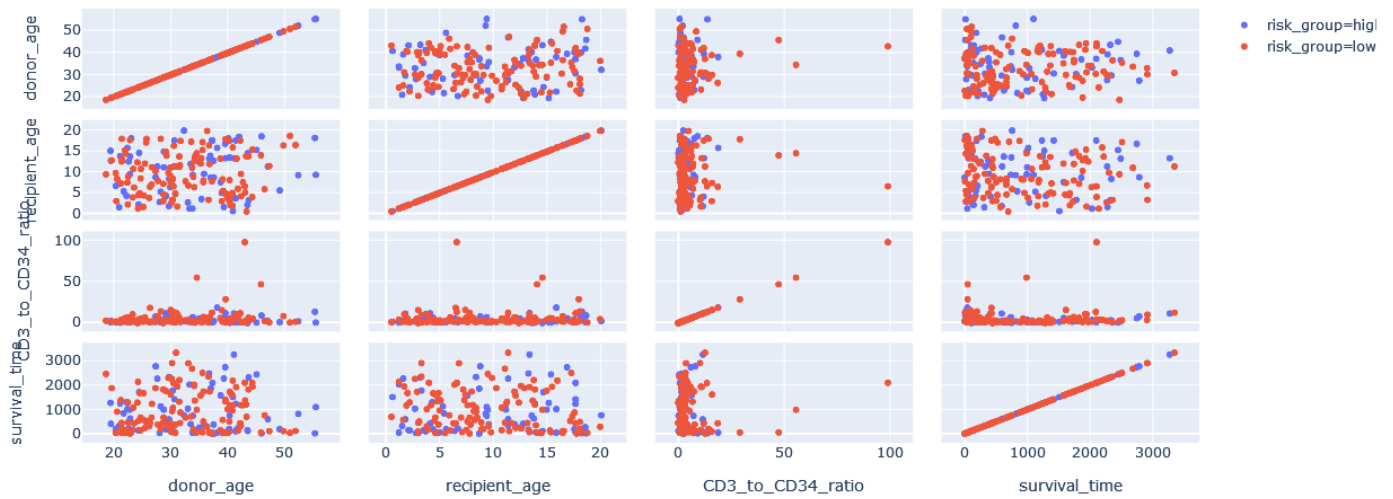


*Lymphoma*: Donor minimum age 30.15 Maximum age 46.00 median is 35.45.

The recipient minimum age is 7.2, and the maximum is 18.8, and the median is 12.5. In the OLS trend line donor age 39.64.86 and recipient 14.57(trend). OLS return 0.81 means these are highly dependent.

Conclusion: Donor age and recipient age do not vary due to disease. The values remained close in every disease study.

*Scatter Matrix Plot*: Here in the scatter plot matrix each column has been plotted with pair. For the disease, three columns has been shown. The distribution is shown based on “risk\_group”.



## 16. Scatter Matrix plot

I developed three functions to automate this visualization. I used the violin, scatters, and KDE plot to automate. These functions are based on survival status.

```
[45] def violin(col):
      fig=px.violin(df,y=col,x='survival_status',color = 'disease',box=True)
      return fig.show()
```

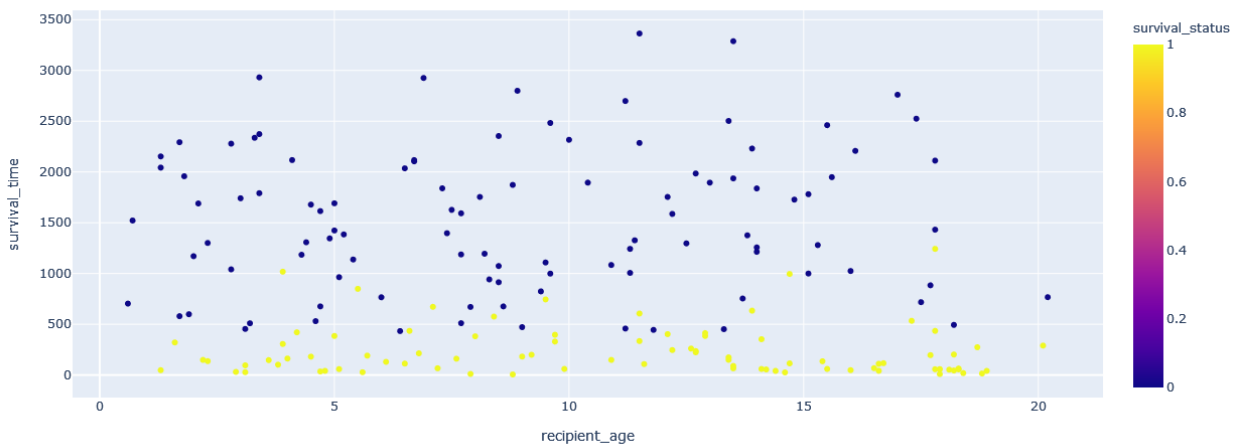
```
[46] def scatters(col1,col2):
      fig = px.scatter(df,x=col1,y=col2,color='survival_status')
      return fig.show()
```

```
[47] def kde_plot(feature):
      grid = sns.FacetGrid(df,hue='survival_status',aspect = 2)
      grid.map(sns.kdeplot,feature)
      grid.add_legend()
```

## 17. Automated EDA by creating functions

scatters:

Here I can conclude that survival time does not depend on recipient age. Because in every age after transplant they have higher to lower survival days.



## 18. Scatter plot among survival time and recipient age

### Conclusion of EDA:

It's a multivariate dataset that contains 187 rows and 39 columns with some missing values. From the study of this dataset, I observed that donors are not



children and they are all above 18. All recipients are children. After transplantation maximum survival days are 3364 days and a minimum of 6 days. The survival days do not depend on the recipient's body mass or age. Neutrophils recovery(ANC\_recovery) and Platelet count (PLT\_Recovery) have a good impact on survival time. The survival time of males and females are mostly the same. For successful transplantation, gender is not an obstacle as it is not showing remarkable behavior in visual. In most of the cases, HLA has been matched which is a very good factor. More than 85 cases HLA has been matched and resulted in 10/10. ABO also matched in 130 cases.

## Prediction:

### Feature Importance:

To approach a better prediction model we have to do feature importance then we have to select the prediction model. Techniques that assign a score to input features depending on how valuable they are at predicting a target variable are known as feature importance. For our dataset, we will import LabelEncoder from sci-kit learn. We have already separated our categorical column and numerical column. Here I need the categorical column to scale for our training column. To prepare the data for training, I'll use "fit transform()" to scale the data and learn the scaling parameters. This will help to scale the features. The model will learn the mean, variance of the features of the training set. This is the standard procedure we follow for scaling in terms of data analysis. (sklearn.preprocessing.StandardScaler, 2021)

Feature Selection: Here we will select those features that have a higher contribution to the whole dataset. It means we will select those features by which we can get the best predictions. If we do not do feature selection correctly our prediction model will not have a proper accuracy. It enables the model to reduce overfitting by reducing redundant data. Less redundant data means there will be no noise in the data and it will give us a proper prediction. It also makes the algorithm work faster as it already reduced redundant data. Besides this, I will use the Chi-Square test to determine if the categorical data has any remarkable correlation in them. It also impacts how well a model matches up with actual data.

Dependent and Independent columns: For the bone marrow dataset I chose the "survival\_status" as an independent column. As it returns 1/0 which denotes if the person is alive or dead. This column had a dominating behavior over other columns. So I am keeping this column as independent and all other columns dependent column.

**Ordered Rank Features & Datastore:** I ordered the feature rank using SelectKBest where score\_func = chi2(Chi-Square) and k=20. It will return me the ordered features data scores.

```
[ ] ordered_rank_features = SelectKBest(score_func=chi2,k=20)
    ordered_feature = ordered_rank_features.fit(X,y)

[ ] ordered_feature

SelectKBest(k=20, score_func=<function chi2 at 0x7f61395714d0>)

[ ] ordered_feature.scores_

array([2.17147479e+00, 1.22723872e+00, 5.40863786e-02, 1.18362390e-01,
       1.90199712e+01, 1.80809192e+00, 1.66699677e-01, 8.23579810e-02,
       1.08080071e+02, 1.27994184e-06, 6.70401039e-02, 2.92629263e-01,
       2.46690923e-02, 1.02156847e+00, 3.88244122e-02, 1.93069307e-01,
       1.01571387e+00, 1.17345579e-01, 4.07448152e-05, 1.03539766e-02,
       2.73805158e-03, 1.01207878e-02, 1.61164615e+00, 7.70349022e-01,
       2.49371556e+00, 3.95011765e+01, 3.06597762e+01, 2.50105869e+01,
       4.98415927e+06, 1.99410709e+07, 2.30124847e-01, 2.23790898e+00,
       4.04380397e+05, 2.98671359e+00, 1.60595282e+01, 7.76599042e+04])
```

## 22. Order rank scores

I got the final data scores using the code below:

```
“datascores = pd.DataFrame(ordered_feature.scores_,columns=['Score'])”
```

This data score will help me to identify the best features for prediction. I ranked the feature which is near largest. I used this

`“features_rank.nlargest(10, 'Score')”` to get the best features which will give me a proper accuracy. I selected 10 columns for my model. My selected columns are

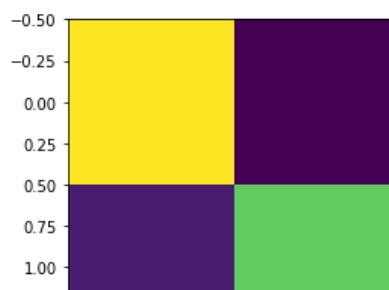
```
'PLT_recovery', 'ANC_recovery', 'time_to_acute_GvHD_III_IV',  
'survival_time', 'recipient_body_mass', 'CD34_xle6_per_kg',  
'CD3_xle8_per_kg', 'CD3_to_CD34_ratio', 'recipient_age', 'relapse'
```

**Test-Train Data:** In this section, I will use a portion of data for training and another portion will be selected for testing. So I imported `train_test_split` from `sklearn` model selection. I took 25% data for testing and 75% data for training. Here I use train data to fit the model and test data to test it.

```
“X_train,X_test, y_train,y_test = train_test_split(X_new,y,  
random_state=109,test_size=0.25)”
```

**Model Selection:** The bone marrow transplant dataset is univariate and the associated task with this dataset is classification and regression. For our classification model, I choose the XGBoost classification.

**XGBoost classification:** XgBoost is powerful and highly scalable. It uses parallel computing and distributed computing as well. Besides these, it offers proper memory management. (Admin, 2018). Data scientists and Kaggle competitors love it. Solving large-scale problems XGBoost is very efficient. XGBoost stands for eXtreme Gradient Boosting, and the goal of designing it was to make sure that every bit of memory and hardware resource was used for tree boosting (KDnuggets, 2017). I used XGBoost as my model for its efficiency, scalability, and parallel and distributed computing performance. I also used `RandomizedSearchCV` to estimate my training dataset. So I will be able to select the best parameter using `random_search.best_estimator_`. As I am solving a classification problem I imported `XGBClassifier`. The confusion matrix of my model gives me an insight into the performance of the XGBoost classification.



```
Confusion Matrix:  
[[25  1]  
 [ 2 18]]
```

The confusion matrix gives us the summary of the classification problem. It is parted into four parts

1. True Negative
2. False Positive
3. False Negative
4. True Positive

In our confusion matrix:

1. True Negative = 25
2. False Positive = 1
3. False Negative = 2
4. True Positive = 18

Finally, we obtain our accuracy score which is 0.93 means our model is 93% accurate in prediction.

We will look into cross-validation after explaining the regression model.

### Linear Regression:

As I said earlier that the task associated with the dataset is classification and regression. We have done our classification part now I will implement a linear regression model for further prediction.

I already selected an important feature for XGBoost classification where I only took 10 columns. In terms of linear regression, I took 26 features out of 36 features. For linear regression, I use logreg fit to make the data scalable.

For this model, I obtain a confusion matrix that is

```
array([[28,  0],
       [ 4, 14]])
```

I got a 0.89 accuracy score that means my prediction model is 89% accurate in predicting the data.

### Cross-Validation with other models:

Cross-validation aims to understand that the models can predict new data which was not used in predicting it. It also helps the model to be safe from overfitting in a predictive model. I used five other models to cross-validate my score

1. LogisticRegression
2. GaussianNB
3. KNeighborsClassifier
4. RandomForestClassifier
5. DecisionTreeClassifier

After executing cross-validation I got a confusion matrix and accuracy score as previous.

```
Naive Bayes
[[28 14]
 [ 0  4]]
0.6956521739130435
```

```
Decision Tree
[[26  3]
 [ 2 15]]
0.8913043478260869
```

```
LogisticRegression
[[28  4]
 [ 0 14]]
0.9130434782608695
```

```
RandomForest
[[27  4]
 [ 1 14]]
0.8913043478260869
```

```
KNN
[[26  5]
 [ 2 13]]
0.8478260869565217
```

Here it's showing that cross validation accuracy from different models are

1. Logistic Regression: 89%
2. Naive Bayes: 82%
3. RandomForest: 89%
4. Decision Tree: 84%
5. KNN: 93%

From my prediction, I got 93% accuracy from XGBClassifier and 89% accuracy from Linear Regression.

## References

Admin, J., 2018. *Analytics Vidhya*. [Online]

Available at:

<https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/>

[Accessed 2 12 2021].

KDnuggets, 2017. *KDnuggets*. [Online]

Available at:

<https://www.kdnuggets.com/2017/10/xgboost-top-machine-learning-method-kaggle-explained.html>

[Accessed 2 12 2021].

Sikora, M., 21. *UCI-Machine-Learning-Repository*. [Online]

Available at: <https://archive.ics.uci.edu/ml/datasets/Bone+marrow+transplant%3A+children>

[Accessed 01 12 2021].

sklearn.preprocessing.StandardScaler, 2021. *Scikit-Learn*. [Online]

Available at:

<https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html#sklearn-preprocessing-standardscaler>

[Accessed 2 12 2021].

Presentation Video Link: