cdac-finalcode

February 19, 2024

1 Prediction Of Preoperative Risk

[1]:
```
#importing all necessary libraries

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import re

# machine learning
from sklearn.model_selection import␣
↪(GridSearchCV,StratifiedKFold,train_test_split)
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import␣
↪(log_loss,confusion_matrix,classification_report,ConfusionMatrixDisplay,roc_curve,accuracy_
from sklearn.model_selection import GridSearchCV,StratifiedKFold
from sklearn.model_selection import KFold,cross_val_score
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import StackingClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
import plotly.offline as pyo
from sklearn.svm import SVC
import warnings
warnings.filterwarnings("ignore")
```

2 Postoperative mortality risk based on preoperative data

Accurate knowledge of individual patient risk is essential to raise awareness for early recognition of postoperative complications and adequate planning of intraoperative management and postoperative care. Furthermore, from an ethical and legal point of view, the patient has the right to know his or her risk of the planned procedure to enable shared decision making with physician and patient. Objective is to make the model comprehensible for the physician.

2.0.1 Loading the dataset

[2]:
```
#reading the data
df=pd.read_csv("clinical_data.csv")
df.head()

#Using list(df) to get the list of all Column Names
column_headers = list(df)
print(column_headers)
```

```
['case_id', 'subjectid', 'casestart', 'caseend', 'anestart', 'aneend',
'opstart', 'opend', 'adm', 'dis', 'icu_days', 'death_inhosp', 'age', 'sex',
'height', 'weight', 'bmi', 'asa', 'emop', 'department', 'optype', 'dx',
'opname', 'approach', 'position', 'ane_type', 'preop_htn', 'preop_dm',
'preop_ecg', 'preop_pft', 'preop_hb', 'preop_plt', 'preop_pt', 'preop_aptt',
'preop_na', 'preop_k', 'preop_gluc', 'preop_alb', 'preop_ast', 'preop_alt',
'preop_bun', 'preop_cr', 'preop_ph', 'preop_hco3', 'preop_be', 'preop_pao2',
'preop_paco2', 'preop_sao2', 'cormack', 'airway', 'tubesize', 'dltubesize',
'lmasize', 'iv1', 'iv2', 'aline1', 'aline2', 'cline1', 'cline2', 'intraop_ebl',
'intraop_uo', 'intraop_rbc', 'intraop_ffp', 'intraop_crystalloid',
'intraop_colloid', 'intraop_ppf', 'intraop_mdz', 'intraop_ftn', 'intraop_rocu',
'intraop_vecu', 'intraop_eph', 'intraop_phe', 'intraop_epi', 'intraop_ca']
```

[3]:
```
# Rename a single column permanently
df.rename(columns={'death_inhosp': 'outcome'}, inplace=True)
```

[47]:
```
#total number of records
print(df.shape) #6388
```

```
(6388, 74)
```

[70]:
```
#As our aim is to assess risk of postoperative mortality based on preoperative␣
↪data
#selecting only relevant columns
preop_df=df[['emop','dx','outcome','adm','icu_days', 'age', 'bmi',␣
↪'asa','sex','department','optype','ane_type'
       ,'preop_htn', 'preop_dm', 'preop_ecg', 'preop_pft', 'preop_hb',
       'preop_plt', 'preop_pt', 'preop_aptt', 'preop_na', 'preop_k',
       'preop_gluc', 'preop_alb', 'preop_ast', 'preop_alt', 'preop_bun',
       'preop_cr']]
```

2.0.2 Excluding underage and emergency patients

[71]:
```
print(preop_df.info())

pd.set_option('display.max_columns', None)# as age column is not displaying
print(preop_df['age'])
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6388 entries, 0 to 6387
Data columns (total 28 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   emop        6388 non-null   int64
 1   dx          6388 non-null   object
 2   outcome     6388 non-null   int64
 3   adm         6388 non-null   int64
 4   icu_days    6388 non-null   int64
 5   age         6388 non-null   object
 6   bmi         6388 non-null   float64
 7   asa         6255 non-null   float64
 8   sex         6388 non-null   object
 9   department  6388 non-null   object
 10  optype      6388 non-null   object
 11  ane_type    6388 non-null   object
 12  preop_htn   6388 non-null   int64
 13  preop_dm    6388 non-null   int64
 14  preop_ecg   6388 non-null   object
 15  preop_pft   6388 non-null   object
 16  preop_hb    6047 non-null   float64
 17  preop_plt   6047 non-null   float64
 18  preop_pt    5998 non-null   float64
 19  preop_aptt  5986 non-null   float64
 20  preop_na    5765 non-null   float64
 21  preop_k     5767 non-null   float64
 22  preop_gluc  6010 non-null   float64
 23  preop_alb   6016 non-null   float64
 24  preop_ast   6022 non-null   float64
 25  preop_alt   6024 non-null   float64
 26  preop_bun   6023 non-null   float64
 27  preop_cr    6016 non-null   float64
dtypes: float64(14), int64(6), object(8)
memory usage: 1.4+ MB
None
0       77
1       54
2       62
3       74
4       66
```

```
        ..
6383    64
6384    69
6385    61
6386    24
6387    47
Name: age, Length: 6388, dtype: object
```

[72]:
```
#as age datatype: object ..we have to convert it into integer for removing␣
↪records of underage

# Identify decimal values in the 'age' column
decimal_values = preop_df[preop_df['age'].str.contains('\.')]

# Check if decimal_values is empty
if decimal_values.empty:
    print("No decimal values found in the 'age' column.")
else:
    print("Found decimal values in the 'age' column:")
    print(decimal_values)

preop_df = preop_df[~preop_df['age'].str.contains('\.')]
# Remove rows with '>89' values in the 'age' column
preop_df = preop_df[~preop_df['age'].str.contains('>')]

print(preop_df['age'])


preop_df['age'] = preop_df['age'].astype('int')
#checking underage

# Count the number of records of underage patients
count_underage = preop_df[preop_df['age'] <18].shape[0]
print("Number of records of underage ", count_underage)

count_valid = preop_df[preop_df['age'] >=18].shape[0]
print("Number of records of patients aged 18 and above ", count_valid)
```

```
Found decimal values in the 'age' column:
      emop                           dx  outcome      adm  icu_days  age  \
263      1        Primary hyperoxaluria        0 -1534620        38  0.7
279      0              Biliary atresia        0  -294240        15  0.6
365      0                 Hyperoxaluria        0 -4091460        38  0.8
2319     1                Hepatoblastoma        0  -844560         6  0.3
3229     1        Primary hyperoxaluria        1 -2212920        38  0.7
3485     1        Primary hyperoxaluria        1 -3281580        38  0.7
4646     1       Liver transplant status        1 -1450080        17  0.4
4877     1              Biliary atresia        0  -874920        15  0.7
```

5502	1	Hyperoxaluria	1	-2056500	38	0.7
6336	1	Hepatic failure without coma	1	-1286520	17	0.4

	bmi	asa	sex	department	optype	ane_type	preop_htn \
263	21.7	4.0	F	General surgery	Transplantation	General	0
279	31.7	3.0	F	General surgery	Transplantation	General	0
365	21.7	4.0	F	General surgery	Others	General	0
2319	14.7	3.0	M	General surgery	Transplantation	General	0
3229	21.7	4.0	F	General surgery	Others	General	0
3485	21.7	4.0	F	General surgery	Transplantation	General	0
4646	16.5	3.0	F	General surgery	Vascular	General	0
4877	31.7	3.0	F	General surgery	Others	General	0
5502	21.7	4.0	F	General surgery	Others	General	0
6336	16.5	3.0	F	General surgery	Transplantation	General	0

	preop_dm	preop_ecg	preop_pft	preop_hb	preop_plt	preop_pt \
263	0	Normal Sinus Rhythm	Normal	6.9	19.0	21.0
279	0	Normal Sinus Rhythm	Normal	8.5	295.0	58.0
365	0	Normal Sinus Rhythm	Normal	6.9	19.0	21.0
2319	0	Normal Sinus Rhythm	Normal	9.3	349.0	103.0
3229	0	Normal Sinus Rhythm	Normal	6.9	19.0	21.0
3485	0	Normal Sinus Rhythm	Normal	6.9	19.0	21.0
4646	0	Normal Sinus Rhythm	Normal	8.5	105.0	24.0
4877	0	Normal Sinus Rhythm	Normal	8.5	295.0	58.0
5502	0	Normal Sinus Rhythm	Normal	6.9	19.0	21.0
6336	0	Normal Sinus Rhythm	Normal	8.5	105.0	24.0

	preop_aptt	preop_na	preop_k	preop_gluc	preop_alb	preop_ast \
263	63.9	132.0	2.9	111.0	2.6	1098.0
279	34.9	138.0	4.5	67.0	3.0	33.0
365	63.9	132.0	2.9	111.0	2.6	1098.0
2319	28.1	137.0	4.8	102.0	2.8	24.0
3229	63.9	132.0	2.9	111.0	2.6	1098.0
3485	63.9	132.0	2.9	111.0	2.6	1098.0
4646	104.7	139.0	3.9	79.0	2.7	673.0
4877	34.9	138.0	4.5	67.0	3.0	33.0
5502	63.9	132.0	2.9	64.0	2.6	1098.0
6336	104.7	139.0	3.9	79.0	2.7	673.0

	preop_alt	preop_bun	preop_cr
263	771.0	34.0	0.27
279	50.0	24.0	0.10
365	771.0	34.0	0.27
2319	8.0	4.0	0.15
3229	771.0	8.0	0.27
3485	771.0	34.0	0.27
4646	94.0	9.0	0.22
4877	50.0	24.0	0.10

```
5502      771.0       34.0      0.27
6336       94.0        9.0      0.22
0       77
1       54
2       62
3       74
4       66
        ..
6383    64
6384    69
6385    61
6386    24
6387    47
Name: age, Length: 6370, dtype: object
Number of records of underage  47
Number of records of patients aged 18 and above  6323
```

[73]:
```python
#removing underage
preop_df=preop_df.loc[(preop_df['age']>=18)]
print(preop_df.shape)
```

```
(6323, 28)
```

[74]:
```python
#removing  emergency patients
preop_df = preop_df.loc[(preop_df['emop']==0)]
print(preop_df.shape)
```

```
(5567, 28)
```

[75]:
```python
# Count the number of records of emergency patients
count_emergency = preop_df[preop_df['emop'] == 1].shape[0]
print("Number of records ", count_emergency)

# Count the number of records of underage patients
count_underage = preop_df[preop_df['age'] <18].shape[0]
print("Number of records ", count_underage)
```

```
Number of records  0
Number of records  0
```

2.0.3 Excluding icu patients and ASA with greater than 5

[76]:
```python
#checking how many patients are icu patients
count_icu = preop_df[preop_df['icu_days'] > 0].shape[0]
print("Number of records of icu patients", count_icu)

# Count the number of records of patients with ASA >5
count_ASA = preop_df[preop_df['asa'] >5].shape[0]
```

```python
print("Number of records of asa >5", count_ASA)
```

```
Number of records of icu patients 927
Number of records of asa >5 11
```

[77]:
```python
final_preop_cases = preop_df.loc[(preop_df['icu_days']==0)& ␣
↪(preop_df['asa']<5)]
print(final_preop_cases.shape)
```

```
(4565, 28)
```

[78]:
```python
#after removing,
#checking how many patients are icu patients
count_icu = final_preop_cases[final_preop_cases['icu_days'] > 0].shape[0]
print("Number of records of icu patients", count_icu)

# Count the number of records of patients with ASA >5
count_ASA = final_preop_cases[final_preop_cases['asa'] >5].shape[0]
print("Number of records of asa >5", count_ASA)
```

```
Number of records of icu patients 0
Number of records of asa >5 0
```

[79]:
```python
final_preop_cases.describe()
```

[79]:
```
          emop      outcome           adm  icu_days          age          bmi  \
count  4565.0  4565.000000  4.565000e+03    4565.0  4565.000000  4565.000000
mean      0.0     0.003943 -2.305566e+05       0.0    56.658708    23.534414
std       0.0     0.062677  2.914652e+05       0.0    14.060439     3.515148
min       0.0     0.000000 -4.969620e+06       0.0    18.000000    12.900000
25%       0.0     0.000000 -2.212200e+05       0.0    47.000000    21.100000
50%       0.0     0.000000 -2.008800e+05       0.0    58.000000    23.300000
75%       0.0     0.000000 -1.287000e+05       0.0    67.000000    25.600000
max       0.0     1.000000 -3.552000e+04       0.0    89.000000    43.200000

               asa    preop_htn     preop_dm     preop_hb    preop_plt  \
count  4565.000000  4565.000000  4565.000000  4431.000000  4429.000000
mean      1.727930     0.292004     0.092881    13.088919   246.273651
std       0.578631     0.454734     0.290297     1.834903    78.843146
min       1.000000     0.000000     0.000000     6.100000     5.000000
25%       1.000000     0.000000     0.000000    12.000000   197.000000
50%       2.000000     0.000000     0.000000    13.200000   238.000000
75%       2.000000     1.000000     0.000000    14.300000   285.000000
max       4.000000     1.000000     1.000000    19.300000  1156.000000

          preop_pt   preop_aptt     preop_na      preop_k   preop_gluc  \
count  4392.000000  4382.000000  4232.000000  4233.000000  4405.000000
```

```
mean    102.247040   32.481378  140.344282    4.202599  111.986379
std      12.630309    3.928839    2.454761    0.376386   36.207688
min      26.000000   19.200000  119.000000    2.900000   44.000000
25%      95.000000   30.100000  139.000000    4.000000   94.000000
50%     103.000000   32.100000  141.000000    4.200000  102.000000
75%     110.000000   34.300000  142.000000    4.400000  117.000000
max     159.000000  101.300000  148.000000    6.300000  525.000000

         preop_alb    preop_ast    preop_alt    preop_bun     preop_cr
count  4419.000000  4423.000000  4422.000000  4419.000000  4420.000000
mean      4.156461    23.772779    23.364767    14.978276     0.967373
std       0.429145    17.734758    25.371266     8.482345     1.229971
min       0.800000     2.000000     1.000000     3.000000     0.280000
25%       3.900000    17.000000    13.000000    11.000000     0.660000
50%       4.200000    20.000000    18.000000    14.000000     0.770000
75%       4.400000    25.000000    26.000000    17.000000     0.930000
max       5.300000   528.000000   767.000000   127.000000    20.730000
```

[80]:

```python
#dropping unnecessary columns
final_preop_cases=final_preop_cases.drop(['emop', 'icu_days','dx','adm'],␣
↪axis=1)
```

[81]:

```python
pip install missingno
```

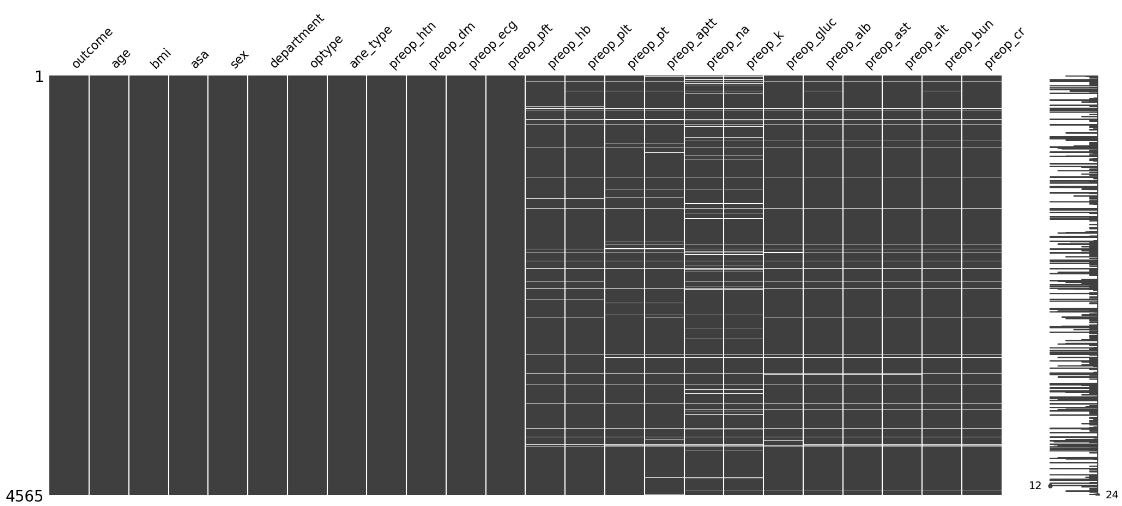
```
Requirement already satisfied: missingno in c:\users\ghild\anaconda3\lib\site-
packages (0.5.2)
Requirement already satisfied: seaborn in c:\users\ghild\anaconda3\lib\site-
packages (from missingno) (0.11.2)
Requirement already satisfied: matplotlib in c:\users\ghild\anaconda3\lib\site-
packages (from missingno) (3.5.1)
Requirement already satisfied: scipy in c:\users\ghild\anaconda3\lib\site-
packages (from missingno) (1.7.3)
Note: you may need to restart the kernel to use updated packages.
Requirement already satisfied: numpy in c:\users\ghild\anaconda3\lib\site-
packages (from missingno) (1.22.4)
Requirement already satisfied: fonttools>=4.22.0 in
c:\users\ghild\anaconda3\lib\site-packages (from matplotlib->missingno) (4.25.0)
Requirement already satisfied: python-dateutil>=2.7 in
c:\users\ghild\anaconda3\lib\site-packages (from matplotlib->missingno) (2.8.2)
Requirement already satisfied: cycler>=0.10 in
c:\users\ghild\anaconda3\lib\site-packages (from matplotlib->missingno) (0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
c:\users\ghild\anaconda3\lib\site-packages (from matplotlib->missingno) (1.3.2)
Requirement already satisfied: packaging>=20.0 in
c:\users\ghild\anaconda3\lib\site-packages (from matplotlib->missingno) (21.3)
Requirement already satisfied: pyparsing>=2.2.1 in
c:\users\ghild\anaconda3\lib\site-packages (from matplotlib->missingno) (3.0.4)
```

```
Requirement already satisfied: pillow>=6.2.0 in
c:\users\ghild\anaconda3\lib\site-packages (from matplotlib->missingno) (9.0.1)
Requirement already satisfied: six>=1.5 in c:\users\ghild\anaconda3\lib\site-
packages (from python-dateutil>=2.7->matplotlib->missingno) (1.16.0)
Requirement already satisfied: pandas>=0.23 in
c:\users\ghild\anaconda3\lib\site-packages (from seaborn->missingno) (1.4.2)
Requirement already satisfied: pytz>=2020.1 in
c:\users\ghild\anaconda3\lib\site-packages (from
pandas>=0.23->seaborn->missingno) (2021.3)
```

2.1 Checking missing data

[82]:
```python
#it has been noticed that data has been missing from any column in the data.

import missingno

missingno.matrix(final_preop_cases)
```

[82]: <AxesSubplot:>



[83]:
```python
print(final_preop_cases.shape)
final_preop_cases.isnull().sum()
```

```
(4565, 24)
```

[83]:
```
outcome         0
age             0
bmi             0
asa             0
sex             0
```

```
department      0
optype          0
ane_type        0
preop_htn       0
preop_dm        0
preop_ecg       0
preop_pft       0
preop_hb      134
preop_plt     136
preop_pt      173
preop_aptt    183
preop_na      333
preop_k       332
preop_gluc    160
preop_alb     146
preop_ast     142
preop_alt     143
preop_bun     146
preop_cr      145
dtype: int64
```

[165]:
```
check_outliers=['preop_hb',
       'preop_plt', 'preop_pt', 'preop_aptt', 'preop_na', 'preop_k',
       'preop_gluc', 'preop_alb', 'preop_ast', 'preop_alt',␣
 ↪'preop_bun','preop_cr']
outliers=[]
def detect_outliers(data):
    threshold=3
    mean=np.mean(data)
    std=np.std(data)

    for i in data:
        z_score=(i-mean)/std
        if np.abs(z_score)>threshold:
            outliers.append(i)
    return outliers
```

[166]:
```
for column_name in check_outliers:
    column_data = final_preop_cases[column_name]
    column_outliers = detect_outliers(column_data)
    outlier_count = len(column_outliers)
    print(f"Number of outliers in column '{column_name}': {outlier_count}")
```

```
Number of outliers in column 'preop_hb': 17
Number of outliers in column 'preop_plt': 71
Number of outliers in column 'preop_pt': 128
Number of outliers in column 'preop_aptt': 169
```

```
Number of outliers in column 'preop_na': 218
Number of outliers in column 'preop_k': 269
Number of outliers in column 'preop_gluc': 368
Number of outliers in column 'preop_alb': 429
Number of outliers in column 'preop_ast': 483
Number of outliers in column 'preop_alt': 519
Number of outliers in column 'preop_bun': 591
Number of outliers in column 'preop_cr': 677
```

[167]:
```
columns_to_impute_mean= ['preop_hb','preop_plt',␣
↪'preop_pt','preop_aptt','preop_na','preop_k','preop_gluc']
final_preop_cases[columns_to_impute_mean] =␣
↪final_preop_cases[columns_to_impute_mean].
↪fillna(final_preop_cases[columns_to_impute_mean].mean())
final_preop_cases[columns_to_impute_mean]=final_preop_cases[columns_to_impute_mean].
↪round(2)
```

[168]:
```
columns_to_impute_median = [
       'preop_ast','preop_bun','preop_alb','preop_cr']

final_preop_cases[columns_to_impute_median] =␣
↪final_preop_cases[columns_to_impute_median].
↪fillna(final_preop_cases[columns_to_impute_median].median().round(2))
final_preop_cases[columns_to_impute_median]=final_preop_cases[columns_to_impute_median].
↪round(2)
```

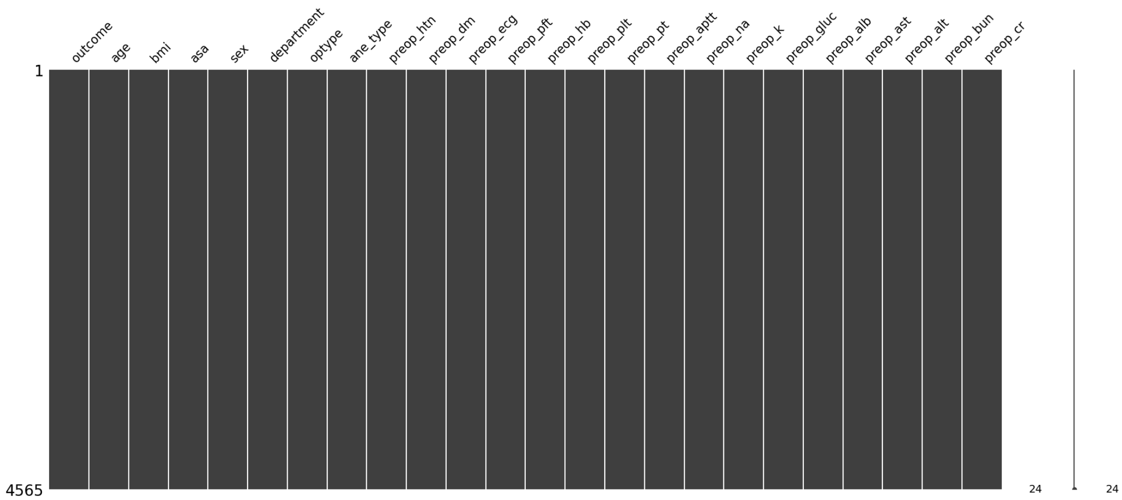
[86]:
```
final_preop_cases.isnull().sum()
```

[86]:
```
outcome       0
age           0
bmi           0
asa           0
sex           0
department    0
optype        0
ane_type      0
preop_htn     0
preop_dm      0
preop_ecg     0
preop_pft     0
preop_hb      0
preop_plt     0
preop_pt      0
preop_aptt    0
preop_na      0
preop_k       0
preop_gluc    0
```

```
preop_alb      0
preop_ast      0
preop_alt      0
preop_bun      0
preop_cr       0
dtype: int64
```

[87]:
```python
missingno.matrix(final_preop_cases)
```

[87]:
```
<AxesSubplot:>
```

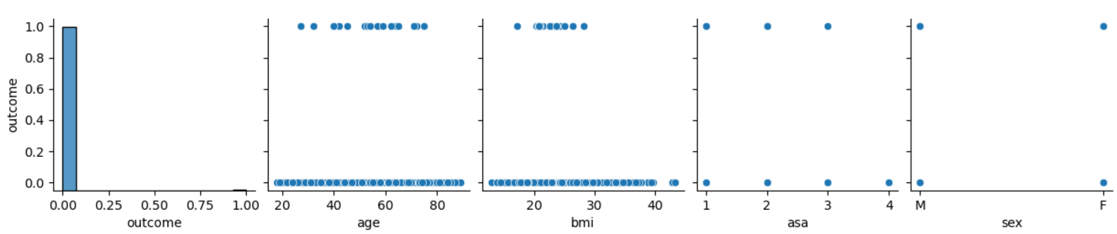


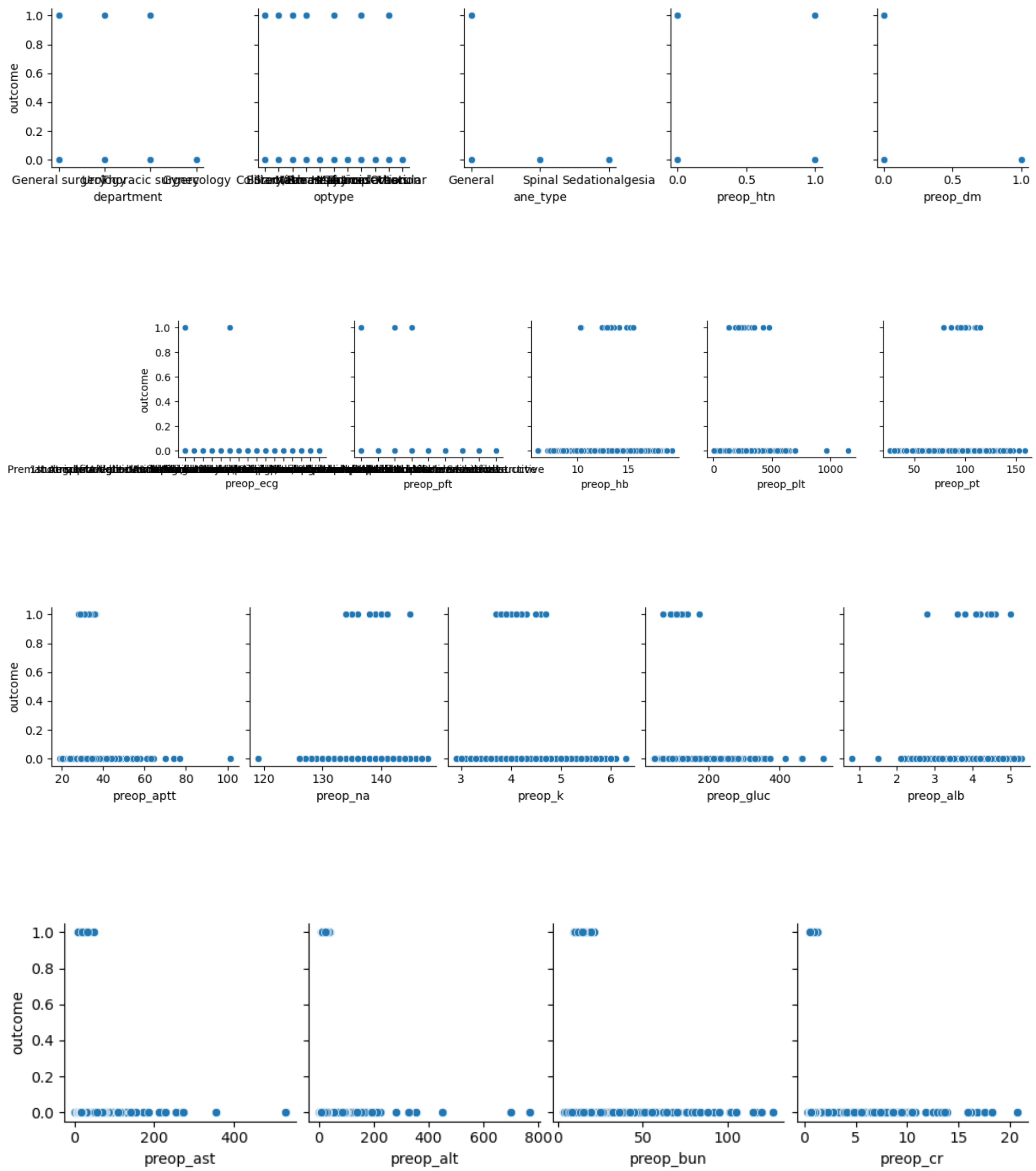
3 Bivariate Analysis

Scatter plots of features(x axis) with target variable(y axis) were plotted out to understand the relationship between the features and target variable.

From the plots, no linear relationship could be found out between the target variable and features

[88]:
```python
for i in range(0, len(final_preop_cases.columns), 5):
    sns.pairplot(data=final_preop_cases,
                 x_vars=final_preop_cases.columns[i:i+5],
                 y_vars=['outcome'])
```



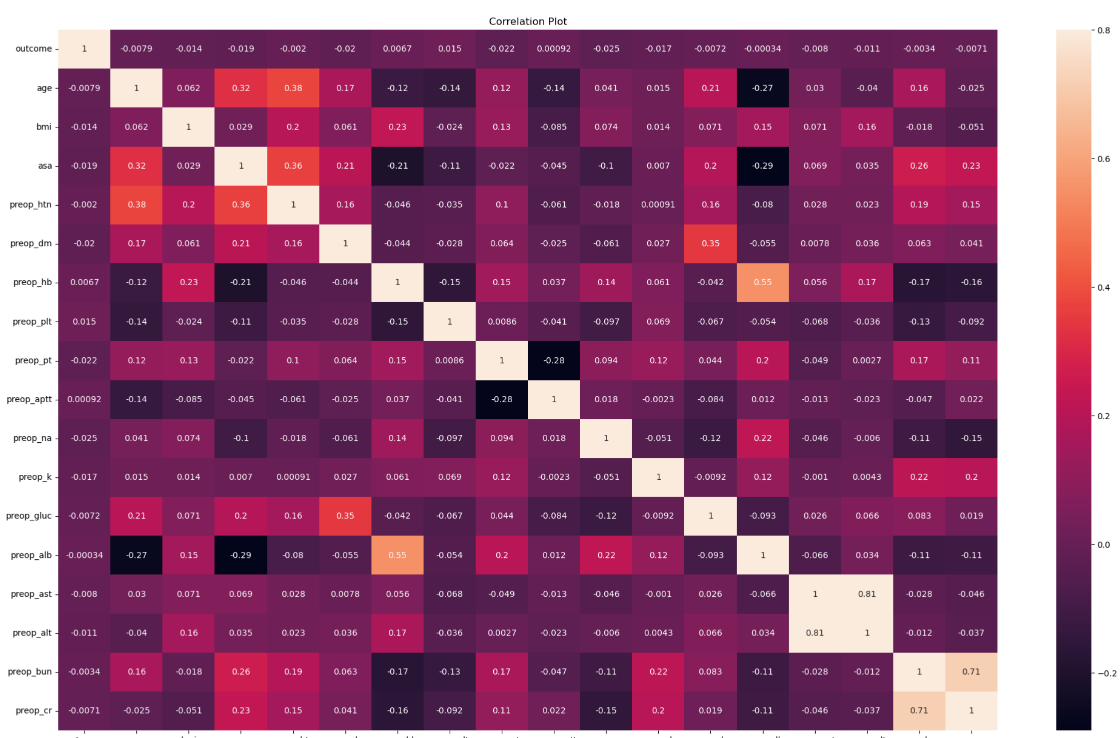


3.0.1 Correlation Matrix

As no linear relationship was found between the target varible and features, Pearsons's correlation was calculated for the data and following insights were found out - 1. Positive Correlations: - Preoperative Platelet Count (preop_plt) and Preoperative Hemoglobin (preop_hb) show a moderate positive correlation. - Preoperative Creatinine (preop_cr) and Blood Urea Nitrogen (preop_bun) exhibit a strong positive correlation. 2. Weak Correlations: - Most of the other selected features show weak correlations with each other, indicating a lack of strong linear relationships.

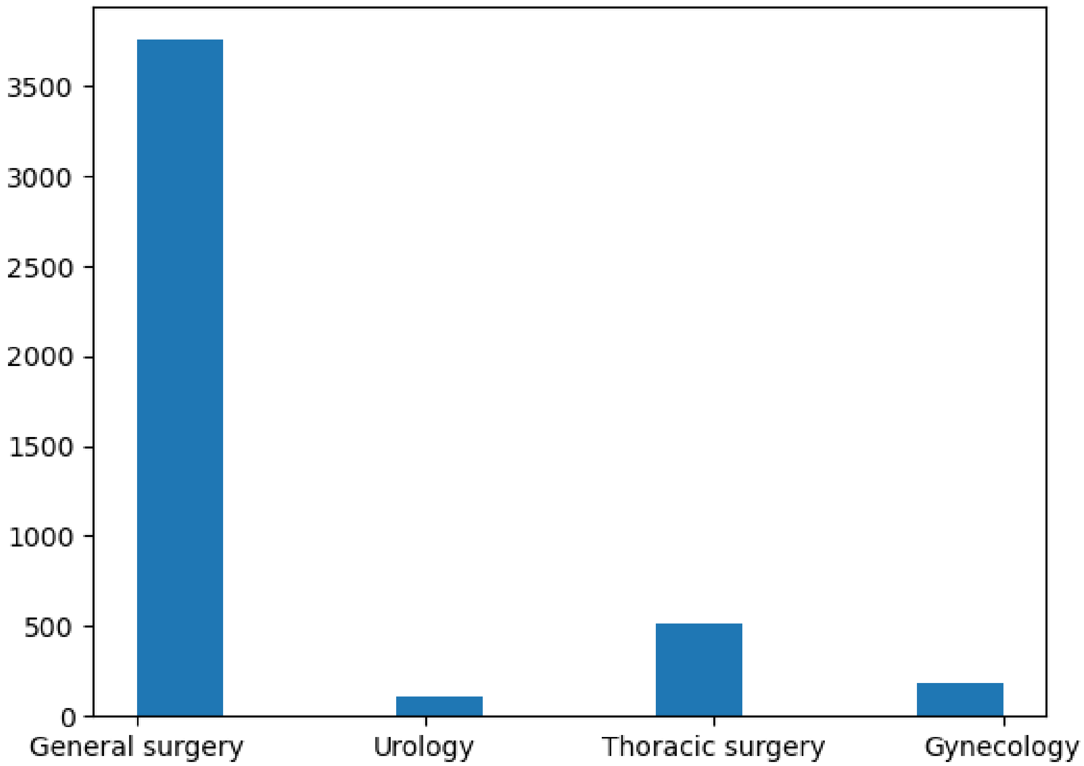
[91]:
```python
import seaborn as sns
import matplotlib.pyplot as plt

final_preop_cases_numeric = final_preop_cases.select_dtypes(include=['number'])
correlation_matrix = final_preop_cases_numeric.corr()


# Assuming 'df' is your DataFrame containing the variables for which you want␣
↪to create the correlation plot
# You can replace 'df' with your actual DataFrame and select the specific␣
↪columns you want to include in the correlation plot



# Create a heatmap to visualize the correlation matrix
plt.figure(figsize=(10, 8))
f, ax = plt.subplots(figsize=(25, 15))
sns.heatmap(correlation_matrix, vmax=.8, annot=True)
plt.title('Correlation Plot')
plt.show()
```

<Figure size 1000x800 with 0 Axes>



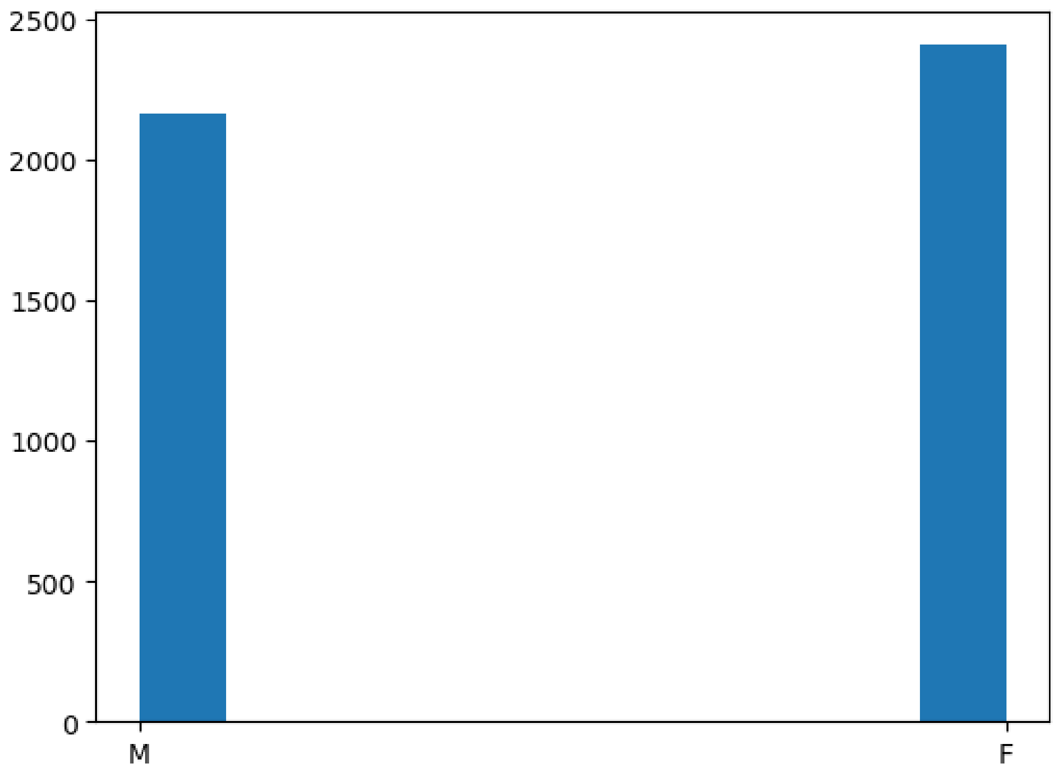
[92]:
```
plt.hist(final_preop_cases['department'])
```

[92]:
```
(array([3752.,    0.,    0.,  112.,    0.,    0.,  515.,    0.,    0.,
         186.]),
 array([0. , 0.3, 0.6, 0.9, 1.2, 1.5, 1.8, 2.1, 2.4, 2.7, 3. ]),
 <BarContainer object of 10 artists>)
```



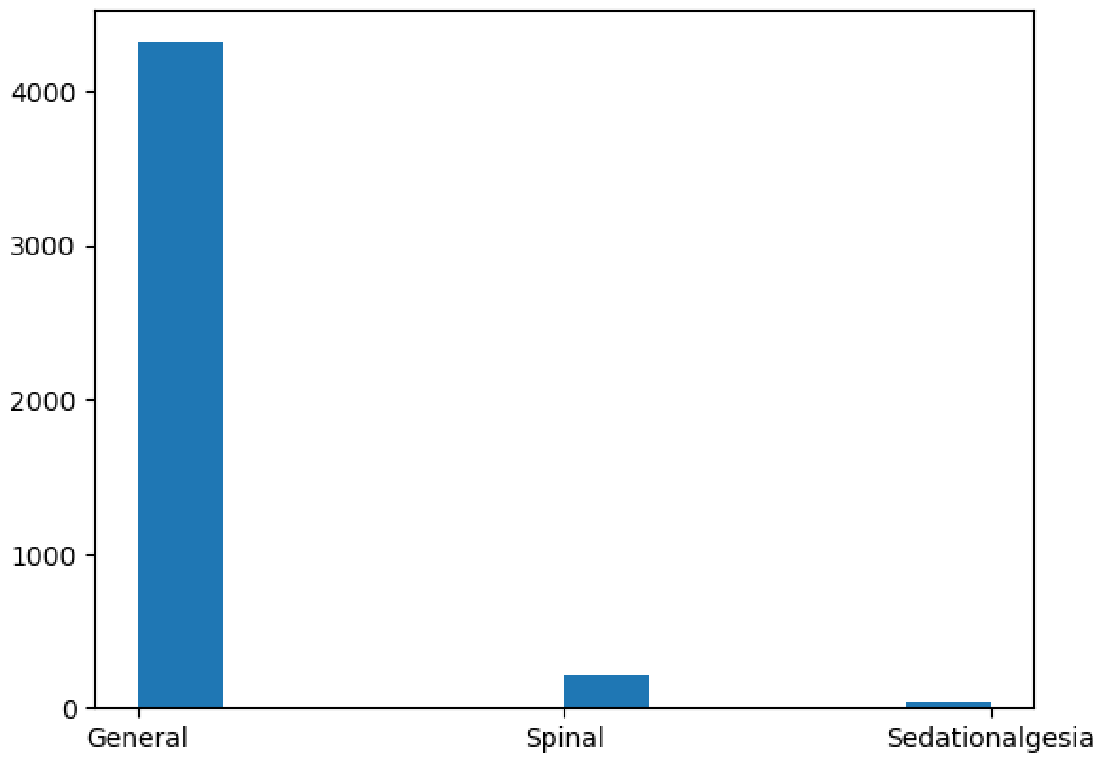
[93]:
```
plt.hist(final_preop_cases['sex'])
```

[93]:
```
(array([2159.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,
        2406.]),
 array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),
 <BarContainer object of 10 artists>)
```



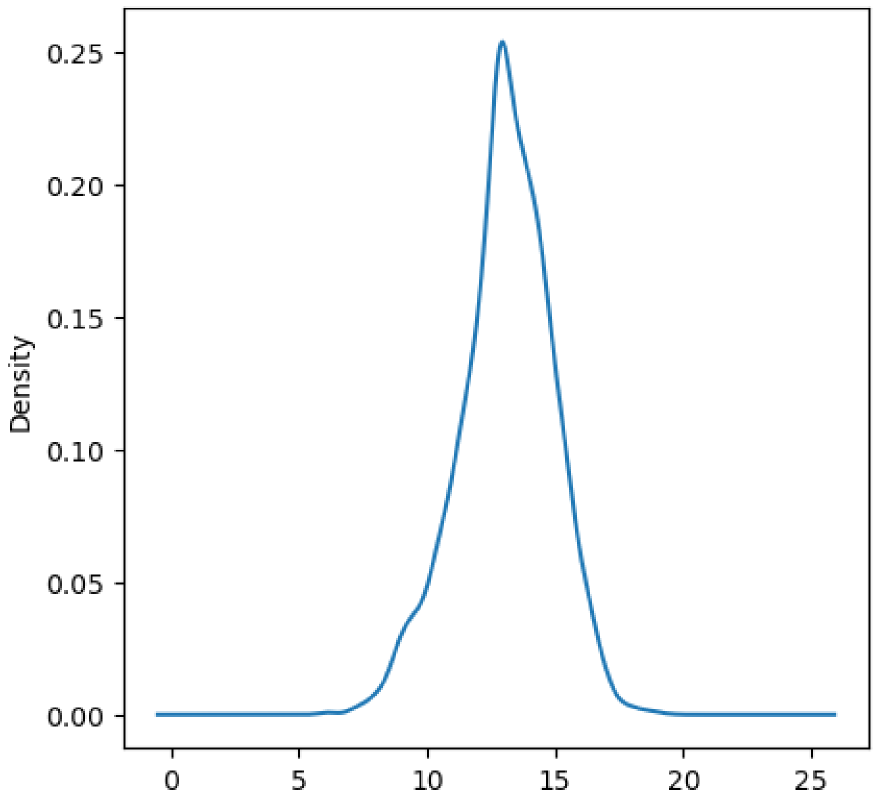
[94]:
```
plt.hist(final_preop_cases['ane_type'])
```

[94]:
```
(array([4315.,    0.,    0.,    0.,    0.,  208.,    0.,    0.,    0.,
           42.]),
 array([0. , 0.2, 0.4, 0.6, 0.8, 1. , 1.2, 1.4, 1.6, 1.8, 2. ]),
 <BarContainer object of 10 artists>)
```



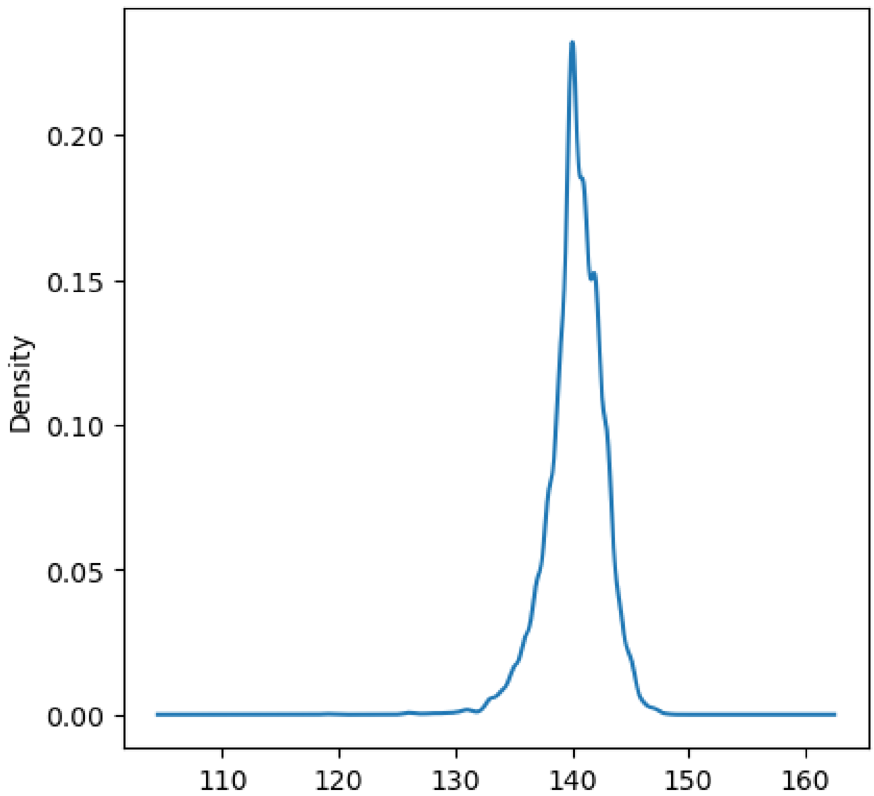

```
[95]: plt.figure(figsize=(5,5))
      final_preop_cases['preop_hb'].plot(kind='density')
      #hb:Hemoglobin (g/dl) [13-17 reference value]
```

```
[95]: <AxesSubplot:ylabel='Density'>
```



[96]:
```
plt.figure(figsize=(5,5))
final_preop_cases['preop_na'].plot(kind='density')
#na:sodium (mmol/l) [135-145 reference value]
```

[96]: <AxesSubplot:ylabel='Density'>



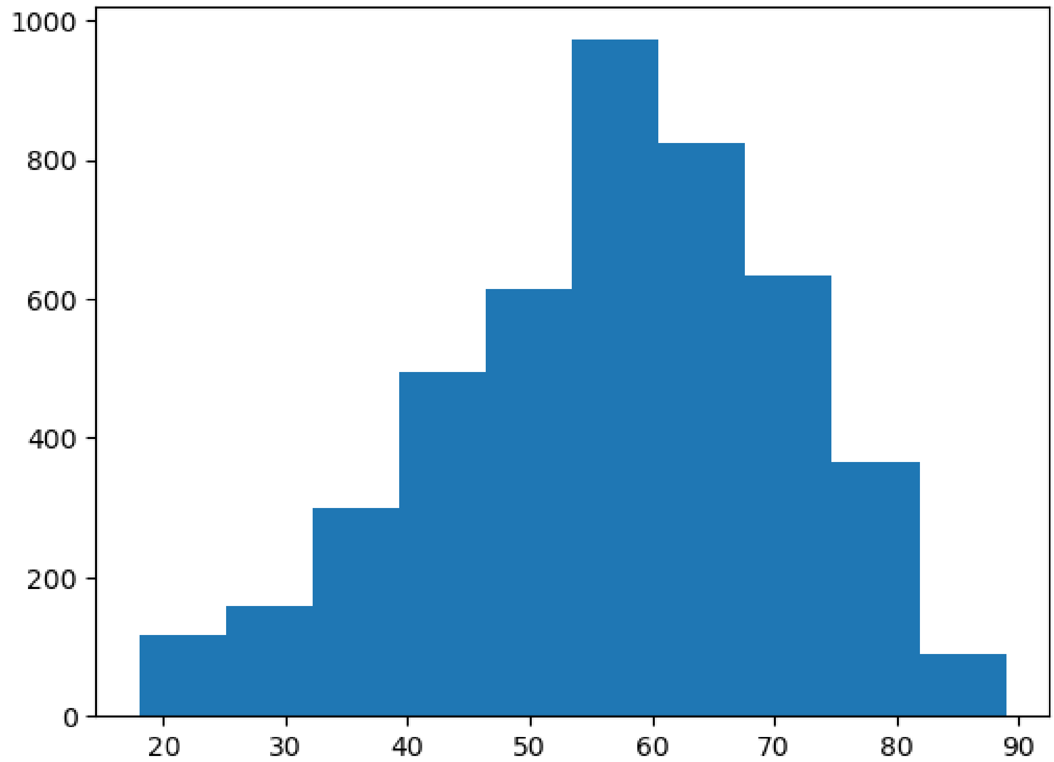
[97]:
```
#categorical and numerical columns
cat_cols=final_preop_cases.select_dtypes(include=['object']).columns
num_cols = final_preop_cases.select_dtypes(include=np.number).columns.tolist()
print("Categorical Variables:")
print(cat_cols)
print("Numerical Variables:")
print(num_cols)
```

```
Categorical Variables:
Index(['sex', 'department', 'optype', 'ane_type', 'preop_ecg', 'preop_pft'],
dtype='object')
Numerical Variables:
['outcome', 'age', 'bmi', 'asa', 'preop_htn', 'preop_dm', 'preop_hb',
'preop_plt', 'preop_pt', 'preop_aptt', 'preop_na', 'preop_k', 'preop_gluc',
'preop_alb', 'preop_ast', 'preop_alt', 'preop_bun', 'preop_cr']
```

[98]:
```
import matplotlib.pyplot as plt
plt.hist(final_preop_cases['age'])
```

[98]:
```
(array([117., 157., 299., 494., 613., 972., 825., 633., 366.,  89.]),
 array([18. , 25.1, 32.2, 39.3, 46.4, 53.5, 60.6, 67.7, 74.8, 81.9, 89. ]),
```

<BarContainer object of 10 artists>)



3.1 label encoding

[99]:
```
cols = ['sex', 'department', 'optype', 'ane_type', 'preop_ecg', 'preop_pft']
#
# Encode labels of multiple columns at once
#
final_preop_cases[cols] = final_preop_cases[cols].apply(LabelEncoder().
  ↪fit_transform)
#
# Print head
#
final_preop_cases.head()
```

[99]:

	outcome	age	bmi	asa	sex	department	optype	ane_type	preop_htn	\
0	0	77	26.3	2.0	1	0	2	0	1	
1	0	54	19.6	2.0	1	0	7	0	0	
2	0	62	24.4	1.0	1	0	0	0	0	
7	0	81	27.4	2.0	0	0	1	0	0	
8	0	32	20.4	1.0	0	0	0	0	0	

```
   preop_dm  preop_ecg  preop_pft  preop_hb  preop_plt  preop_pt  preop_aptt  \
0         0         10          6      14.1      189.0      94.0        33.2
1         0         10          6      10.2      251.0     110.0        31.9
2         0         10          6      14.2      373.0     103.0        30.3
7         0         10          6      12.1      186.0      92.0        31.3
8         0         10          6      13.7      141.0      96.0        32.1

   preop_na  preop_k  preop_gluc  preop_alb  preop_ast  preop_alt  preop_bun  \
0     141.0      3.1       134.0        4.3       18.0       16.0       10.0
1     143.0      4.7        88.0        3.8       18.0       15.0       14.0
2     144.0      4.9        87.0        4.2       17.0       34.0       14.0
7     142.0      4.5       101.0        3.7       16.0       10.0       11.0
8     140.0      4.2        91.0        4.5       14.0       11.0        8.0

   preop_cr
0      0.82
1      0.86
2      1.18
7      0.69
8      0.58
```

[33]:

```python
print(final_preop_cases.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4565 entries, 0 to 6387
Data columns (total 25 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   case_id     4565 non-null   int64
 1   outcome     4565 non-null   int64
 2   age         4565 non-null   int32
 3   bmi         4565 non-null   float64
 4   asa         4565 non-null   float64
 5   sex         4565 non-null   int32
 6   department  4565 non-null   int32
 7   optype      4565 non-null   int32
 8   ane_type    4565 non-null   int32
 9   preop_htn   4565 non-null   int64
 10  preop_dm    4565 non-null   int64
 11  preop_ecg   4565 non-null   int32
 12  preop_pft   4565 non-null   int32
 13  preop_hb    4565 non-null   float64
 14  preop_plt   4565 non-null   float64
 15  preop_pt    4565 non-null   float64
 16  preop_aptt  4565 non-null   float64
 17  preop_na    4565 non-null   float64
```

```
 18  preop_k     4565 non-null   float64
 19  preop_gluc  4565 non-null   float64
 20  preop_alb   4565 non-null   float64
 21  preop_ast   4565 non-null   float64
 22  preop_alt   4565 non-null   float64
 23  preop_bun   4565 non-null   float64
 24  preop_cr    4565 non-null   float64
dtypes: float64(14), int32(7), int64(4)
memory usage: 802.4 KB
None
```

[100]:
```
final_preop_cases['asa'] = final_preop_cases['asa'].astype(int)  # cast data␣
 ↪type to int
```

[101]:
```
final_preop_cases[['preop_hb', 'preop_plt', 'preop_pt',
                   'preop_aptt', 'preop_na', 'preop_k',
                   'preop_gluc', 'preop_alb', 'preop_ast', 'preop_alt',
                   'preop_bun', 'preop_cr','bmi']] =␣
 ↪final_preop_cases[['preop_hb', 'preop_plt', 'preop_pt',
                   'preop_aptt', 'preop_na', 'preop_k',
                   'preop_gluc', 'preop_alb', 'preop_ast', 'preop_alt',
                   'preop_bun', 'preop_cr','bmi']].round(3)
```

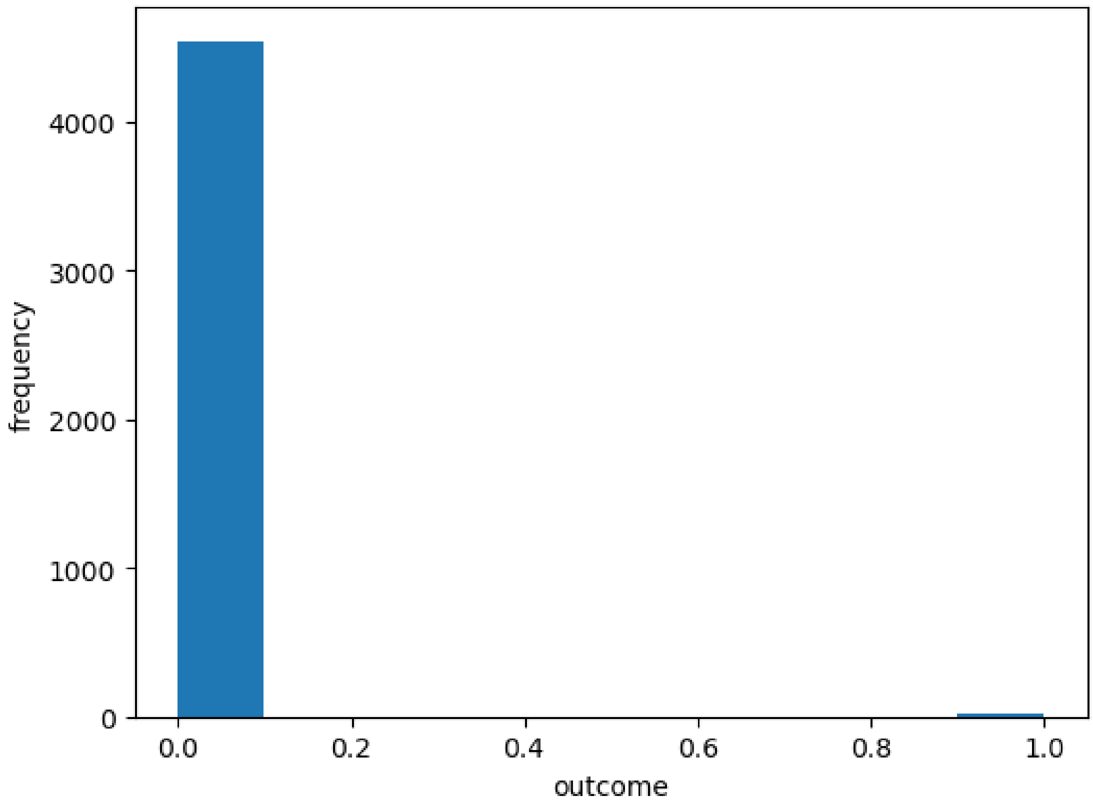
[102]:
```
print(final_preop_cases[['preop_hb','preop_cr']])
```

```
      preop_hb  preop_cr
0         14.1      0.82
1         10.2      0.86
2         14.2      1.18
7         12.1      0.69
8         13.7      0.58
…          …       …
6383      14.5      0.99
6384      15.2      0.84
6385      12.6      0.66
6386      12.5      0.65
6387       8.6      0.64

[4565 rows x 2 columns]
```

[103]:
```
#unbalanced data plot histogram
plt.hist(final_preop_cases['outcome' ])
plt.xlabel('outcome')
plt.ylabel('frequency')
final_preop_cases['outcome'].value_counts()
```

[103]:

```
0    4547
1      18
Name: outcome, dtype: int64
```



3.2 features and label

[104]:

```
X = final_preop_cases.drop(columns=['outcome','age','sex'],axis=1)
y = final_preop_cases['outcome']
```

[105]:

```
print(len(X))
```

```
4565
```

4 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a dimensionality reduction technique helps in reducing the number of features (dimensions) within a dataset while retaining the most important information. To explore and understand the underlying structure of the data.

[106]:

```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
```

```
# Assuming 'X' contains your 23-dimensional feature data
# Standardize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Apply PCA to reduce the dimensionality to 15
pca = PCA(n_components=15,random_state=42)
X_pca = pca.fit_transform(X_scaled)

# X_pca now contains the top 15 principal components, which can be considered␣
↪as the selected top 15 features
```

5 Balance the data

SMOTE (Synthetic Minority Over-sampling Technique) is a technique used to address class imbalance in a dataset. As, in our dataset ,the number of survivors exceeds the number of fatalities."

[107]:
```
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split

# Apply SMOTE to the entire dataset
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_pca, y)

# Split the resampled data into training (70%), validation (15%), and test␣
↪(15%) sets
X_train, X_temp, y_train, y_temp = train_test_split(X_resampled, y_resampled,␣
↪test_size=0.3, random_state=42)
X_validation, X_test, y_validation, y_test = train_test_split(X_temp, y_temp,␣
↪test_size=0.5, random_state=42)

# Now, you have balanced training data in X_train, y_train,
# and validation/test sets in X_validation, X_test, y_validation, y_test
```

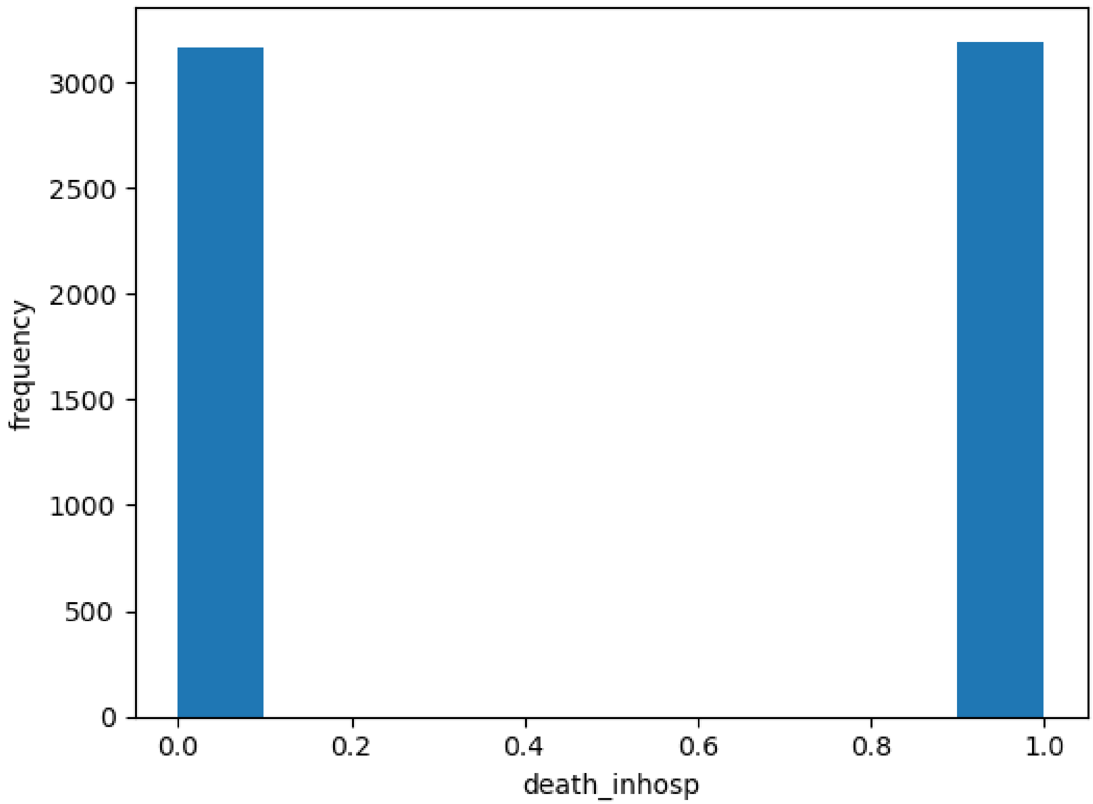
[108]:
```
print("Training Set   :: X_train :",X_train.shape, " y_train :",y_train.shape)
print("Validation Set :: X_ validation: ",X_validation.shape," y_validation: "␣
↪,y_validation.shape)
print("Testing Set    :: X_test :",X_test.shape," y_test:" ,y_test.shape)
```

```
Training Set   :: X_train : (6365, 15)  y_train : (6365,)
Validation Set :: X_ validation:  (1364, 15)  y_validation:  (1364,)
Testing Set    :: X_test : (1365, 15)  y_test: (1365,)
```

[110]:
```
#balanced data plot histogram
plt.hist(y_train)
```

```
plt.xlabel('death_inhosp')
plt.ylabel('frequency')
print("No. of deaths in balanced data(training set) ",sum(y_train==1))
print("No. of alive in balanced data(training set) ",sum(y_train==0))
```

```
No. of deaths in balanced data(training set)  3197
No. of alive in balanced data(training set)  3168
```



[111]:
```
print("No. of deaths in testing set ",sum(y_test==1))
print("No. of alive in testing set ",sum(y_test==0))
```

```
No. of deaths in testing set  676
No. of alive in testing set  689
```

[112]:
```
print("No. of deaths in validation set ",sum(y_validation==1))
print("No. of alive in validation set ",sum(y_validation==0))
```

```
No. of deaths in validation set  674
No. of alive in validation set  690
```

6 lazypredict

We have used lazyClassifier for quick model selection(comprehensive overview of various machine learning models' performances) and top 4 models which performed best here would be used for further Hyperparameter tuning

[113]:
```
pip install lazypredict
```

```
Requirement already satisfied: lazypredict in c:\users\ghild\anaconda3\lib\site-
packages (0.2.12)
Requirement already satisfied: joblib in c:\users\ghild\anaconda3\lib\site-
packages (from lazypredict) (1.2.0)
Requirement already satisfied: scikit-learn in
c:\users\ghild\anaconda3\lib\site-packages (from lazypredict) (1.0.2)
Requirement already satisfied: xgboost in c:\users\ghild\anaconda3\lib\site-
packages (from lazypredict) (2.0.3)
Requirement already satisfied: pandas in c:\users\ghild\anaconda3\lib\site-
packages (from lazypredict) (1.4.2)
Requirement already satisfied: tqdm in c:\users\ghild\anaconda3\lib\site-
packages (from lazypredict) (4.64.0)
Requirement already satisfied: click in c:\users\ghild\anaconda3\lib\site-
packages (from lazypredict) (8.0.4)
Requirement already satisfied: lightgbm in c:\users\ghild\anaconda3\lib\site-
packages (from lazypredict) (4.3.0)
Requirement already satisfied: colorama in c:\users\ghild\anaconda3\lib\site-
packages (from click->lazypredict) (0.4.4)
Requirement already satisfied: numpy in c:\users\ghild\anaconda3\lib\site-
packages (from lightgbm->lazypredict) (1.22.4)
Requirement already satisfied: scipy in c:\users\ghild\anaconda3\lib\site-
packages (from lightgbm->lazypredict) (1.7.3)
Requirement already satisfied: python-dateutil>=2.8.1 in
c:\users\ghild\anaconda3\lib\site-packages (from pandas->lazypredict) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
c:\users\ghild\anaconda3\lib\site-packages (from pandas->lazypredict) (2021.3)
Requirement already satisfied: six>=1.5 in c:\users\ghild\anaconda3\lib\site-
packages (from python-dateutil>=2.8.1->pandas->lazypredict) (1.16.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
c:\users\ghild\anaconda3\lib\site-packages (from scikit-learn->lazypredict)
(2.2.0)
Note: you may need to restart the kernel to use updated packages.
```

[114]:
```
import lazypredict
```

[115]:
```
from lazypredict.Supervised import LazyClassifier
```

[116]:
```
clf = LazyClassifier(verbose=0,ignore_warnings=True, custom_metric=None)
models,predictions = clf.fit(X_train, X_test, y_train, y_test)
```

```python
print(models)
```

```
100%|
   | 29/29 [00:12<00:00,  2.29it/s]
```

```
[LightGBM] [Info] Number of positive: 3197, number of negative: 3168
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of
testing was 0.000894 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 3825
[LightGBM] [Info] Number of data points in the train set: 6365, number of used
features: 15
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.502278 -> initscore=0.009112
[LightGBM] [Info] Start training from score 0.009112
                               Accuracy  Balanced Accuracy  ROC AUC  F1 Score  \
Model
ExtraTreesClassifier               1.00               1.00     1.00      1.00
LGBMClassifier                     0.99               0.99     0.99      0.99
RandomForestClassifier             0.99               0.99     0.99      0.99
XGBClassifier                      0.99               0.99     0.99      0.99
BaggingClassifier                  0.99               0.99     0.99      0.99
ExtraTreeClassifier                0.98               0.98     0.98      0.98
LabelPropagation                   0.98               0.98     0.98      0.98
LabelSpreading                     0.98               0.98     0.98      0.98
DecisionTreeClassifier             0.98               0.98     0.98      0.98
SVC                                0.98               0.98     0.98      0.98
QuadraticDiscriminantAnalysis      0.98               0.98     0.98      0.98
KNeighborsClassifier               0.95               0.96     0.96      0.95
AdaBoostClassifier                 0.94               0.94     0.94      0.94
NuSVC                              0.92               0.92     0.92      0.92
CalibratedClassifierCV             0.77               0.77     0.77      0.77
LogisticRegression                 0.77               0.77     0.77      0.77
BernoulliNB                        0.77               0.77     0.77      0.77
LinearSVC                          0.77               0.77     0.77      0.77
GaussianNB                         0.76               0.76     0.76      0.75
SGDClassifier                      0.75               0.75     0.75      0.75
LinearDiscriminantAnalysis         0.75               0.75     0.75      0.74
RidgeClassifier                    0.75               0.75     0.75      0.74
RidgeClassifierCV                  0.75               0.75     0.75      0.74
NearestCentroid                    0.74               0.74     0.74      0.74
Perceptron                         0.70               0.70     0.70      0.69
PassiveAggressiveClassifier        0.68               0.68     0.68      0.66
DummyClassifier                    0.50               0.50     0.50      0.33

                               Time Taken
Model
ExtraTreesClassifier                 0.33
```

```
LGBMClassifier                       0.33
RandomForestClassifier               1.62
XGBClassifier                        0.19
BaggingClassifier                    0.72
ExtraTreeClassifier                  0.01
LabelPropagation                     2.00
LabelSpreading                       2.73
DecisionTreeClassifier               0.12
SVC                                  0.58
QuadraticDiscriminantAnalysis        0.04
KNeighborsClassifier                 0.14
AdaBoostClassifier                   0.54
NuSVC                                1.71
CalibratedClassifierCV               0.94
LogisticRegression                   0.06
BernoulliNB                          0.02
LinearSVC                            0.26
GaussianNB                           0.01
SGDClassifier                        0.03
LinearDiscriminantAnalysis           0.05
RidgeClassifier                      0.03
RidgeClassifierCV                    0.03
NearestCentroid                      0.01
Perceptron                           0.03
PassiveAggressiveClassifier          0.07
DummyClassifier                      0.01
```

Top 3 models are : 1.ExtraTreeClassifier 2.LGBMClassifier 3.RandomForestClassifier

7 Hyperparameter tuning

7.1 1. ExtraTreeClassifier

[143]:
```
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.tree import ExtraTreeClassifier
from sklearn.metrics import accuracy_score

# Create the ExtraTreeClassifier
extra_tree_model = ExtraTreeClassifier(random_state=42)

# Define hyperparameters for tuning
param_grid = {
    'max_depth': [32],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

```
# Create GridSearchCV instance
grid_search = GridSearchCV(extra_tree_model, param_grid, cv=5,␣
 ↪scoring='accuracy', n_jobs=-1)

# Fit the model on the training data
grid_search.fit(X_train, y_train)

# Get the best hyperparameters
best_params = grid_search.best_params_
print(best_params)
# Train the ExtraTreeClassifier with the best hyperparameters on the combined␣
 ↪train and validation sets
best_extra_tree_model = ExtraTreeClassifier(**best_params, random_state=42)
best_extra_tree_model.fit(np.concatenate((X_train, X_validation)), np.
 ↪concatenate((y_train, y_validation)))

# Make predictions on the test set
predictions_test = best_extra_tree_model.predict(X_test)

# Make predictions on the validation set
predictions_validation =best_extra_tree_model.predict(X_validation)



y_pred_prob_ml1 = best_extra_tree_model.predict_proba(X_test)[:, 1]  #  [:, 1]␣
 ↪for the positive class label

# y_pred_prob_ml1 now contains the predicted probabilities for the positive␣
 ↪class


# Evaluate the model on the test set
accuracy_test_etc = accuracy_score(y_test, predictions_test)
print(f"Test Accuracy: {accuracy_test_etc}")



# Measure accuracy on the validation set
val_accuracy = accuracy_score(y_validation, predictions_validation)
print("Validation Accuracy :", val_accuracy)
```

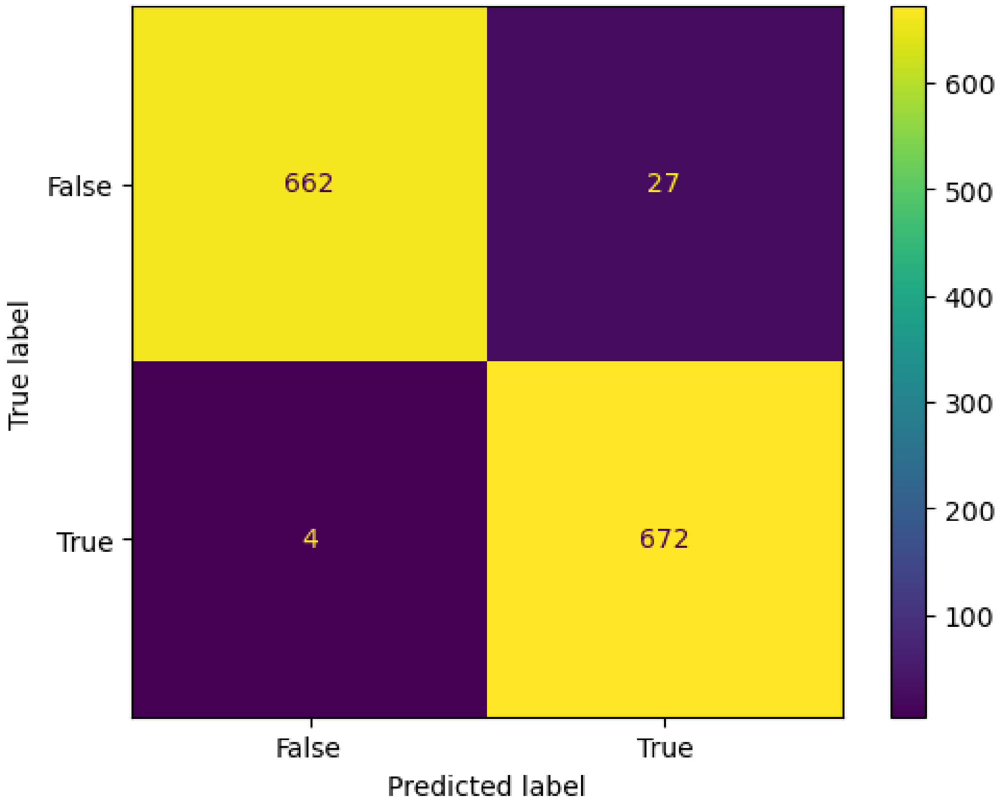
```
{'max_depth': 32, 'min_samples_leaf': 1, 'min_samples_split': 2}
Test Accuracy: 0.9772893772893773
Validation Accuracy : 0.999266862170088
```

7.1.1 confusion matrix of ExtraTreeClassifier

[163]:
```
from sklearn.metrics import confusion_matrix, classification_report


# Confusion Matrix
conf_matrix_etc = confusion_matrix(y_test, predictions_test)
print("Confusion Matrix:")
print(conf_matrix)

# Classification Report
class_report = classification_report(y_test, predictions_test)
print("Classification Report:")
print(class_report)
```

```
Confusion Matrix:
[[651  38]
 [  4 672]]
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.96      0.98       689
           1       0.96      0.99      0.98       676

    accuracy                           0.98      1365
   macro avg       0.98      0.98      0.98      1365
weighted avg       0.98      0.98      0.98      1365
```

[164]:
```
cm_display = ConfusionMatrixDisplay(confusion_matrix = conf_matrix_etc,␣
 ↪display_labels = [False, True])

cm_display.plot()
plt.show()
```



7.2 2. LGBMClassifier

[171]:
```
import lightgbm as lgb
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score

# Assuming you already have your feature and target variables: X_train,␣
↪y_train, X_test, y_test, X_validation, y_validation

# Convert the datasets to LightGBM dataset format
train_data = lgb.Dataset(X_train, label=y_train)
test_data = lgb.Dataset(X_test, label=y_test)
val_data = lgb.Dataset(X_validation, label=y_validation)

# Define the parameter grid for hyperparameter tuning
param_grid = {
    'num_leaves': [31, 64],
    'learning_rate': [0.1, 0.01],
    'n_estimators': [100, 200]
}
```

```python
# Initialize the LGBMClassifier
lgbm = lgb.LGBMClassifier()

# Initialize GridSearchCV with the classifier and parameter grid
grid_search = GridSearchCV(estimator=lgbm, param_grid=param_grid, cv=3,␣
↪n_jobs=-1, scoring='accuracy')

# Perform the grid search on the training dataset
grid_search.fit(X_train, y_train)

# Get the best hyperparameters from the grid search
best_params = grid_search.best_params_
print(best_params)
# Use the best hyperparameters to train the LGBMClassifier
best_lgbm = lgb.LGBMClassifier(**best_params)
best_lgbm.fit(X_train, y_train)


y_pred_prob_ml2 = best_lgbm.predict_proba(X_test)[:, 1]
# Make predictions on the validation set
y_pred_val_lgb = best_lgbm.predict(X_validation)



# Evaluate the model's accuracy on the validation set
accuracy_val_lgb = accuracy_score(y_validation, y_pred_val_lgb)
print("Validation Set Accuracy:", accuracy_val_lgb)

# predict the results
y_pred_lgb=best_lgbm.predict(X_test)
print(y_pred_lgb)

# view accuracy
from sklearn.metrics import accuracy_score
test_accuracy_lgb=accuracy_score(y_pred_lgb, y_test)
print('LightGBM Model accuracy score: {0:0.4f}'.format(test_accuracy_lgb))
```

```
[LightGBM] [Info] Number of positive: 3197, number of negative: 3168
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of
testing was 0.000603 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 3825
[LightGBM] [Info] Number of data points in the train set: 6365, number of used
features: 15
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.502278 -> initscore=0.009112
[LightGBM] [Info] Start training from score 0.009112
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
{'learning_rate': 0.1, 'n_estimators': 200, 'num_leaves': 64}
[LightGBM] [Info] Number of positive: 3197, number of negative: 3168
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of
testing was 0.000702 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 3825
[LightGBM] [Info] Number of data points in the train set: 6365, number of used
features: 15
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.502278 -> initscore=0.009112
[LightGBM] [Info] Start training from score 0.009112
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
Validation Set Accuracy: 0.9970674486803519
```

```
[1 1 1 … 1 1 1]
LightGBM Model accuracy score: 0.9971
```

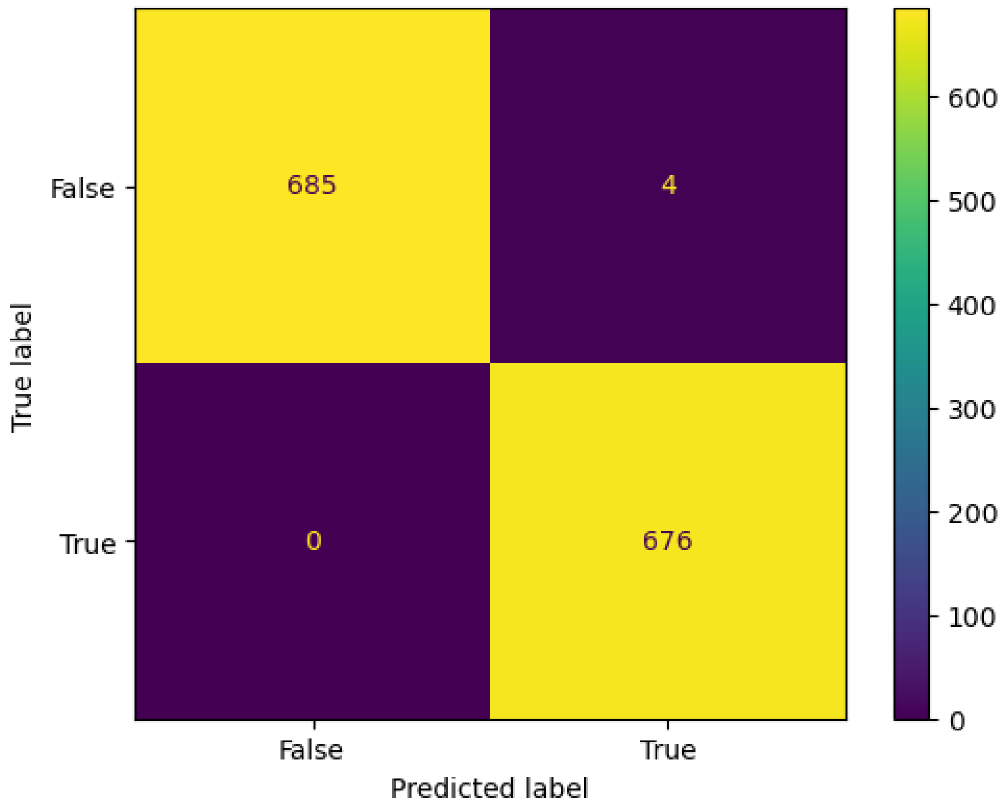
#best parameters of LGBM:{'learning_rate': 0.1, 'n_estimators': 200, 'num_leaves': 64}

7.2.1 confusion matrix of LGBMClassifier

[161]:
```
from sklearn.metrics import confusion_matrix, classification_report
# Confusion Matrix
conf_matrix_lgb = confusion_matrix(y_test, y_pred_lgb)
print("Confusion Matrix:")
print(conf_matrix)

# Classification Report
class_report = classification_report(y_test, y_pred_lgb)
print("Classification Report:")
print(class_report)
```

```
Confusion Matrix:
[[651  38]
 [  4 672]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.99      1.00       689
           1       0.99      1.00      1.00       676

    accuracy                           1.00      1365
   macro avg       1.00      1.00      1.00      1365
weighted avg       1.00      1.00      1.00      1365
```

[162]:
```
cm_display = ConfusionMatrixDisplay(confusion_matrix = conf_matrix_lgb,␣
 ↪display_labels = [False, True])

cm_display.plot()
plt.show()
```



7.3 3. RandomForestClassifier

[172]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.metrics import accuracy_score
from scipy.stats import randint

param_dist = {
    "n_estimators": randint(10, 100),
    "max_depth": randint(1, 20),
    "min_samples_split": randint(2, 20),
    "min_samples_leaf": randint(1, 20),
    "bootstrap": [ False,True]
}

# Instantiate the RandomForestClassifier

# Instantiate the RandomForestClassifier with the hyperparameter bootstrap set␣
↪to False
rfc = RandomForestClassifier(random_state=42)
```

```
# Perform Randomized Search to find the best hyperparameters
random_search = RandomizedSearchCV(rfc, param_distributions=param_dist,␣
↪n_iter=10, cv=5, random_state=42, n_jobs=-1)
random_search.fit(X_train, y_train)

# Get the best estimator
best_rf = random_search.best_estimator_
print("best rf:",best_rf)
# Make predictions on the validation set using the best estimator
y_pred_val_rf = best_rf.predict(X_validation)


y_pred_prob_ml3  = best_rf.predict_proba(X_test)[:, 1]

# Measure accuracy on the validation set
val_accuracy_rf = accuracy_score(y_validation, y_pred_val_rf)
print("RandomForestClassifier Validation Accuracy with Hyperparameter Tuning:",␣
↪val_accuracy_rf)

# Make predictions on the test set using the best estimator
y_pred_test_rf= best_rf.predict(X_test)

# Measure accuracy on the test set
test_accuracy_rf = accuracy_score(y_test, y_pred_test_rf)
print("RandomForestClassifier Test Accuracy with Hyperparameter Tuning:",␣
↪test_accuracy_rf)
```

```
best rf: RandomForestClassifier(bootstrap=False, max_depth=15,
min_samples_leaf=11,
                       min_samples_split=9, n_estimators=70, random_state=42)
RandomForestClassifier Validation Accuracy with Hyperparameter Tuning:
0.9611436950146628
RandomForestClassifier Test Accuracy with Hyperparameter Tuning:
0.9692307692307692
```

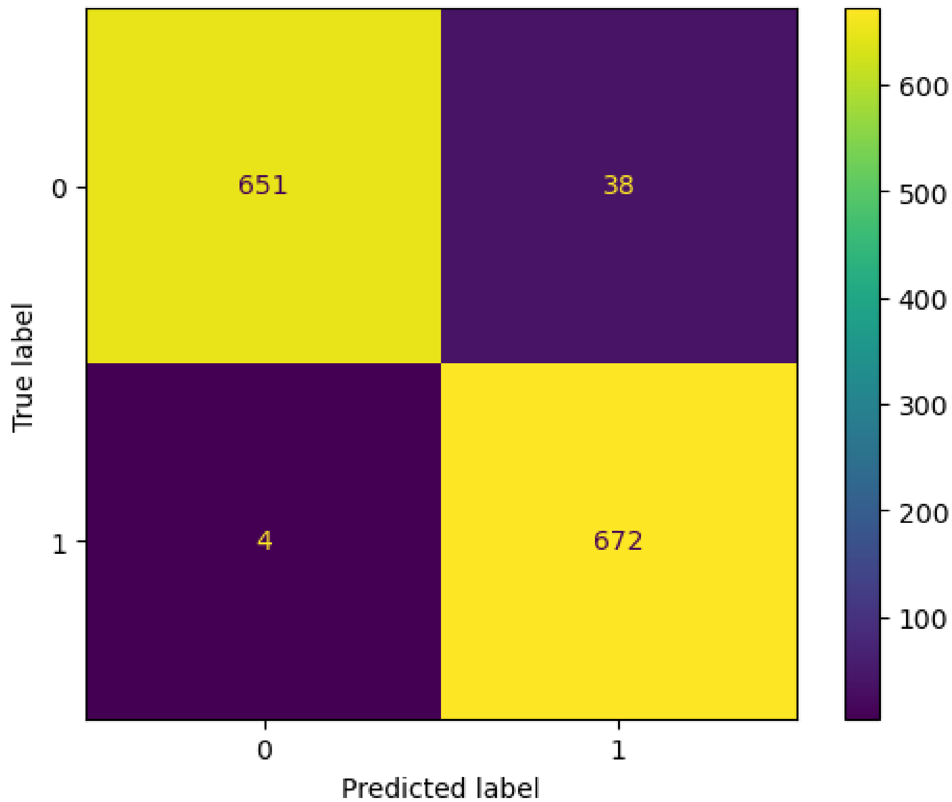
7.3.1 confusion matrix of RandomForestClassifier

[158]:
```
from sklearn.metrics import confusion_matrix, classification_report
# Confusion Matrix
conf_matrix_rf = confusion_matrix(y_test, y_pred_test_rf)
print("Confusion Matrix:")
print(conf_matrix)

# Classification Report
class_report = classification_report(y_test, y_pred_test_rf)
```

```
print("Classification Report:")
print(class_report)
```

```
Confusion Matrix:
[[651  38]
 [  4 672]]
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.94      0.97       689
           1       0.95      0.99      0.97       676

    accuracy                           0.97      1365
   macro avg       0.97      0.97      0.97      1365
weighted avg       0.97      0.97      0.97      1365
```

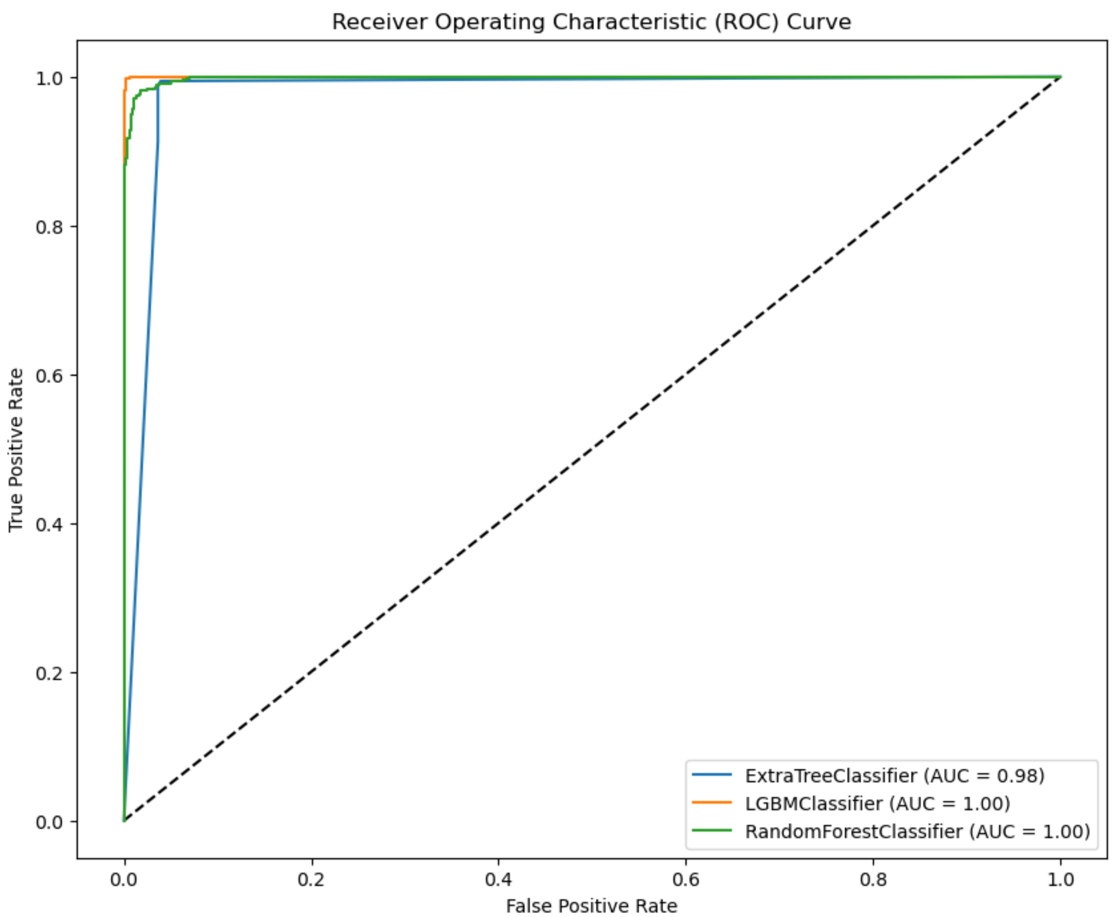
[170]:
```
cm_display = ConfusionMatrixDisplay(confusion_matrix = conf_matrix_rf,␣
 ↪display_labels = [0, 1])

cm_display.plot()
plt.show()
#0:patient survived 1: patient dead
#TN:651,FP:38,FN:4,TP:672
```



7.4 ROC-AUC curve

[151]:
```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

# Assuming you have the predicted probabilities for each algorithm named as␣
↪y_pred_prob_ml1, y_pred_prob_ml2, etc.
# And the true labels named as y_true

# Plotting the ROC curves for each algorithm
plt.figure(figsize=(10, 8))
plt.plot([0, 1], [0, 1], 'k--')

# Calculating and plotting ROC curve for EXtraTreeClassifier
fpr_ml1, tpr_ml1, _ = roc_curve(y_test, y_pred_prob_ml1)
roc_auc_ml1 = auc(fpr_ml1, tpr_ml1)
plt.plot(fpr_ml1, tpr_ml1, label='ExtraTreeClassifier (AUC = %0.2f)' %␣
↪roc_auc_ml1)

# Calculating and plotting ROC curve for LGBMClassifier
fpr_ml2, tpr_ml2, _ = roc_curve(y_test, y_pred_prob_ml2)
```

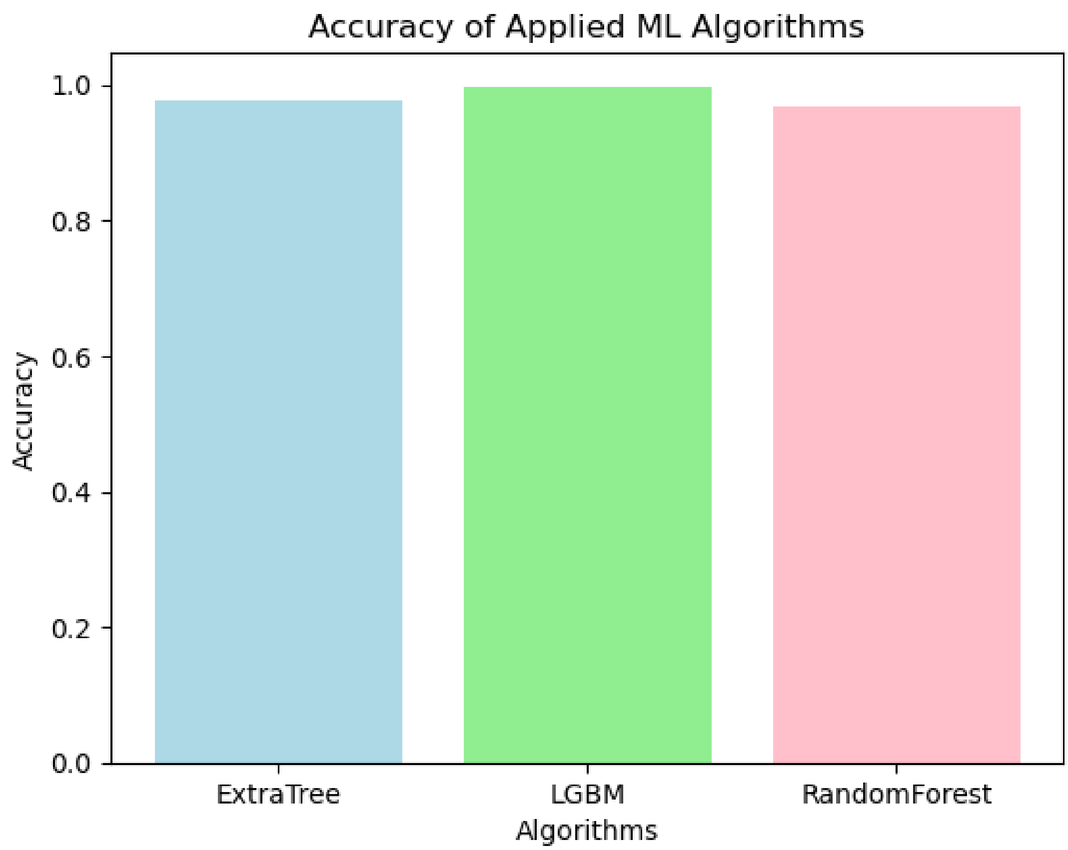
```
roc_auc_ml2 = auc(fpr_ml2, tpr_ml2)
plt.plot(fpr_ml2, tpr_ml2, label='LGBMClassifier (AUC = %0.2f)' % roc_auc_ml2)

# Calculating and plotting ROC curve for RandomForestClassifier
fpr_ml3, tpr_ml3, _ = roc_curve(y_test, y_pred_prob_ml3)
roc_auc_ml3 = auc(fpr_ml3, tpr_ml3)
plt.plot(fpr_ml3, tpr_ml3, label='RandomForestClassifier (AUC = %0.2f)' %␣
↪roc_auc_ml3)



# Adding titles and labels
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")

# Show the plot
plt.show()
```



This indicates that the EXtraTreeClassifier model has very high discrimination ability, with a strong ability to distinguish between the positive and negative classes.

7.5 Comparison of Accuracies

[155]:
```
import matplotlib.pyplot as plt

# Assuming you have the accuracy values of the four ML algorithms stored in a␣
↪list or array
accuracies = [accuracy_test_etc, test_accuracy_lgb, test_accuracy_rf]  #␣
↪Replace with actual accuracy values

# Names of the machine learning algorithms
labels = ['ExtraTree', 'LGBM', 'RandomForest']

# Plotting the accuracy scores
plt.bar(labels, accuracies, color=['lightblue', 'lightgreen',  'pink'])

# Adding titles and labels
plt.title('Accuracy of Applied ML Algorithms')
plt.xlabel('Algorithms')
plt.ylabel('Accuracy')

# Displaying the plot
plt.show()
```



7.6 conclusion

Based on our analysis, the ExtraTreeClassifier has emerged as the most effective model among the three. With a test accuracy of 0.977 and a robust ROC-AUC score of 0.98, it consistently demonstrates superior performance in distinguishing between the classes it is trained to predict. The crucial insight from the confusion matrix of this model is its ability to accurately identify cases where a patient would unfortunately pass away, a pivotal factor in a medical context.

In summary, the ExtraTreeClassifier, configured with a 'max_depth' of 32, 'min_samples_leaf' of 1, and 'min_samples_split' of 2, exhibits exceptional accuracy in predicting patient outcomes, particularly in accurately identifying cases where a patient would not survive.

7.7 Future scope

Exploring the inclusion of emergency cases and underage patients can be a valuable expansion for this analysis. Incorporating these additional demographics could provide a more comprehensive understanding of predictive capabilities across a wider range of scenarios. Moreover, extending the analysis to cover both intraoperative and postoperative periods can offer insights into the performance of the model at different stages of patient care, potentially enhancing its overall utility and effectiveness.