```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
# Import Dependencies -To see the graphs in the notebook.
%matplotlib inline

# Python Imports
import math,time,random,datetime

# Data Manipulation
import numpy as np
import pandas as pd

# Visualization -This is where the graphs come in.
import matplotlib.pyplot as plt
import seaborn as sns
import missingno
plt.style.use('ggplot')

## Statistical Analysis
from scipy import stats
from scipy.stats import norm,skew

# Display all Columns
pd.set_option('display.max_columns', None)

# Ignore Warnings
import warnings
warnings.filterwarnings('ignore')


# Import the train and test data.
train = pd.read_csv('/content/drive/MyDrive/genesight data/Genetic-Disorder-Prediction-main/Genetic-Disorder-Prediction-main/dataset/train.c
test = pd.read_csv('/content/drive/MyDrive/genesight data/Genetic-Disorder-Prediction-main/Genetic-Disorder-Prediction-main/dataset/test.csv


## Understanding the Division of Genetic Disorders.
table1 = pd.pivot_table(train, index= ['Genetic Disorder', 'Disorder Subclass'], values= 'Patient Age', aggfunc= np.mean)
table1
```

		Patient Age
Genetic Disorder	**Disorder Subclass**	
Mitochondrial genetic inheritance disorders	**Leber's hereditary optic neuropathy**	6.748644
	Leigh syndrome	6.946026
	Mitochondrial myopathy	6.955514
Multifactorial genetic inheritance disorders	**Alzheimer's**	6.358779
	Cancer	7.616279
	Diabetes	6.907623
Single-gene inheritance diseases	**Cystic fibrosis**	6.925110
	Hemochromatosis	6.856771
	Tay-Sachs	7.096241

```
# This loop assign these values to the Missing Values according to the Disorder Subclass.
for i in range(0,22083):
    if (train['Genetic Disorder'].isnull()[i] == True):
        if (train['Disorder Subclass'][i] == "Leber's hereditary optic neuropathy"):
          train['Genetic Disorder'][i] = 'Mitochondrial genetic inheritance disorders'
        elif (train['Disorder Subclass'][i] == 'Leigh syndrome'):
          train['Genetic Disorder'][i] = 'Mitochondrial genetic inheritance disorders'
        elif (train['Disorder Subclass'][i] == 'Mitochondrial myopathy'):
          train['Genetic Disorder'][i] = 'Mitochondrial genetic inheritance disorders'
        elif (train['Disorder Subclass'][i] == "Alzheimer's"):
          train['Genetic Disorder'][i] = 'Multifactorial genetic inheritance disorders'
        elif (train['Disorder Subclass'][i] == 'Cancer'):
          train['Genetic Disorder'][i] = 'Multifactorial genetic inheritance disorders'
        elif (train['Disorder Subclass'][i] == 'Diabetes'):
```

```
          train['Genetic Disorder'][i] = 'Multifactorial genetic inheritance disorders'
        elif (train['Disorder Subclass'][i] == "Cystic fibrosis"):
          train['Genetic Disorder'][i] = 'Single-gene inheritance diseases'
        elif (train['Disorder Subclass'][i] == 'Hemochromatosis'):
          train['Genetic Disorder'][i] = 'Single-gene inheritance diseases'
        elif (train['Disorder Subclass'][i] == 'Tay-Sachs'):
          train['Genetic Disorder'][i] = 'Single-gene inheritance diseases'
        else:
          pass
    else:
      continue


## Understanding the Division of Genetic Disorders.
table2 = pd.pivot_table(train, index= ['Genetic Disorder', 'Disorder Subclass'], values= 'Patient Id', aggfunc= 'count')
table2
```

Genetic Disorder	Disorder Subclass	Patient Id
Mitochondrial genetic inheritance disorders	**Leber's hereditary optic neuropathy**	648
	Leigh syndrome	5160
	Mitochondrial myopathy	4405
Multifactorial genetic inheritance disorders	**Alzheimer's**	152
	Cancer	97
	Diabetes	1817
Single-gene inheritance diseases	**Cystic fibrosis**	3448
	Hemochromatosis	1355
	Tay-Sachs	2833

```
# This loop assign these values to the Missing Values according to the Most Frequent Disorder Subclass for each Genetic Disorder.
for i in range(0,22083):
    if (train['Disorder Subclass'].isnull()[i] == True):
        if (train['Genetic Disorder'][i] == "Mitochondrial genetic inheritance disorders"):
          train['Disorder Subclass'][i] = 'Leigh syndrome'
        elif (train['Genetic Disorder'][i] == "Multifactorial genetic inheritance disorders"):
          train['Disorder Subclass'][i] = 'Diabetes'
        elif (train['Genetic Disorder'][i] == "Single-gene inheritance diseases"):
          train['Disorder Subclass'][i] = 'Cystic fibrosis'
        else:
          pass
    else:
      continue


print(f'The Missing Values in Disorder Subclass Column is: {train["Disorder Subclass"].isnull().sum()}')
print(f'The Missing Values in Genetic Disorder Column is: {train["Genetic Disorder"].isnull().sum()}')
```

```
The Missing Values in Disorder Subclass Column is: 278
The Missing Values in Genetic Disorder Column is: 278
```

```
## Dropping the Missing Values from the Target Variables.
train.dropna(subset=['Genetic Disorder', "Disorder Subclass"], inplace= True)


train.shape
```

```
(21805, 45)
```

```
## Proper Indexing
train.insert(0, '', range(0, 0 + len(train)))
train.set_index('', inplace= True)


## Dropping all Unnecessary Columns
train.drop(columns= ['Patient Id', 'Place of birth', 'H/O serious maternal illness', 'Patient First Name', 'Family Name', "Father's name",
                     'Institute Name', 'Location of Institute'], inplace= True)

test.drop(columns= ['Patient Id', 'Place of birth', 'H/O serious maternal illness', 'Patient First Name', 'Family Name', "Father's name",
                     'Institute Name', 'Location of Institute'], inplace= True)
```

```
train.shape
```

```
(21805, 37)
```

```
train['Patient Age'].describe()
```

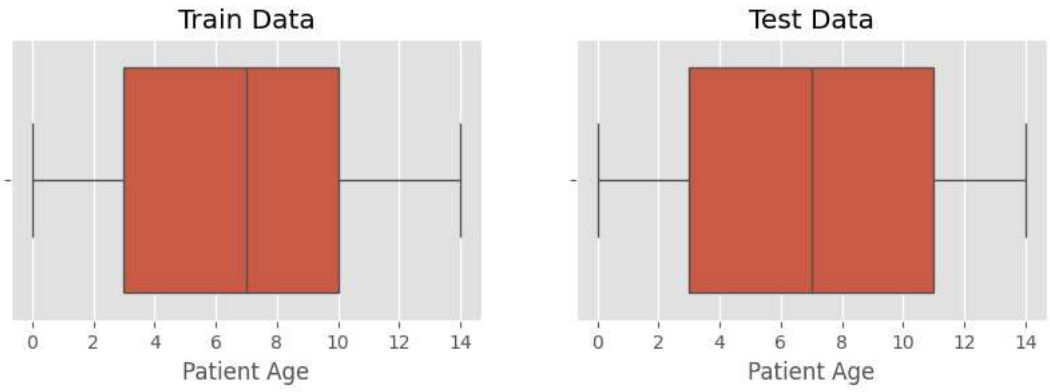
	Patient Age
count	20397.000000
mean	6.972349
std	4.317753
min	0.000000
25%	3.000000
50%	7.000000
75%	11.000000
max	14.000000

dtype: float64

```
## Filling the Missing Values with the Average Mean.
train['Patient Age'].fillna(round(train['Patient Age'].mean()), inplace= True)


## Missing Values
train['Patient Age'].isnull().any()
```

```
False
```

```
## Checking for Outliers
plt.figure(figsize= (10,6))

plt.subplot(2,2,1)
sns.boxplot(x= 'Patient Age', data= train)
plt.title('Train Data')
plt.subplot(2,2,2)
sns.boxplot(x= 'Patient Age', data= test)
plt.title('Test Data')

plt.show()
```




```
train['Inherited from father'].unique()
```

```
array(['No', 'Yes', nan], dtype=object)
```

```
## Filling Missing Values with 'No'
#### train['Inherited from father'].fillna('No', inplace= True)
#### test['Inherited from father'].fillna('No', inplace= True)

### Train Data
for i in range(0,21805):
    if (train['Inherited from father'].isnull()[i] == True):
        if (train['Paternal gene'][i] == "Yes"):
```

```
            train['Inherited from father'][i] = 'Yes'
        elif (train['Paternal gene'][i] == "No"):
            train['Inherited from father'][i] = 'No'
        else:
            train['Inherited from father'][i] = 'No'
    else:
      continue


### Test Data
for i in range(0,9465):
    if (test['Inherited from father'].isnull()[i] == True):
        if (test['Paternal gene'][i] == "Yes"):
            test['Inherited from father'][i] = 'Yes'
        elif (test['Paternal gene'][i] == "No"):
            test['Inherited from father'][i] = 'No'
        else:
            test['Inherited from father'][i] = 'No'
    else:
      continue


train['Inherited from father'].value_counts()
```

Inherited from father	count
No	13122
Yes	8683

dtype: int64

```
train['Maternal gene'].unique()
```

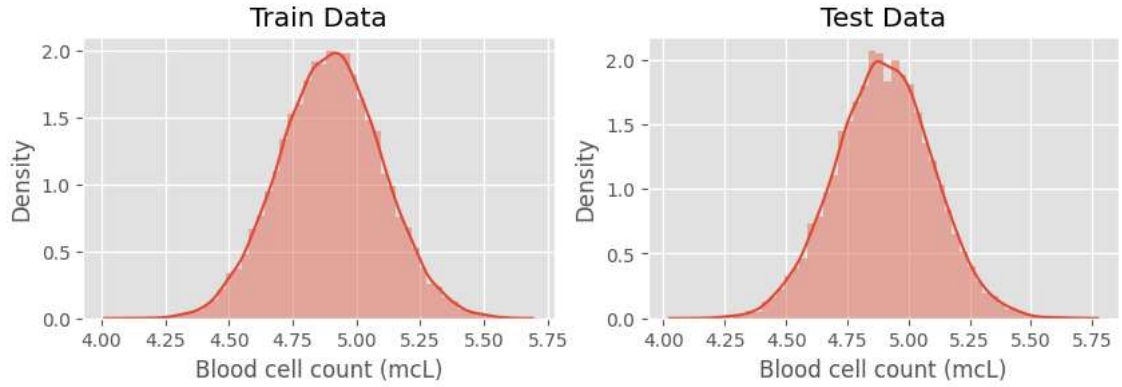
```
array(['Yes', 'No', nan], dtype=object)
```

```
## Filling Missing Values with 'No'
## train['Maternal gene'].fillna('No', inplace= True)
## test['Maternal gene'].fillna('No', inplace= True)

### Train Data
for i in range(0,21805):
    if (train['Maternal gene'].isnull()[i] == True):
        if (train["Genes in mother's side"][i] == "Yes"):
            train['Maternal gene'][i] = 'Yes'
        elif (train["Genes in mother's side"][i] == "No"):
            train['Maternal gene'][i] = 'No'
        else:
            train['Maternal gene'][i] = 'No'
    else:
      continue


### Test Data
for i in range(0,9465):
    if (test['Maternal gene'].isnull()[i] == True):
        if (test["Genes in mother's side"][i] == "Yes"):
            test['Maternal gene'][i] = 'Yes'
        elif (test["Genes in mother's side"][i] == "No"):
            test['Maternal gene'][i] = 'No'
        else:
            test['Maternal gene'][i] = 'No'
    else:
      continue


train['Maternal gene'].value_counts()
```

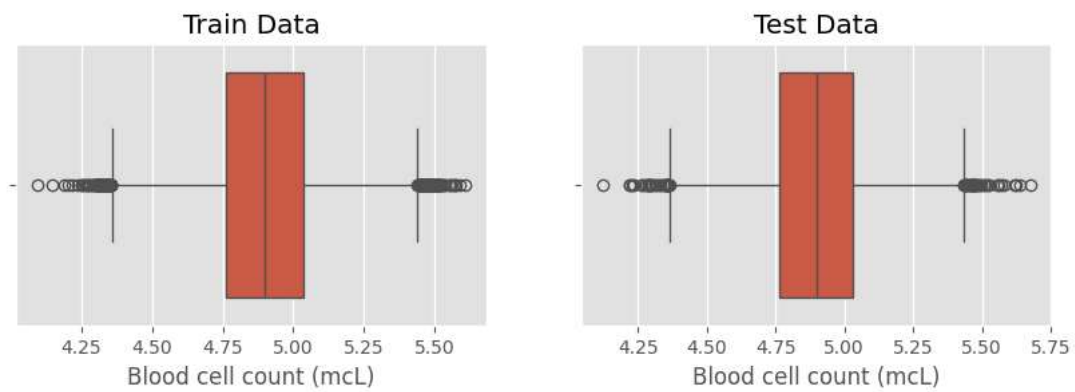
Maternal gene	count
Yes	12174
No	9631

dtype: int64

```
## Checking the Distribution
plt.figure(figsize= (10,6))

plt.subplot(2,2,1)
sns.distplot(train['Blood cell count (mcL)'])
plt.title('Train Data')
plt.subplot(2,2,2)
sns.distplot(test['Blood cell count (mcL)'])
plt.title('Test Data')

plt.show()
```




```
## Checking for Outliers
plt.figure(figsize= (10,6))

plt.subplot(2,2,1)
sns.boxplot(x= 'Blood cell count (mcL)', data= train)
plt.title('Train Data')
plt.subplot(2,2,2)
sns.boxplot(x= 'Blood cell count (mcL)', data= test)
plt.title('Test Data')

plt.show()
```




```
## Z-Score --> When Feature follows Normal Distibution.

### Train Dataset
mean = train['Blood cell count (mcL)'].mean()
std = train['Blood cell count (mcL)'].std()

Upper_Bound = mean + (3*std)
Lower_Bound = mean - (3*std)
```

```
train['Blood cell count (mcL)'][train['Blood cell count (mcL)'] < Lower_Bound] = Lower_Bound
train['Blood cell count (mcL)'][train['Blood cell count (mcL)'] > Upper_Bound] = Upper_Bound
```

```
### Test Dataset
mean = test['Blood cell count (mcL)'].mean()
std = test['Blood cell count (mcL)'].std()

Upper_Bound = mean + (3*std)
Lower_Bound = mean - (3*std)

test['Blood cell count (mcL)'][test['Blood cell count (mcL)'] < Lower_Bound] = Lower_Bound
test['Blood cell count (mcL)'][test['Blood cell count (mcL)'] > Upper_Bound] = Upper_Bound
```

```
## Filling the Missing Values with the Average Mean.
train["Mother's age"].fillna(round(train["Mother's age"].mean()), inplace= True)

## Filling the Missing Values with the Average Mean.
train["Father's age"].fillna(round(train["Father's age"].mean()), inplace= True)
```

```
## Filling Missing Values with 'Normal (30-60)'
## train['Respiratory Rate (breaths/min)'].fillna('Normal (30-60)', inplace= True)
test['Respiratory Rate (breaths/min)'].replace('-99', np.nan, inplace= True)
## test['Respiratory Rate (breaths/min)'].fillna('Normal (30-60)', inplace= True)
```

```
### Train Data
for i in range(0,21805):
    if (train['Respiratory Rate (breaths/min)'].isnull()[i] == True):
        if (train["Status"][i] == "Alive"):
          train['Respiratory Rate (breaths/min)'][i] = 'Normal (30-60)'
        else:
          train['Respiratory Rate (breaths/min)'][i] = 'Tachypnea'
    else:
      continue
```

```
### Test Data
for i in range(0,9465):
    if (test['Respiratory Rate (breaths/min)'].isnull()[i] == True):
        if (test["Status"][i] == "Alive"):
          test['Respiratory Rate (breaths/min)'][i] = 'Normal (30-60)'
        else:
          test['Respiratory Rate (breaths/min)'][i] = 'Tachypnea'
    else:
      continue
```

```
train['Respiratory Rate (breaths/min)'].value_counts()
```

Respiratory Rate (breaths/min)	count
Normal (30-60)	10991
Tachypnea	10814

dtype: int64

```
## Filling Missing Values with 'Normal'
## train['Heart Rate (rates/min'].fillna('Normal', inplace= True)
test['Heart Rate (rates/min'].replace('-99', np.nan, inplace= True)
## test['Heart Rate (rates/min'].fillna('Normal', inplace= True)
```

```
### Train Data
for i in range(0,21805):
    if (train['Heart Rate (rates/min'].isnull()[i] == True):
        if (train["Respiratory Rate (breaths/min)"][i] == "Normal (30-60)"):
          train['Heart Rate (rates/min'][i] = 'Normal'
        else:
          train['Heart Rate (rates/min'][i] = 'Tachycardia'
```

```
    else:
      continue


### Test Data
for i in range(0,9465):
    if (test['Heart Rate (rates/min'].isnull()[i] == True):
        if (test["Respiratory Rate (breaths/min)"][i] == "Normal (30-60)"):
          test['Heart Rate (rates/min'][i] = 'Normal'
        else:
          test['Heart Rate (rates/min'][i] = 'Tachycardia'
    else:
      continue


train['Heart Rate (rates/min'].value_counts()
```

Heart Rate (rates/min	count
Normal	11156
Tachycardia	10649

dtype: int64

```
train.drop(columns= ['Test 1', 'Test 2', 'Test 3', 'Test 4', 'Test 5'], inplace= True)
test.drop(columns= ['Test 1', 'Test 2', 'Test 3', 'Test 4', 'Test 5'], inplace= True)


## Filling Missing Values with 'No'
train['Parental consent'].fillna('No', inplace= True)
test['Parental consent'].replace('-99', np.nan, inplace= True)
test['Parental consent'].fillna('No', inplace= True)


train['Parental consent'].value_counts()
```

Parental consent	count
Yes	19719
No	2086

dtype: int64

```
# This loop is for Train Data.
for i in range(0,21805):
    if (train['Follow-up'].isnull()[i] == True):
        if (train['Respiratory Rate (breaths/min)'][i] == "Tachypnea"):
          train['Follow-up'][i] = 'High'
        elif (train['Heart Rate (rates/min'][i] == "Tachycardia"):
          train['Follow-up'][i] = 'High'
        else:
          train['Follow-up'][i] = 'Low'
    else:
      continue



test['Follow-up'].replace('-99', np.nan, inplace= True)
# This loop is for Test Data.
for i in range(0,9465):
    if (test['Follow-up'].isnull()[i] == True):
        if (test['Respiratory Rate (breaths/min)'][i] == "Tachypnea"):
          test['Follow-up'][i] = 'High'
        elif (test['Heart Rate (rates/min'][i] == "Tachycardia"):
          test['Follow-up'][i] = 'High'
        else:
          test['Follow-up'][i] = 'Low'
    else:
      continue
```

```
train['Follow-up'].value_counts()
```

Follow-up	count
High	11248
Low	10557

dtype: int64

```
train["Gender"].mode()
```

	Gender
0	Ambiguous

dtype: object

```
## Filling Missing Values with 'Ambiguous'
train['Gender'].fillna('Ambiguous', inplace= True)
test['Gender'].replace('-99', np.nan, inplace= True)
test['Gender'].fillna('Ambiguous', inplace= True)


### Train Data
train["Birth asphyxia"].replace('No record', 'Not available', inplace= True)
train["Birth asphyxia"].fillna('No', inplace= True)


### Test Data
test["Birth asphyxia"].replace('No record', 'Not available', inplace= True)
test["Birth asphyxia"].replace('-99', np.nan, inplace= True)
test["Birth asphyxia"].fillna('No', inplace= True)


### Train Data
train["Autopsy shows birth defect (if applicable)"].replace('None', 'No', inplace= True)

for i in range(0,21805):
    if (train["Autopsy shows birth defect (if applicable)"].isnull()[i] == True):
        if (train['Inherited from father'][i] == "Yes"):
          train["Autopsy shows birth defect (if applicable)"][i] = 'Yes'
        else:
          train["Autopsy shows birth defect (if applicable)"][i] = 'No'
    else:
      continue



### Test Data
test["Autopsy shows birth defect (if applicable)"].replace('-99', np.nan, inplace= True)
test["Autopsy shows birth defect (if applicable)"].replace('None', 'No', inplace= True)

for i in range(0,9465):
    if (test["Autopsy shows birth defect (if applicable)"].isnull()[i] == True):
        if (test['Inherited from father'][i] == "Yes"):
          test["Autopsy shows birth defect (if applicable)"][i] = 'Yes'
        else:
          test["Autopsy shows birth defect (if applicable)"][i] = 'No'
    else:
      continue


train["Autopsy shows birth defect (if applicable)"].value_counts()
```

Autopsy shows birth defect (if applicable)	count
Not applicable	10955
No	5758
Yes	5092

dtype: int64

```
train["Folic acid details (peri-conceptional)"].fillna('No', inplace= True)

test["Folic acid details (peri-conceptional)"].replace('-99', np.nan, inplace= True)
test["Folic acid details (peri-conceptional)"].fillna('No', inplace= True)


train["Folic acid details (peri-conceptional)"].value_counts()
```

Folic acid details (peri-conceptional)	count
No	11840
Yes	9965

dtype: int64

```
train["H/O radiation exposure (x-ray)"].fillna('No', inplace= True)
train["H/O radiation exposure (x-ray)"].replace('-', 'No', inplace= True)
train["H/O radiation exposure (x-ray)"].replace('Not applicable', 'No', inplace= True)


test["H/O radiation exposure (x-ray)"].replace('-99', np.nan, inplace= True)
test["H/O radiation exposure (x-ray)"].fillna('No', inplace= True)
test["H/O radiation exposure (x-ray)"].replace('-', 'No', inplace= True)
test["H/O radiation exposure (x-ray)"].replace('Not applicable', 'No', inplace= True)


train["H/O radiation exposure (x-ray)"].value_counts()
```

H/O radiation exposure (x-ray)	count
No	16881
Yes	4924

dtype: int64

```
train["H/O substance abuse"].fillna('No', inplace= True)
train["H/O substance abuse"].replace('-', 'No', inplace= True)
train["H/O substance abuse"].replace('Not applicable', 'No', inplace= True)


test["H/O substance abuse"].replace('-99', np.nan, inplace= True)
test["H/O substance abuse"].fillna('No', inplace= True)
test["H/O substance abuse"].replace('-', 'No', inplace= True)
test["H/O substance abuse"].replace('Not applicable', 'No', inplace= True)


train["H/O substance abuse"].value_counts()
```

H/O substance abuse	count
No	16891
Yes	4914

dtype: int64

```
test["Assisted conception IVF/ART"].replace('-99', np.nan, inplace= True)

# This loop is for test Data.
for i in range(0,21805):
    if (train['Assisted conception IVF/ART'].isnull()[i] == True):
        if (train['Folic acid details (peri-conceptional)'][i] == "Yes"):
          train['Assisted conception IVF/ART'][i] = 'Yes'
        else:
          train['Assisted conception IVF/ART'][i] = 'No'
    else:
      continue



# This loop is for Test Data.
for i in range(0,9465):
    if (test['Assisted conception IVF/ART'].isnull()[i] == True):
        if (test['Folic acid details (peri-conceptional)'][i] == "Yes"):
          test['Assisted conception IVF/ART'][i] = 'Yes'
        else:
          test['Assisted conception IVF/ART'][i] = 'No'
    else:
      continue


train["Assisted conception IVF/ART"].value_counts()
```

Assisted conception IVF/ART	count
No	10969
Yes	10836

dtype: int64

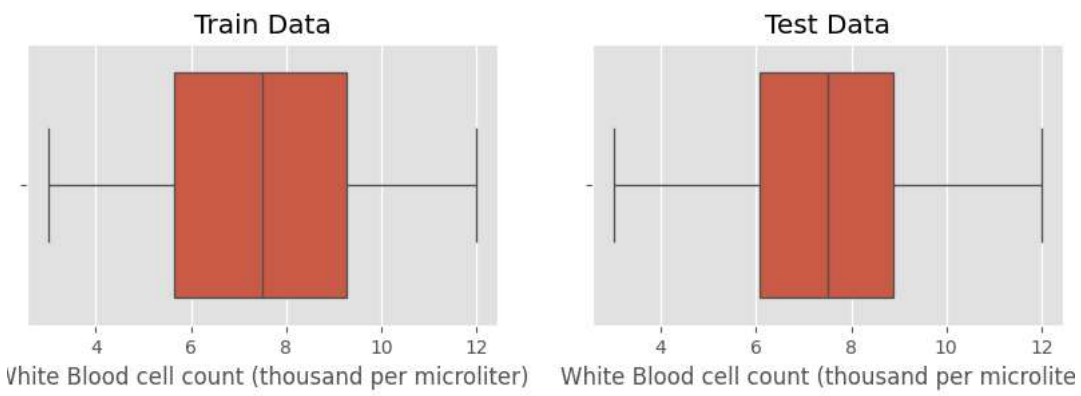
```
train["History of anomalies in previous pregnancies"].fillna('No', inplace= True)

test["History of anomalies in previous pregnancies"].replace('-99', np.nan, inplace= True)
test["History of anomalies in previous pregnancies"].fillna('No', inplace= True)


train["No. of previous abortion"].fillna(0, inplace= True)

test["No. of previous abortion"].replace(-99, 0, inplace= True)


test["Birth defects"].replace('-99', np.nan, inplace= True)

# This loop is for test Data.
for i in range(0,21805):
    if (train['Birth defects'].isnull()[i] == True):
        if (train["Mother's age"][i] > 34):
          if (train["H/O substance abuse"][i] == 'Yes'):
            train['Birth defects'][i] = 'Multiple'
          else:
            train['Birth defects'][i] = 'Singular'
        else:
            train['Birth defects'][i] = 'Singular'
    else:
      continue



# This loop is for Test Data.
for i in range(0,9465):
    if (test['Birth defects'].isnull()[i] == True):
        if (test["Mother's age"][i] > 34):
          if (test["H/O substance abuse"][i] == 'Yes'):
            test['Birth defects'][i] = 'Multiple'
          else:
            test['Birth defects'][i] = 'Singular'
        else:
            test['Birth defects'][i] = 'Singular'
    else:
      continue
```

```
train["Birth defects"].value_counts()
```

Birth defects	count
Singular	11647
Multiple	10158

dtype: int64

```
train["White Blood cell count (thousand per microliter)"].fillna(train["White Blood cell count (thousand per microliter)"].mean(), inplace=

test["White Blood cell count (thousand per microliter)"].replace(-99, np.nan, inplace= True)
test["White Blood cell count (thousand per microliter)"].fillna(test["White Blood cell count (thousand per microliter)"].mean(), inplace= Tr


## Checking for Outliers
plt.figure(figsize= (10,6))

plt.subplot(2,2,1)
sns.boxplot(x= 'White Blood cell count (thousand per microliter)', data= train)
plt.title('Train Data')
plt.subplot(2,2,2)
sns.boxplot(x= 'White Blood cell count (thousand per microliter)', data= test)
plt.title('Test Data')

plt.show()
```




```
## Understanding the Division of Genetic Disorders.
table3 = pd.pivot_table(train, index= ['Blood test result'],
                        values= 'White Blood cell count (thousand per microliter)', aggfunc= np.mean)
table3
```

Blood test result	White Blood cell count (thousand per microliter)
abnormal	7.462121
inconclusive	7.482236
normal	7.502300
slightly abnormal	7.499233

```
## train["Blood test result"].fillna('slightly abnormal', inplace= True)
## test["Blood test result"].fillna('slightly abnormal', inplace= True)

test["Blood test result"].replace('-99', np.nan, inplace= True)

# This loop is for test Data.
for i in range(0,21805):
    if (train['Blood test result'].isnull()[i] == True):
        if (5 < train["White Blood cell count (thousand per microliter)"][i] < 10):
          train['Blood test result'][i] = 'normal'
        elif (10 < train["White Blood cell count (thousand per microliter)"][i] < 1):
          train['Blood test result'][i] = 'abnormal'
        elif (1 < train["White Blood cell count (thousand per microliter)"][i] > 4):
```

```
            train['Blood test result'][i] = 'slightly abnormal'
          else:
              train['Blood test result'][i] = 'inconclusive'
      else:
        continue



# This loop is for Test Data.
for i in range(0,9465):
    if (test['Blood test result'].isnull()[i] == True):
        if (5 < test["White Blood cell count (thousand per microliter)"][i] < 10):
          test['Blood test result'][i] = 'normal'
        elif (10 < test["White Blood cell count (thousand per microliter)"][i] < 1):
          test['Blood test result'][i] = 'abnormal'
        elif (1 < test["White Blood cell count (thousand per microliter)"][i] > 4):
          test['Blood test result'][i] = 'slightly abnormal'
        else:
            test['Blood test result'][i] = 'inconclusive'
    else:
      continue


train["Blood test result"].value_counts()
```

Blood test result	count
normal	6298
slightly abnormal	5562
inconclusive	5104
abnormal	4841

dtype: int64

```
train["Symptom 1"].fillna(0, inplace= True)
train["Symptom 2"].fillna(0, inplace= True)
train["Symptom 3"].fillna(0, inplace= True)
train["Symptom 4"].fillna(0, inplace= True)
train["Symptom 5"].fillna(0, inplace= True)


## Replacing Boolean values
test["Symptom 1"] = np.where(test["Symptom 1"] == True, 1, 0)
test["Symptom 2"] = np.where(test["Symptom 2"] == True, 1, 0)
test["Symptom 3"] = np.where(test["Symptom 3"] == True, 1, 0)
test["Symptom 4"] = np.where(test["Symptom 4"] == True, 1, 0)
test["Symptom 5"] = np.where(test["Symptom 5"] == True, 1, 0)


train.head(25)
```

	Patient Age	Genes in mother's side	Inherited from father	Maternal gene	Paternal gene	Blood cell count (mcL)	Mother's age	Father's age	Status	Respiratory Rate (breaths/min)	Heart Rate (rates/min	Parental consent	Fol
0	2.0	Yes	No	Yes	No	4.760603	35.0	42.0	Alive	Normal (30-60)	Normal	Yes	
1	4.0	Yes	Yes	No	No	4.910669	35.0	23.0	Deceased	Tachypnea	Normal	Yes	
2	6.0	Yes	No	No	No	4.893297	41.0	22.0	Alive	Normal (30-60)	Tachycardia	Yes	
3	12.0	Yes	No	Yes	No	4.705280	21.0	42.0	Deceased	Tachypnea	Normal	Yes	
4	11.0	Yes	No	Yes	Yes	4.720703	32.0	42.0	Alive	Tachypnea	Tachycardia	No	
5	14.0	Yes	No	Yes	No	5.103188	35.0	42.0	Deceased	Tachypnea	Normal	Yes	
6	3.0	Yes	No	Yes	Yes	4.901080	35.0	63.0	Alive	Normal (30-60)	Normal	No	
7	3.0	No	No	Yes	Yes	4.964816	40.0	42.0	Alive	Tachypnea	Normal	Yes	
8	11.0	No	No	Yes	No	5.209058	45.0	44.0	Alive	Tachypnea	Tachycardia	Yes	
9	4.0	No	Yes	Yes	Yes	4.752272	44.0	42.0	Alive	Tachypnea	Tachycardia	Yes	
10	6.0	Yes	No	Yes	No	4.750824	35.0	42.0	Deceased	Tachypnea	Tachycardia	Yes	
11	7.0	No	No	No	Yes	4.848795	35.0	42.0	Deceased	Normal (30-60)	Tachycardia	Yes	

```
# Obtaining the Categorical Columns
categorical_features = [features for features in train.columns if train[features].dtypes == 'O']
categorical_features
```

```
["Genes in mother's side",
 'Inherited from father',
 'Maternal gene',
 'Paternal gene',
 'Status',
 'Respiratory Rate (breaths/min)',
 'Heart Rate (rates/min',
 'Parental consent',
 'Follow-up',
 'Gender',
 'Birth asphyxia',
 'Autopsy shows birth defect (if applicable)',
 'Folic acid details (peri-conceptional)',
 'H/O radiation exposure (x-ray)',
 'H/O substance abuse',
 'Assisted conception IVF/ART',
 'History of anomalies in previous pregnancies',
 'Birth defects',
 'Blood test result',
 'Genetic Disorder',
 'Disorder Subclass']
```

```
# Obtaining the Categorical Columns
categorical_features_test = [features for features in test.columns if test[features].dtypes == 'O']
```

```
from sklearn.preprocessing import OneHotEncoder, LabelEncoder

label_encoder = LabelEncoder()
for feature in categorical_features:
    ## Encode labels in all Categorical Columns.
    train[feature]= label_encoder.fit_transform(train[feature])



for features in categorical_features_test:
    ## Encode labels in all Categorical Columns.
    test[features]= label_encoder.fit_transform(test[features])
```

```
train.head(25)
```

	Patient Age	Genes in mother's side	Inherited from father	Maternal gene	Paternal gene	Blood cell count (mcL)	Mother's age	Father's age	Status	Respiratory Rate (breaths/min)	Heart Rate (rates/min	Parental consent	Follow-up
0	2.0	1	0	1	0	4.760603	35.0	42.0	0	0	0	1	0
1	4.0	1	1	0	0	4.910669	35.0	23.0	1	1	0	1	0
2	6.0	1	0	0	0	4.893297	41.0	22.0	0	0	1	1	1
3	12.0	1	0	1	0	4.705280	21.0	42.0	1	1	0	1	0
4	11.0	1	0	1	1	4.720703	32.0	42.0	0	1	1	0	1
5	14.0	1	0	1	0	5.103188	35.0	42.0	1	1	0	1	1
6	3.0	1	0	1	1	4.901080	35.0	63.0	0	0	0	0	1
7	3.0	0	0	1	1	4.964816	40.0	42.0	0	1	0	1	1
8	11.0	0	0	1	0	5.209058	45.0	44.0	0	1	1	1	1
9	4.0	0	1	1	1	4.752272	44.0	42.0	0	1	1	1	1
10	6.0	1	0	1	0	4.750824	35.0	42.0	1	1	1	1	1
11	7.0	0	0	0	1	4.848795	35.0	42.0	1	0	1	1	1
12	1.0	1	1	0	0	4.612265	50.0	56.0	1	0	1	1	0
13	0.0	1	0	0	1	4.807778	28.0	42.0	1	1	1	1	1
14	6.0	1	0	1	0	4.620420	41.0	20.0	0	1	1	1	0
15	0.0	0	0	0	0	4.918570	30.0	24.0	0	0	1	0	0
16	0.0	1	1	0	0	4.798520	35.0	57.0	0	0	0	0	1
17	7.0	0	0	0	0	4.952457	24.0	24.0	1	1	0	0	0
18	10.0	1	1	1	0	4.751452	40.0	57.0	1	0	1	1	1
19	6.0	0	1	1	1	4.876896	36.0	48.0	1	1	0	1	1
20	2.0	0	0	1	0	4.808872	35.0	30.0	0	1	1	1	1
21	10.0	1	1	1	0	5.077720	30.0	42.0	1	1	1	1	1
22	7.0	1	0	0	0	4.998895	35.0	42.0	0	1	0	1	1
23	4.0	1	1	1	0	4.963480	35.0	55.0	0	1	1	1	1
24	5.0	1	1	1	1	4.661035	35.0	62.0	0	0	0	1	0

```
# Obtaining Numerical Columns
numerical_features = [features for features in train.columns if train[features].dtypes != 'O']
numerical_features
```

```
['Patient Age',
 "Genes in mother's side",
 'Inherited from father',
```

```
        'Maternal gene',
        'Paternal gene',
        'Blood cell count (mcL)',
        "Mother's age",
        "Father's age",
        'Status',
        'Respiratory Rate (breaths/min)',
        'Heart Rate (rates/min',
        'Parental consent',
        'Follow-up',
        'Gender',
        'Birth asphyxia',
        'Autopsy shows birth defect (if applicable)',
        'Folic acid details (peri-conceptional)',
        'H/O radiation exposure (x-ray)',
        'H/O substance abuse',
        'Assisted conception IVF/ART',
        'History of anomalies in previous pregnancies',
        'No. of previous abortion',
        'Birth defects',
        'White Blood cell count (thousand per microliter)',
        'Blood test result',
        'Symptom 1',
        'Symptom 2',
        'Symptom 3',
        'Symptom 4',
        'Symptom 5',
        'Genetic Disorder',
        'Disorder Subclass']


# Selecting all Features that need to be Scaled except the Target Variable.
scale_feature = [features for features in numerical_features if features not in ['Genetic Disorder', 'Disorder Subclass', "Genes in mother's
                                                                                 'Inherited from father', 'Maternal gene', 'Paternal gene',
                                                                                 'Status', 'Respiratory Rate (breaths/min)',
                                                                                 'Heart Rate (rates/min', 'Parental consent', 'Follow-up',
                                                                                 'Gender', 'Birth asphyxia',
                                                                                 'Autopsy shows birth defect (if applicable)',
                                                                                 'Folic acid details (peri-conceptional)',
                                                                                 'H/O radiation exposure (x-ray)', 'H/O substance abuse',
                                                                                 'Assisted conception IVF/ART', 'Symptom 1', 'Symptom 2',
                                                                                 'Symptom 3', 'Symptom 4', 'Symptom 5',
                                                                                 'History of anomalies in previous pregnancies',
                                                                                 'Birth defects', 'Blood test result']]

from sklearn.preprocessing import RobustScaler
scaler = RobustScaler()


# Transform the train set and add the 'Genetic Disorder', 'Disorder Subclass' Columns.
train = pd.concat([train[['Genetic Disorder', 'Disorder Subclass', "Genes in mother's side", 'Inherited from father', 'Maternal gene',
                          'Paternal gene', 'Status', 'Respiratory Rate (breaths/min)', 'Heart Rate (rates/min', 'Parental consent', 'Follow-
                          'Gender', 'Birth asphyxia', 'Autopsy shows birth defect (if applicable)', 'Folic acid details (peri-conceptional)'
                          'H/O radiation exposure (x-ray)', 'H/O substance abuse', 'Assisted conception IVF/ART', 'Symptom 1', 'Symptom 2',
                          'Symptom 3', 'Symptom 4', 'Symptom 5', 'History of anomalies in previous pregnancies', 'Birth defects',
                          'Blood test result']].reset_index(drop= True),
       pd.DataFrame(scaler.fit_transform(train[scale_feature]), columns= scale_feature)], axis= 1)
       ## Converting to Dataframe as after transform its an array.

train.head()
```

	Genetic Disorder	Disorder Subclass	Genes in mother's side	Inherited from father	Maternal gene	Paternal gene	Status	Respiratory Rate (breaths/min)	Heart Rate (rates/min	Parental consent	Follow-up	Gender	Birth asphyxia	s a
0	0	5	1	0	1	0	0	0	0	1	0	0	0	
1	2	2	1	1	0	0	1	1	0	1	0	0	0	
2	1	3	1	0	0	0	0	0	1	1	1	0	1	
3	0	6	1	0	1	0	1	1	0	1	0	2	1	
4	1	1	1	0	1	1	0	1	1	0	1	2	1	

```
# Selecting all Features that need to be Scaled except the Target Variable.
scale_feature = [features for features in numerical_features if features not in ['Genetic Disorder', 'Disorder Subclass', "Genes in mother's
```

```
                                                'Inherited from father', 'Maternal gene', 'Paternal gene',
                                                'Status', 'Respiratory Rate (breaths/min)',
                                                'Heart Rate (rates/min', 'Parental consent', 'Follow-up',
                                                'Gender', 'Birth asphyxia',
                                                'Autopsy shows birth defect (if applicable)',
                                                'Folic acid details (peri-conceptional)',
                                                'H/O radiation exposure (x-ray)', 'H/O substance abuse',
                                                'Assisted conception IVF/ART', 'Symptom 1', 'Symptom 2',
                                                'Symptom 3', 'Symptom 4', 'Symptom 5',
                                                'History of anomalies in previous pregnancies',
                                                'Birth defects', 'Blood test result']]
```

```
from sklearn.preprocessing import RobustScaler
scaler = RobustScaler()


# Transform the test set and add the remaining Column.
test = pd.concat([test[["Genes in mother's side", 'Inherited from father', 'Maternal gene',
                        'Paternal gene', 'Status', 'Respiratory Rate (breaths/min)', 'Heart Rate (rates/min', 'Parental consent',
                        'Follow-up', 'Gender', 'Birth asphyxia', 'Autopsy shows birth defect (if applicable)',
                        'Folic acid details (peri-conceptional)', 'H/O radiation exposure (x-ray)', 'H/O substance abuse',
                        'Assisted conception IVF/ART', 'Symptom 1', 'Symptom 2', 'Symptom 3', 'Symptom 4', 'Symptom 5',
                        'History of anomalies in previous pregnancies', 'Birth defects', 'Blood test result']].reset_index(drop= True),
       pd.DataFrame(scaler.fit_transform(test[scale_feature]), columns= scale_feature)], axis= 1)
       ## Converting to Dataframe as after transform its an array.

test.head()
```

	Genes in mother's side	Inherited from father	Maternal gene	Paternal gene	Status	Respiratory Rate (breaths/min)	Heart Rate (rates/min	Parental consent	Follow-up	Gender	Birth asphyxia	Autopsy shows birth defect (if applicable)	Folic de (concepti
0	0	1	0	0	0	1	0	0	1	2	2	1	
1	1	0	1	1	0	0	0	1	1	2	2	1	
2	0	0	0	0	1	1	0	0	1	0	1	0	
3	0	1	1	0	0	0	0	0	1	0	0	1	
4	0	1	0	1	1	1	1	1	1	0	0	2	

```
## Capture the Independent Variables
X = train.drop(columns= ['Genetic Disorder', 'Disorder Subclass'], axis= 1)

## Capture the Dependent Variable
y1 = train['Genetic Disorder']
y2 = train['Disorder Subclass']


from sklearn.ensemble import ExtraTreesClassifier

## Genetic Disorder
model = ExtraTreesClassifier()
model.fit(X,y1)

## Disorder Subclass
model2 = ExtraTreesClassifier()
model2.fit(X,y2)
```

```
ExtraTreesClassifier
ExtraTreesClassifier()
```

```
print('Y1')
print(model.feature_importances_)
print()
print('Y2')
print(model2.feature_importances_)
```

```
Y1
[0.01988713 0.02161216 0.02321659 0.02390907 0.02406634 0.0298637
 0.03040084 0.01645434 0.02870018 0.04057402 0.04106365 0.02896194
 0.03021546 0.02480366 0.02432444 0.0302497  0.02318277 0.02549665
 0.02688267 0.02953338 0.03055735 0.02942801 0.0306339  0.04466615
```

```
 0.05448456 0.05654191 0.05305208 0.05305848 0.04802212 0.05615676]

Y2
[0.01952394 0.01968313 0.01977513 0.02144577 0.02294765 0.03090455
 0.03060902 0.01577946 0.02917377 0.04082674 0.04114868 0.02780857
 0.0306416  0.02395055 0.0234815  0.02974292 0.021195   0.02474221
 0.02631744 0.02925733 0.03727332 0.03136536 0.02934429 0.04646388
 0.05514171 0.05714436 0.05351809 0.05469552 0.04897653 0.05712197]
```

```
ranked_features = pd.Series(model.feature_importances_, index= X.columns)
ranked_features.nlargest(19)
```

	0
Blood cell count (mcL)	0.056542
White Blood cell count (thousand per microliter)	0.056157
Patient Age	0.054485
Father's age	0.053058
Mother's age	0.053052
No. of previous abortion	0.048022
Blood test result	0.044666
Birth asphyxia	0.041064
Gender	0.040574
Birth defects	0.030634
Symptom 5	0.030557
Heart Rate (rates/min	0.030401
Assisted conception IVF/ART	0.030250
Folic acid details (peri-conceptional)	0.030215
Respiratory Rate (breaths/min)	0.029864
Symptom 4	0.029533
History of anomalies in previous pregnancies	0.029428
Autopsy shows birth defect (if applicable)	0.028962
Follow-up	0.028700

dtype: float64

```
ranked_features2 = pd.Series(model2.feature_importances_, index= X.columns)
ranked_features2.nlargest(19)
```

	0
Blood cell count (mcL)	0.057144
White Blood cell count (thousand per microliter)	0.057122
Patient Age	0.055142
Father's age	0.054696
Mother's age	0.053518
No. of previous abortion	0.048977
Blood test result	0.046464
Birth asphyxia	0.041149
Gender	0.040827
Symptom 5	0.037273
History of anomalies in previous pregnancies	0.031365
Respiratory Rate (breaths/min)	0.030905
Folic acid details (peri-conceptional)	0.030642
Heart Rate (rates/min	0.030609
Assisted conception IVF/ART	0.029743
Birth defects	0.029344
Symptom 4	0.029257
Follow-up	0.029174
Autopsy shows birth defect (if applicable)	0.027809

dtype: float64

```
ranked_features2.nlargest(19).index
```

```
Index(['Blood cell count (mcL)',
       'White Blood cell count (thousand per microliter)', 'Patient Age',
       'Father's age', 'Mother's age', 'No. of previous abortion',
       'Blood test result', 'Birth asphyxia', 'Gender', 'Symptom 5',
       'History of anomalies in previous pregnancies',
       'Respiratory Rate (breaths/min)',
       'Folic acid details (peri-conceptional)', 'Heart Rate (rates/min',
       'Assisted conception IVF/ART', 'Birth defects', 'Symptom 4',
       'Follow-up', 'Autopsy shows birth defect (if applicable)'],
      dtype='object')
```

```
X = X[['White Blood cell count (thousand per microliter)', 'Blood cell count (mcL)', 'Patient Age', "Father's age", "Mother's age",
       'No. of previous abortion', 'Blood test result', 'Gender', 'Birth asphyxia', 'Symptom 5', 'Heart Rate (rates/min',
       'Respiratory Rate (breaths/min)', 'Folic acid details (peri-conceptional)', 'History of anomalies in previous pregnancies',
       'Autopsy shows birth defect (if applicable)', 'Assisted conception IVF/ART', 'Symptom 4', 'Follow-up', 'Birth defects']]
```

```
X.head()
```

	White Blood cell count (thousand per microliter)	Blood cell count (mcL)	Patient Age	Father's age	Mother's age	No. of previous abortion	Blood test result	Gender	Birth asphyxia	Symptom 5	Heart Rate (rates/min	Respiratory Rate (breaths/min)	Folic de (concepti
0	0.652979	-0.511845	-0.714286	0.000000	0.000000	-0.666667	2	0	0	1.0	0	0	
1	-0.541763	0.042123	-0.428571	-1.357143	0.000000	-0.666667	2	0	0	0.0	0	1	
2	0.000000	-0.022004	-0.142857	-1.428571	0.545455	0.666667	2	0	1	1.0	1	0	
3	0.118793	-0.716068	0.714286	0.000000	-1.272727	-0.333333	1	2	1	0.0	0	1	

```
X.shape
```

```
(21805, 19)
```

```
test = test[['White Blood cell count (thousand per microliter)', 'Blood cell count (mcL)', 'Patient Age', "Father's age",
             "Mother's age", 'No. of previous abortion', 'Blood test result', 'Gender', 'Birth asphyxia', 'Symptom 5',
             'Heart Rate (rates/min', 'Respiratory Rate (breaths/min)', 'Folic acid details (peri-conceptional)',
             'History of anomalies in previous pregnancies', 'Autopsy shows birth defect (if applicable)',
             'Assisted conception IVF/ART', 'Symptom 4', 'Follow-up', 'Birth defects']]

test.head()
```

	White Blood cell count (thousand per microliter)	Blood cell count (mcL)	Patient Age	Father's age	Mother's age	No. of previous abortion	Blood test result	Gender	Birth asphyxia	Symptom 5	Heart Rate (rates/min	Respiratory Rate (breaths/min)	Folic det (p conceptio
0	0.000000	0.309384	-0.125	0.826087	0.176471	0.333333	3	2	2	1	0	1	
1	0.242998	0.822755	0.375	0.478261	-0.117647	-0.333333	2	2	2	0	0	0	
2	0.000000	-0.085089	-0.250	0.782609	0.764706	-0.333333	3	0	1	0	0	1	
3	-0.216795	-0.789999	0.750	0.565217	-0.588235	-0.333333	2	0	0	1	0	0	

```
## Combining the Target Variables.
train = pd.concat([X, y1, y2], axis=1)
train.head()
```

	White Blood cell count (thousand per microliter)	Blood cell count (mcL)	Patient Age	Father's age	Mother's age	No. of previous abortion	Blood test result	Gender	Birth asphyxia	Symptom 5	Heart Rate (rates/min	Respiratory Rate (breaths/min)	Folic de (concepti
0	0.652979	-0.511845	-0.714286	0.000000	0.000000	-0.666667	2	0	0	1.0	0	0	
1	-0.541763	0.042123	-0.428571	-1.357143	0.000000	-0.666667	2	0	0	0.0	0	1	
2	0.000000	-0.022004	-0.142857	-1.428571	0.545455	0.666667	2	0	1	1.0	1	0	
3	0.118793	-0.716068	0.714286	0.000000	-1.272727	-0.333333	1	2	1	0.0	0	1	

```
## Re-checking for Missing Values
train.isnull().any()
```

```
0
```