CombinedDataset For Sepsis Prediction

Take this notebook as a continuation of the MAINnotebook

Please keep in mind that i did go through a lot of back and forward between my C# code, DataGrip and This Notebook. This is an Experiment and this notebook shows only vague walkthrough of the main steps i took .(i will be not going in dept on steps already explained in the previous notebooks)

In [ ]:
```
import sklearn
import pandas
import seaborn
import numpy as np

from sklearn.preprocessing import StandardScaler
from sklearn import preprocessing
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix
```

The Combination Process

the dataset i am adding to my existing one:

- contains 40336 patients and contains different test results and Vital signs for each hour after admission, a patient becomes labeled as positive only in the hour he was diagnosed with sepsis , some test results are only taken periodically so it is full of Null values .
- there are a few common columns between the two datasets : Plasma Glucose, Bloud Presure, and Age
  - the Bloud Presure in this dataset is split into 3 difrent values : Systolic, Diastolic, and Mean Arterial Pressure we will be taking the average.
- we will be taking the average values over the whole period of admition for negative patients, and for positive patients we will be taking the average only after the patient was diagnosed with sepsis. This way we insure ourselves we won`t be missing a peek value.

The Dataset I am going to be adding

unfortunatly we have 0 insight on where the data is coming from. Source: https://www.kaggle.com/datasets/salikhussaini49/prediction-of-sepsis

but since we are only going to be using it as aditional data to our existing dataSet it can only add more veriaty, if we were to use it as a standalone dataset it woud be a no go (more detail on what we requre from our data in the Data Reliability section of the Main notebook)

In [ ]:
```
newDataset_df = pandas.read_csv("newSourceSepsisDataset.csv" )
newDataset_df.sample(10)
```

Out[ ]:

	ID	Hour	HR	O2Sat	Temp	SBP	MAP	DBP	Resp	EtCO2	...	Fibrinogen	Platel
681136	18	18	60.0	95.0	NaN	141.0	91.00	65.0	15.0	NaN	...	NaN	N
626567	5	5	105.0	97.0	37.28	153.0	87.67	NaN	16.0	NaN	...	NaN	7
805829	49	49	108.0	98.0	NaN	94.0	67.00	58.0	20.0	NaN	...	NaN	N
939561	2	2	115.0	100.0	NaN	143.0	104.00	94.0	17.0	NaN	...	NaN	N
751573	9	9	71.0	95.0	37.00	100.0	65.33	NaN	18.0	NaN	...	NaN	N
1128714	6	6	96.0	100.0	NaN	111.0	80.00	64.0	NaN	NaN	...	NaN	N
634440	18	18	87.0	98.0	NaN	143.0	75.00	NaN	19.0	NaN	...	NaN	N
1390960	29	29	68.0	96.0	37.00	110.0	72.00	61.0	17.0	NaN	...	NaN	N
78432	15	15	113.0	92.0	38.83	162.0	90.67	NaN	17.0	NaN	...	NaN	N
957585	34	34	84.0	94.0	NaN	123.0	87.00	62.0	18.0	NaN	...	NaN	N

10 rows × 44 columns

How i combined the dataset in .NET

1. imported the 2 datasets in
2. crating aproparietе DTOs for the SourceCSV, AdditionCSV, and the CombinedCSV

- i decided to do the required custom calculations while transforming the type insted of using a model mapper

```
public OutputCsvRowDTO(AdditionCsvRowDTO additionCsvRowDTO)
    {
        Sepssis = Convert.ToInt32(additionCsvRowDTO.SepsisLabel);
        ID = additionCsvRowDTO.ID;
        Age = Convert.ToInt32(additionCsvRowDTO.Age);
        PRG =
(int)ScaleGlucose(Convert.ToInt32(additionCsvRowDTO.Glucose), 9, 999, 0,
9);
        var bpARR = new double?[] {
Convert.ToDouble(additionCsvRowDTO.SBP),
Convert.ToDouble(additionCsvRowDTO.MAP),
Convert.ToDouble(additionCsvRowDTO.DBP) };
```

```
        PR = (int)bpARR.Average();
        //the rest of the fields
```

4. Transforming the data

- calculating avgResultofPostiveSepsisPatientsInLateStage

```
List<AdditionCsvRowDTO> avgResultofPostiveSepsisPatientsInLateStage =
_additionCsvRowDtos
            .Where(x => x.SepsisLabel == "1").GroupBy(x =>
x.Patient_ID).Select(x => new AdditionCsvRowDTO
            {
                 ID = x.First().Patient_ID,
                        HR = x.Average(y => y.HR),
                        O2Sat = x.Average(y => y.O2Sat),
                        //the rest of the fields (for some value ofcourse
we dont get the avarage like Age, Gender ...)
```

- calculating avgResultofNegativeSepsisPatientsInLateStage

since i am going to be taking a multi threaded aproach beacuase iterating over the collection is unavoidable in this case (ultra slow) (we cant use LINQ which only goes over memory refresher addresses to find an object: IndexOF(lambda expresion) and also ofcourse uses SPANS which i am not going to) we need to define a couple of help refrence Colections which are multi thread safe to avoid using a lock statement (locking acess to execution of only one thread)

```
List<AdditionCsvRowDTO> tmpListWithAllRowsWhereSepsisIsNegative =
            _additionCsvRowDtos.Where(x => x.SepsisLabel == "0").ToList();
        ConcurrentBag<string> avgResultOfNegativePatientsPatientIDS =
new();
        ConcurrentBag<AdditionCsvRowDTO> avgResultOfNegativePatients =
new();
        List<string> positiveSepsisPatientIds =
avgResultofPostiveSepsisPatientsInLateStage
            .Select(x => x.Patient_ID)
            .ToList();
Parallel.ForEach(tmpListWithAllRowsWhereSepsisIsNegative.AsParallel().AsOrder
(row, state) =>
        {

            if ( positiveSepsisPatientIds.Contains(row.Patient_ID) ||
avgResultOfNegativePatientsPatientIDS.Contains(row.Patient_ID))
            {
                return;
            }
            else
            {

                avgResultOfNegativePatientsPatientIDS.Add(row.Patient_ID);
```

```
avgResultOfNegativePatients.Add(_additionCsvRowDtos.Where(x =>
x.Patient_ID == row.Patient_ID)
                    .GroupBy(x => x.Patient_ID).Select(x => new
AdditionCsvRowDTO
                    {
                        ID = x.First().Patient_ID,
                        HR = x.Average(y => y.HR),
                        O2Sat = x.Average(y => y.O2Sat),
                        //the rest of the fields
                        //and ofcourse we  .First() to put only one record
in the Bag
```

5. creating the output Object collection

```
outputCsvRowDtos.AddRange( _sourceCsvRowDtos.Select(x => new
OutputCsvRowDTO(x)).ToList());

outputCsvRowDtos.AddRange(avgResultofPostiveSepsisPatientsInLateStage.Select(
=> new OutputCsvRowDTO(x))
            .ToList());
        outputCsvRowDtos.AddRange(avgResultOfNegativePatients.Select(x =>
new OutputCsvRowDTO(x)).ToList());
```

6. writing the output to a csv file

```
File.WriteAllText(outputCSVpath, Output.ToCsv());
```

7. Importing the resulting DataSet in DataGrip to evaluate and check for any mistakes in the prosess (which there were :) and going back to the code to fix it)

8. Balancing the Dataset in SQLITE

```
DELETE FROM "OUTPUT"  WHERE ID IN (
    SELECT ID FROM "OUTPUT"  WHERE Sepssis = 0 ORDER BY RANDOM() LIMIT
6505
)
```

Crating the new Model

1. Taking a Look at the new CombinedDataset
2. Loking for new feature candidates

In [ ]:
```
columns = ["ID", "PRG", "PL", "PR", "SK", "TS", "M11", "BD2", "Age","Insurance", "H
df = pandas.read_csv("balancedCombinedDataset.csv" , names=columns)
df.sample(10)
```

Out[ ]:

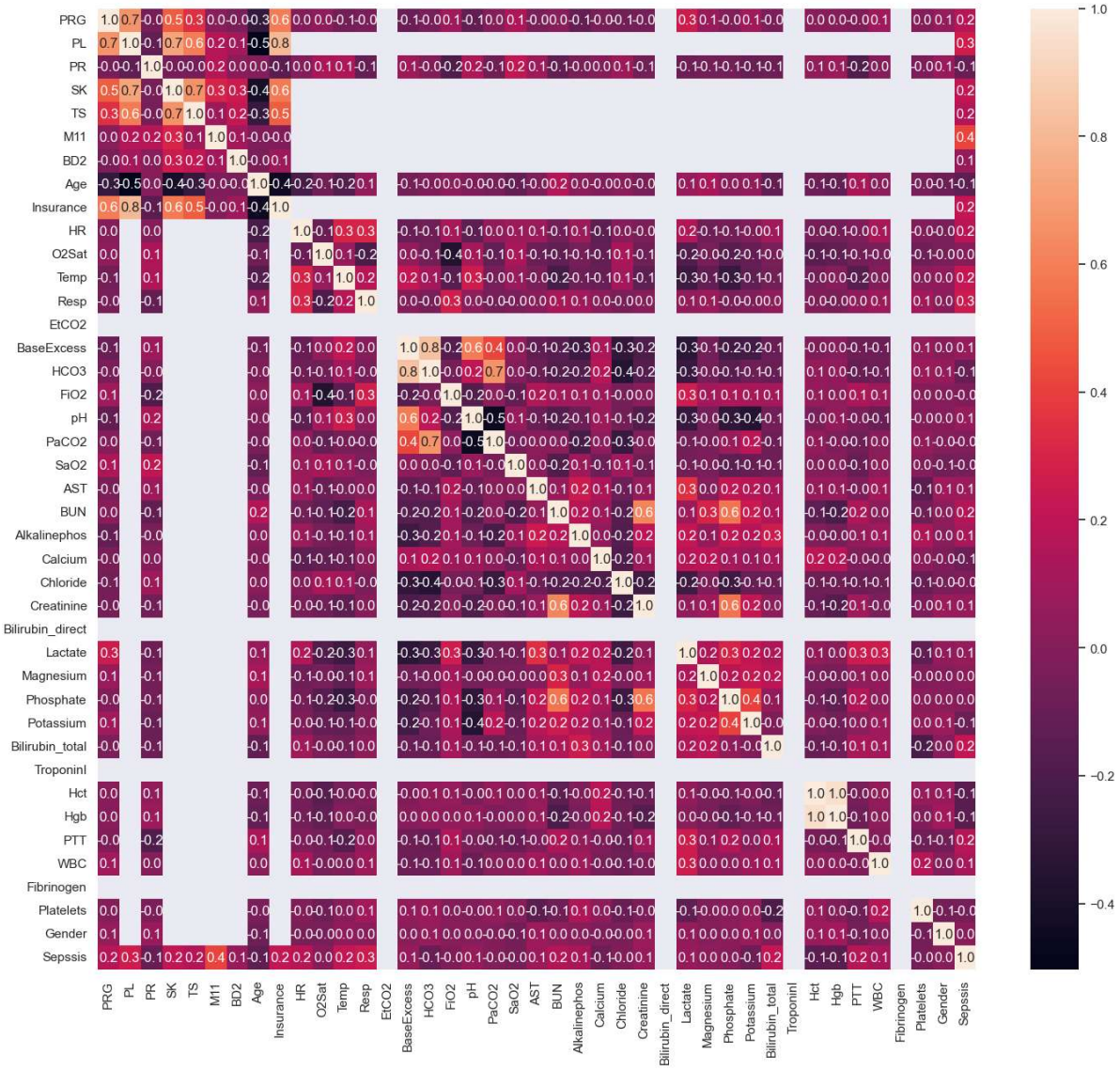
	ID	PRG	PL	PR	SK	TS	M11	BD2	Age	Insurance	...	Bilirubin_total	TroponinI	H
1371	913	1	0	82	0	0	NaN	NaN	70	0	...	NaN	NaN	25
566	11317	0	0	82	0	0	NaN	NaN	53	0	...	NaN	NaN	36
433	8399	1	0	26	0	0	NaN	NaN	69	0	...	NaN	NaN	25
1617	7316	1	0	75	0	0	NaN	NaN	72	0	...	0.40	NaN	34
596	616	0	0	69	0	0	NaN	NaN	88	0	...	NaN	NaN	N
1080	13161	1	0	97	0	0	NaN	NaN	52	0	...	7.98	NaN	28
1696	10683	1	0	78	0	0	NaN	NaN	66	0	...	NaN	NaN	36
1481	10184	1	0	86	0	0	NaN	NaN	69	0	...	NaN	NaN	32
693	13054	0	0	57	0	0	NaN	NaN	88	0	...	NaN	NaN	30
635	4376	1	0	91	0	0	NaN	NaN	77	0	...	1.60	NaN	32

10 rows × 42 columns

In [ ]:
```
group_info = df.groupby(["Sepssis"])["Sepssis"].count()
group_info
```

Out[ ]:
```
Sepssis
0    881
1    881
Name: Sepssis, dtype: int64
```

In [ ]:
```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(17, 15))
corr = df.corr(numeric_only=True)
plot = seaborn.heatmap(corr, cbar=True, annot=True, fmt=".1f", ax=ax)
```



Explenation

after looking at the HeatMap we can safely remove EtCO2 , Bilirubin_direct, TroponinI, Fibrinogen since they don't contain any data

let`s dive deeper into the domain understanding and reasone the high corelation values:

o2Sat vs FiO2 (Fraction of inspired oxygen (%)) - 0.4 this is a corelation between how much oxygen a person inhales and how much it ends up in the blood

hcO3 (Bicarbonate (mmol/L)) vs PaCO2 (Partial pressure of carbon dioxide from arterial blood (mm Hg)) - 0.7 This is because carbon dioxide reacts with water to form bicarbonate ions in the blood, and therefore an increase in PaCO2 will result in an increase in the concentration of bicarbonate ions.
BaseEcess (Measure of excess bicarbonate (mmol/L)) vs hco3 (Bicarbonate (mmol/L)) -0.8 This is expected, as base excess is a measure of the amount of excess bicarbonate in the

blood

Ph (N/A) vs PaCo2 (Partial pressure of carbon dioxide from arterial blood (mm Hg)) - 0.5 The dataset didnt provide information of what the Ph is mesured, that doesnt meen to ignore the unlabled data

Phosphate vs BUN (Blood urea nitrogen (mg/dL)) - 0.6 Phosphate and urea nitrogen are both waste products of metabolism that are excreted by the kidneys

Scaling and Preparation for Feature Selection

In [ ]:
```
from sklearn.preprocessing import StandardScaler, MinMaxScaler

df.drop(columns=["ID"], inplace=True)
from sklearn import preprocessing
encoder = preprocessing.LabelEncoder()
df["Sepssis"] = encoder.fit_transform(df["Sepssis"])

# Separate the target variable from the input features
X = df.iloc[:,:-1]  # input features
y = df.iloc[:, -1]  # target variable (sepsis)

# Scale the input features only\
#scaler = StandardScaler()
scaler = MinMaxScaler(feature_range=(0, 1))
scaler.fit(X)
X_scaled = scaler.transform(X)

# Combine the scaled input features and the target variable
df_scaled = pandas.concat([pandas.DataFrame(X_scaled, columns=X.columns), y], axis=
df_scaled.head()
```

```
C:\Users\ra408\AppData\Roaming\Python\Python39\site-packages\sklearn\preprocessing
\_data.py:473: RuntimeWarning: All-NaN slice encountered
  data_min = np.nanmin(X, axis=0)
C:\Users\ra408\AppData\Roaming\Python\Python39\site-packages\sklearn\preprocessing
\_data.py:474: RuntimeWarning: All-NaN slice encountered
  data_max = np.nanmax(X, axis=0)
```

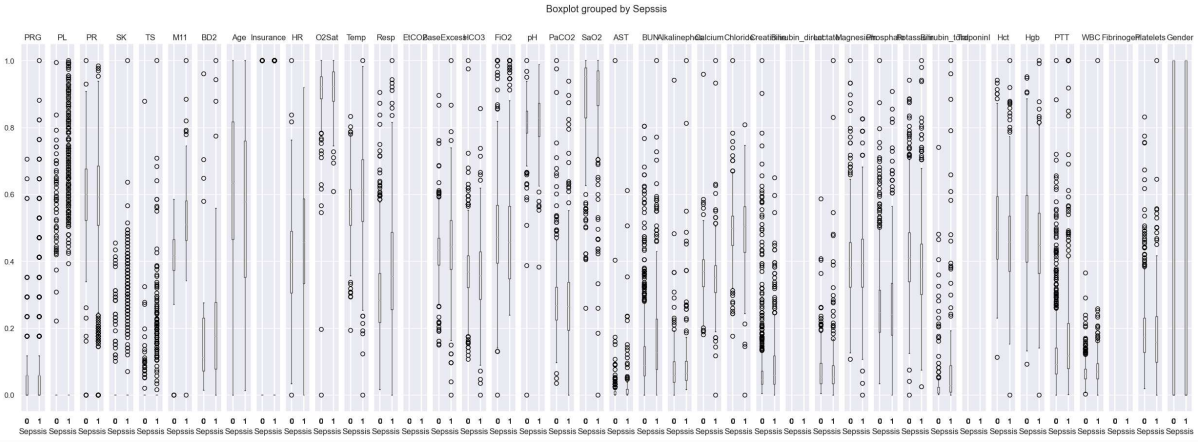
Out[ ]:

	PRG	PL	PR	SK	TS	M11	BD2	Age	Insurance	HR
0	0.352941	0.747475	0.553846	0.353535	0.000000	0.500745	0.231132	0.450704	0.0	NaN
1	0.058824	0.429293	0.507692	0.292929	0.000000	0.396423	0.112779	0.183099	0.0	NaN
2	0.470588	0.924242	0.492308	0.000000	0.000000	0.347243	0.250429	0.197183	1.0	NaN
3	0.000000	0.691919	0.307692	0.353535	0.198582	0.642325	0.943396	0.211268	1.0	NaN
4	0.294118	0.585859	0.569231	0.000000	0.000000	0.381520	0.048456	0.169014	1.0	NaN

5 rows × 41 columns

In [ ]:
```
candidates = columns[1:-1]
fig, ax = plt.subplots(figsize=(30, 10))
plot = df_scaled.boxplot(column=candidates,  by="Sepssis", layout=(1,40), ax=ax)
```

```
C:\Users\ra408\AppData\Local\Temp\ipykernel_17148\3943215225.py:3: UserWarning: To
output multiple subplots, the figure containing the passed axes is being cleared.
  plot = df_scaled.boxplot(column=candidates,  by="Sepssis", layout=(1,40), ax=ax)
```



a lot of circles could represent outliers or extreme values in the dataset. is that the case here?

selecting features , after evaluating the BoxPlots and the Correlation HeatMap

In [ ]:
```
features = ["PL", "Age", "M11", "PR", "TS", "BD2", "HR", "Resp" , "BUN" , "PTT"]
target = "Sepssis"

X = df_scaled[features]
y = df_scaled[target]


from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
print("There are in total", len(X), "observations, of which", len(X_train), "are no
```

```
There are in total 1762 observations, of which 1409 are now in the train set, and
353 in the test set.
```

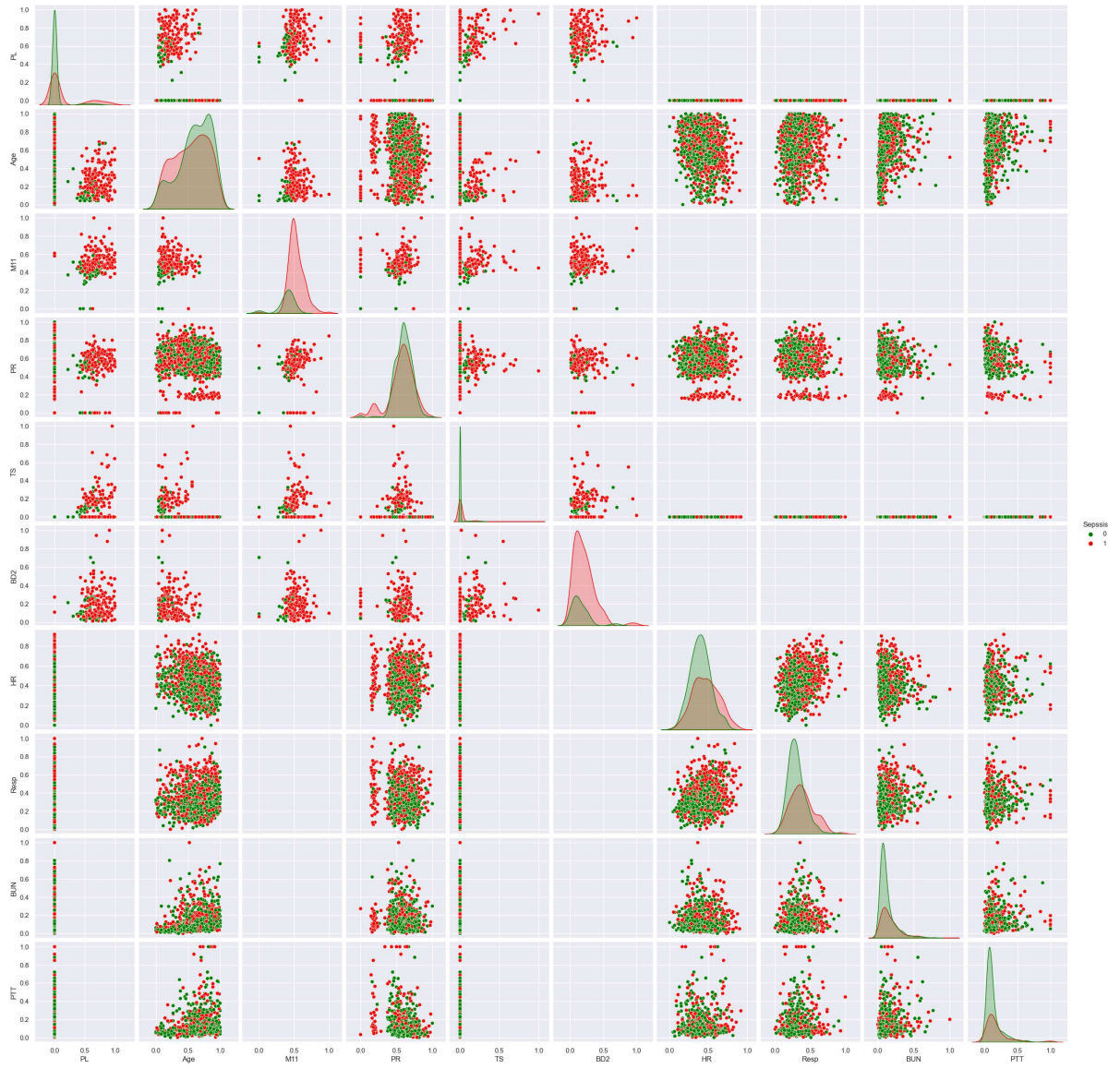
Reasoning

I added HartRate, Respiration rate, Blood urea nitrogen, partial thromboplastin time. I chose these features since they provided the most divivation in the box plots , as well as the hiest corelation to sepsis in the heatMap

taking a look for paterns in our selected features

In [ ]:
```
train_df = pandas.concat([X_train, y_train], axis=1)
seaborn.set_theme()
cmap = {1: "red", 0: "green"}
seaborn.pairplot(train_df, hue=target, palette=cmap )
```

Out[ ]:
```
<seaborn.axisgrid.PairGrid at 0x2c0900d36d0>
```



it is hard to see any patterns or coloration between the features, but we can see that there are some lines forming at 0.0 which is unavoidable since we have a lot of features who never

meet each other on a same row (connection between them cant be made)

pushing the data to a model that supports NaN values

In [ ]:
```
from sklearn.experimental import enable_hist_gradient_boosting  # Required to use H
from sklearn.ensemble import HistGradientBoostingClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.model_selection import cross_val_score
import pandas as pd

# Create and fit HistGradientBoostingClassifier
clf = HistGradientBoostingClassifier()
clf.fit(X_train, y_train)

# Make predictions and calculate accuracy
y_pred = clf.predict(X_test)
acc = accuracy_score(y_test, y_pred)
print("Accuracy:", acc)

# Convert y_test and y_pred to pandas Series to ensure 1D arrays
y_test_series = pd.Series(y_test)
y_pred_series = pd.Series(y_pred)

# Convert class labels to strings before creating target_names
target_names = [str(label) for label in encoder.classes_]
print(classification_report(y_test_series, y_pred_series, target_names=target_names

# Perform cross-validation and print results
cv_scores = cross_val_score(clf, X, y, cv=5)
print("Cross validation results:", cv_scores)
```

```
Accuracy: 0.7535410764872521
              precision    recall  f1-score   support

           0       0.72      0.77      0.74       164
           1       0.79      0.74      0.76       189

    accuracy                           0.75       353
   macro avg       0.75      0.75      0.75       353
weighted avg       0.76      0.75      0.75       353

Cross validation results: [0.80453258 0.7592068  0.72727273 0.75568182 0.69886364]
```

we can see that there wasn't a significant change in the performance of the model

what are the benefits:

we might not always have all the parameters available to us, but we might have difrent once, by broadening the the range of posible inputs of our model(and adding support for NaN values) we will increase the cases that the system can be used

detailed reasoning and explanation at the end of the notebook:

what if we use only the transformedAdditionDataset

In [ ]:

```
dff = pandas.read_csv("balancedCombinedDataset.csv", names=columns)
dff.drop(dff.index[:257], inplace=True)   # droping the old dataset

#dff = dff[dff['Sepssis'].notna()]
dff.reset_index(drop=True, inplace=True)


from sklearn.preprocessing import StandardScaler, MinMaxScaler

dff.drop(columns=["ID"], inplace=True)
from sklearn import preprocessing
encoder = preprocessing.LabelEncoder()
dff["Sepssis"] = encoder.fit_transform(dff["Sepssis"])

# Separate the target variable from the input features
X = dff.iloc[:,:-1]  # input features
y = dff.iloc[:, -1]  # target variable (sepsis)

# Scale the input features only\
#scaler = StandardScaler()
scaler = MinMaxScaler(feature_range=(0, 1))
scaler.fit(X)
X_scaled = scaler.transform(X)

# Combine the scaled input features and the target variable
df_scaled = pandas.concat([pandas.DataFrame(X_scaled, columns=X.columns), y], axis=


features = [ "Age", "PR",  "BD2", "HR", "Resp" , "BUN" , "PTT"]
target = "Sepssis"

X = df_scaled[features]
y = df_scaled[target]


from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
print("There are in total", len(X), "observations, of which", len(X_train), "are no




from sklearn.ensemble import HistGradientBoostingClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.model_selection import cross_val_score
import pandas as pd



# Create and fit HistGradientBoostingClassifier
clf = HistGradientBoostingClassifier()
```

```
clf.fit(X_train, y_train)

# Make predictions and calculate accuracy
y_pred = clf.predict(X_test)
acc = accuracy_score(y_test, y_pred)
print("Accuracy:", acc)

# Convert y_test and y_pred to pandas Series to ensure 1D arrays
y_test_series = pd.Series(y_test)
y_pred_series = pd.Series(y_pred)

# Convert class labels to strings before creating target_names
target_names = [str(label) for label in encoder.classes_]
print(classification_report(y_test_series, y_pred_series, target_names=target_names

# Perform cross-validation and print results
cv_scores = cross_val_score(clf, X, y, cv=5)
print("Cross validation results:", cv_scores)
```

```
There are in total 1505 observations, of which 1204 are now in the train set, and
301 in the test set.
C:\Users\ra408\AppData\Roaming\Python\Python39\site-packages\sklearn\preprocessing
\_data.py:473: RuntimeWarning: All-NaN slice encountered
  data_min = np.nanmin(X, axis=0)
C:\Users\ra408\AppData\Roaming\Python\Python39\site-packages\sklearn\preprocessing
\_data.py:474: RuntimeWarning: All-NaN slice encountered
  data_max = np.nanmax(X, axis=0)
Accuracy: 0.7375415282392026
              precision    recall  f1-score   support

           0       0.72      0.83      0.77       159
           1       0.77      0.63      0.69       142

    accuracy                           0.74       301
   macro avg       0.74      0.73      0.73       301
weighted avg       0.74      0.74      0.73       301

Cross validation results: [0.75083056 0.73089701 0.7641196  0.71428571 0.73421927]
```

evaluation of the original dataset for refrence (first dataset)

```
Accuracy:
              precision    recall  f1-score   support

    Negative       0.78      0.78      0.78        45
    Positive       0.74      0.74      0.74        38

    accuracy                           0.76        83
   macro avg       0.76      0.76      0.76        83
weighted avg       0.76      0.76      0.76        83

Cross validation results [0.74698795 0.71084337 0.69879518
0.75609756 0.76829268]
```

Conclusion

Based on the evaluation results, it can be concluded that the performance of the combined dataset is relatively similar to the individual datasets in terms of accuracy, precision, recall, and F1-score.

In the comparison, the second dataset's evaluation had an overall accuracy of 72.4%, with a macro average and weighted average F1-score of 0.71. The cross-validation results for this dataset ranged from 71.4% to 76.4%. The first dataset exhibited an overall accuracy of 75.9%, with macro average and weighted average F1-scores of 0.76. The cross-validation results for the first dataset ranged from 69.9% to 76.8%.

When comparing the datasets, the combined dataset shows a slightly higher overall accuracy compared to the individual datasets. Additionally, the macro average and weighted average F1-scores for the combined dataset are similar to those of the first dataset and higher than those of the second dataset. The cross-validation results for the combined dataset show a wider range of accuracy scores compared to the individual datasets, suggesting that the combined dataset might exhibit higher variance in performance.

However,The lower recall for Positive cases in the second dataset suggests that the model has difficulty identifying true instances of positive cases in this specific dataset. in our cases this is cosed by the differences in the distribution of features if we look up at the box plots we can see that even the best candidates that I have chosen for features have much worse separation between the two classes in the second dataset in comparison to the first one

based on this reasoning I will choose the combined dataset as it builds upon the first dataset and adds more features that gives medical professionals more flexibility in the use of the model