ml-final-project-5

May 17, 2024

[424]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('fivethirtyeight')
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

[425]:
```
# Loading the trainig dataset into Pandas Dataframes
df = pd.read_csv("titanic.csv")
df
```

[425]:

	pclass	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	survived
0	1	Allen, Miss. Elisabeth Walton	female	29.0000	0	0	24160	211.3375	B5	S	1
1	1	Allison, Master. Hudson Trevor	male	0.9167	1	2	113781	151.5500	C22 C26	S	1
2	1	Allison, Miss. Helen Loraine	female	2.0000	1	2	113781	151.5500	C22 C26	S	0
3	1	Allison, Mr. Hudson Joshua Creighton	male	30.0000	1	2	113781	151.5500	C22 C26	S	0
4	1	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0000	1	2	113781	151.5500	C22 C26	S	0
…	…	…	…	…	…	…	…	…	…	…	…
1304	3	Zabour, Miss. Hileni	female	14.5000	1	0	2665	14.4542	NaN	C	0
1305	3	Zabour, Miss. Thamine	female	NaN	1	0	2665	14.4542	NaN	C	0
1306	3	Zakarian, Mr. Mapriededer	male	26.5000	0	0	2656	7.2250	NaN	C	0
1307	3	Zakarian, Mr. Ortin	male	27.0000	0	0	2670	7.2250	NaN	C	0
1308	3	Zimmerman, Mr. Leo	male								

```
1308  29.0000      0      0  315082    7.8750      NaN        S         0

[1309 rows x 11 columns]
```

1 Exploratory data analysis

[426]:
```
df.shape
```

[426]:
```
(1309, 11)
```

[427]:
```
df.head()
```

[427]:
```
   pclass                                            name     sex      age  \
0       1                    Allen, Miss. Elisabeth Walton  female  29.0000
1       1                   Allison, Master. Hudson Trevor    male   0.9167
2       1                     Allison, Miss. Helen Loraine  female   2.0000
3       1             Allison, Mr. Hudson Joshua Creighton    male  30.0000
4       1  Allison, Mrs. Hudson J C (Bessie Waldo Daniels)  female  25.0000

   sibsp  parch  ticket      fare    cabin embarked  survived
0      0      0   24160  211.3375       B5        S         1
1      1      2  113781  151.5500  C22 C26        S         1
2      1      2  113781  151.5500  C22 C26        S         0
3      1      2  113781  151.5500  C22 C26        S         0
4      1      2  113781  151.5500  C22 C26        S         0
```

[428]:
```
df.tail()
```

[428]:
```
      pclass                        name     sex   age  sibsp  parch  ticket  \
1304       3       Zabour, Miss. Hileni  female  14.5      1      0    2665
1305       3      Zabour, Miss. Thamine  female   NaN      1      0    2665
1306       3  Zakarian, Mr. Mapriededer    male  26.5      0      0    2656
1307       3        Zakarian, Mr. Ortin    male  27.0      0      0    2670
1308       3         Zimmerman, Mr. Leo    male  29.0      0      0  315082

         fare cabin embarked  survived
1304  14.4542   NaN        C         0
1305  14.4542   NaN        C         0
1306   7.2250   NaN        C         0
1307   7.2250   NaN        C         0
1308   7.8750   NaN        S         0
```

[429]:
```
df.sample()
```

[429]:
```
     pclass                          name   sex   age  sibsp  parch  ticket  \
968       3  Lindell, Mr. Edvard Bengtsson  male  36.0      1      0  349910
```

```
      fare cabin embarked  survived
968  15.55   NaN        S         0
```

[430]:
```
columns_list = list(df.columns)
columns_list
```

[430]:
```
['pclass',
 'name',
 'sex',
 'age',
 'sibsp',
 'parch',
 'ticket',
 'fare',
 'cabin',
 'embarked',
 'survived']
```

[431]:
```
df.describe()
```

[431]:
```
             pclass          age        sibsp        parch         fare  \
count  1309.000000  1046.000000  1309.000000  1309.000000  1308.000000
mean      2.294882    29.881135     0.498854     0.385027    33.295479
std       0.837836    14.413500     1.041658     0.865560    51.758668
min       1.000000     0.166700     0.000000     0.000000     0.000000
25%       2.000000    21.000000     0.000000     0.000000     7.895800
50%       3.000000    28.000000     0.000000     0.000000    14.454200
75%       3.000000    39.000000     1.000000     0.000000    31.275000
max       3.000000    80.000000     8.000000     9.000000   512.329200

          survived
count  1309.000000
mean      0.381971
std       0.486055
min       0.000000
25%       0.000000
50%       0.000000
75%       1.000000
max       1.000000
```

[432]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
Data columns (total 11 columns):
 #   Column    Non-Null Count  Dtype
```

```
---  ------    --------------  -----
 0   pclass    1309 non-null   int64
 1   name      1309 non-null   object
 2   sex       1309 non-null   object
 3   age       1046 non-null   float64
 4   sibsp     1309 non-null   int64
 5   parch     1309 non-null   int64
 6   ticket    1309 non-null   object
 7   fare      1308 non-null   float64
 8   cabin     295 non-null    object
 9   embarked  1307 non-null   object
 10  survived  1309 non-null   int64
dtypes: float64(2), int64(4), object(5)
memory usage: 112.6+ KB
```

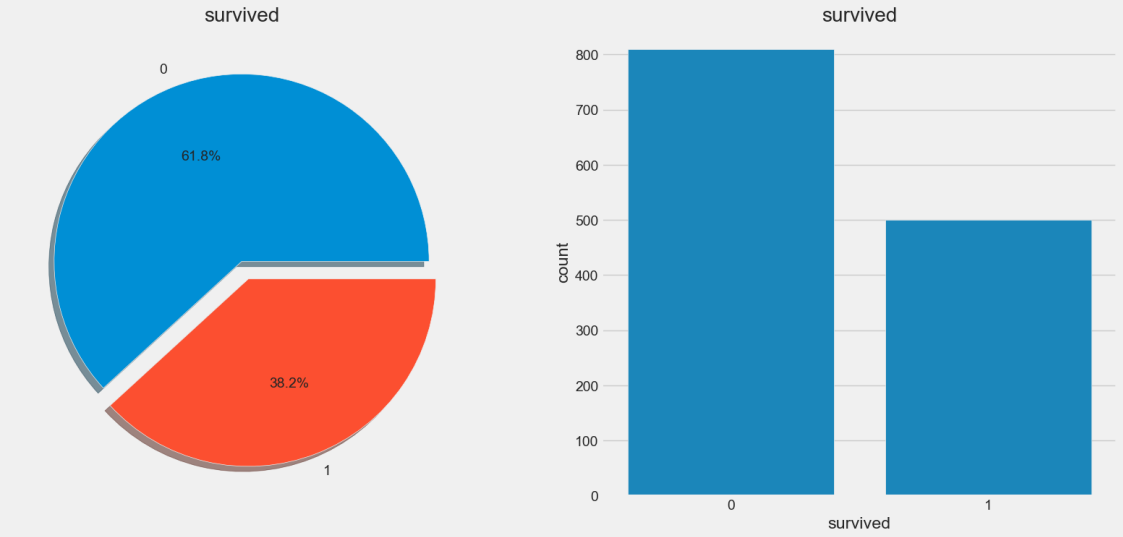
[433]:
```
df.isnull().sum()
```

[433]:
```
pclass          0
name            0
sex             0
age           263
sibsp           0
parch           0
ticket          0
fare            1
cabin        1014
embarked        2
survived        0
dtype: int64
```

[434]:
```
df.duplicated().sum()
```

[434]:
```
0
```

[435]:
```
f,ax=plt.subplots(1,2,figsize=(18,8))
df['survived'].value_counts().plot.pie(explode=[0,0.1],autopct='%1.
  ↪1f%%',ax=ax[0],shadow=True)
ax[0].set_title('survived')
ax[0].set_ylabel('')
sns.countplot(x='survived',data=df,ax=ax[1])
ax[1].set_title('survived')
plt.show()
```



2 Explore categorical variables

[436]:
```
# find categorical variables
categorical = [var for var in df.columns if df[var].dtype=='O']
print('There are {} categorical variables\n'.format(len(categorical)))
print('The categorical variables are :\n\n', categorical)
```

```
There are 5 categorical variables

The categorical variables are :

 ['name', 'sex', 'ticket', 'cabin', 'embarked']
```

[437]:
```
# view the categorical variables
df[categorical].head()
```

[437]:

	name	sex	ticket	cabin	embarked
0	Allen, Miss. Elisabeth Walton	female	24160	B5	S
1	Allison, Master. Hudson Trevor	male	113781	C22 C26	S
2	Allison, Miss. Helen Loraine	female	113781	C22 C26	S
3	Allison, Mr. Hudson Joshua Creighton	male	113781	C22 C26	S
4	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	113781	C22 C26	S

```
4        S
```

[438]:
```
# check missing values in categorical variables
df[categorical].isnull().sum()
```

[438]:
```
name           0
sex            0
ticket         0
cabin       1014
embarked       2
dtype: int64
```

[439]:
```
# view frequency counts of values in categorical variables
for var in categorical:
    print(df[var].value_counts())
```

```
name
Connolly, Miss. Kate             2
Kelly, Mr. James                 2
Allen, Miss. Elisabeth Walton    1
Ilmakangas, Miss. Ida Livija     1
Ilieff, Mr. Ylio                 1
                                ..
Hart, Miss. Eva Miriam           1
Harris, Mr. Walter               1
Harris, Mr. George               1
Harper, Rev. John                1
Zimmerman, Mr. Leo               1
Name: count, Length: 1307, dtype: int64
sex
male      843
female    466
Name: count, dtype: int64
ticket
CA. 2343    11
1601         8
CA 2144      8
PC 17608     7
347077       7
            ..
373450       1
2223         1
350046       1
3101281      1
315082       1
Name: count, Length: 929, dtype: int64
cabin
```

```
C23 C25 C27        6
G6                 5
B57 B59 B63 B66    5
F4                 4
F33                4
                  ..
C132               1
E60                1
B52 B54 B56        1
C49                1
F38                1
Name: count, Length: 186, dtype: int64
embarked
S    914
C    270
Q    123
Name: count, dtype: int64
```

[440]:
```
# view frequency distribution of categorical variables
for var in categorical:
    frequency_distribution = df[var].value_counts() / float(len(df))
    print(f"Frequency distribution for {var}:")
    print(frequency_distribution, "\n")
```

```
Frequency distribution for name:
name
Connolly, Miss. Kate             0.001528
Kelly, Mr. James                 0.001528
Allen, Miss. Elisabeth Walton    0.000764
Ilmakangas, Miss. Ida Livija     0.000764
Ilieff, Mr. Ylio                 0.000764
                                    …
Hart, Miss. Eva Miriam           0.000764
Harris, Mr. Walter               0.000764
Harris, Mr. George               0.000764
Harper, Rev. John                0.000764
Zimmerman, Mr. Leo               0.000764
Name: count, Length: 1307, dtype: float64

Frequency distribution for sex:
sex
male      0.644003
female    0.355997
Name: count, dtype: float64

Frequency distribution for ticket:
ticket
CA. 2343    0.008403
```

```
1601        0.006112
CA 2144     0.006112
PC 17608    0.005348
347077      0.005348
              …
373450      0.000764
2223        0.000764
350046      0.000764
3101281     0.000764
315082      0.000764
Name: count, Length: 929, dtype: float64

Frequency distribution for cabin:
cabin
C23 C25 C27        0.004584
G6                 0.003820
B57 B59 B63 B66    0.003820
F4                 0.003056
F33                0.003056
                     …
C132               0.000764
E60                0.000764
B52 B54 B56        0.000764
C49                0.000764
F38                0.000764
Name: count, Length: 186, dtype: float64

Frequency distribution for embarked:
embarked
S    0.698243
C    0.206264
Q    0.093965
Name: count, dtype: float64
```

[441]:
```
# check for cardinality in categorical variables
for var in categorical:
    print(var, ' contains ', len(df[var].unique()), ' labels')
```

```
name  contains  1307  labels
sex  contains  2  labels
ticket  contains  929  labels
cabin  contains  187  labels
embarked  contains  4  labels
```

[442]:
```
# check missing values in categorical variables
df[categorical].isnull().sum()
```

[442]:
```
name            0
sex             0
ticket          0
cabin        1014
embarked        2
dtype: int64
```

[443]:
```
# Calculate the percentage of missing values in the categorical variables
missing_values = df[categorical].isnull().sum()
percentage_missing = ((missing_values / len(df)) * 100).
  ↪sort_values(ascending=False)[missing_values>0]
percentage_missing
```

[443]:
```
cabin       77.463713
embarked     0.152788
dtype: float64
```

3 Analysing The Categorical Features

[444]:
```
df.groupby(['sex','survived'])['survived'].count()
```

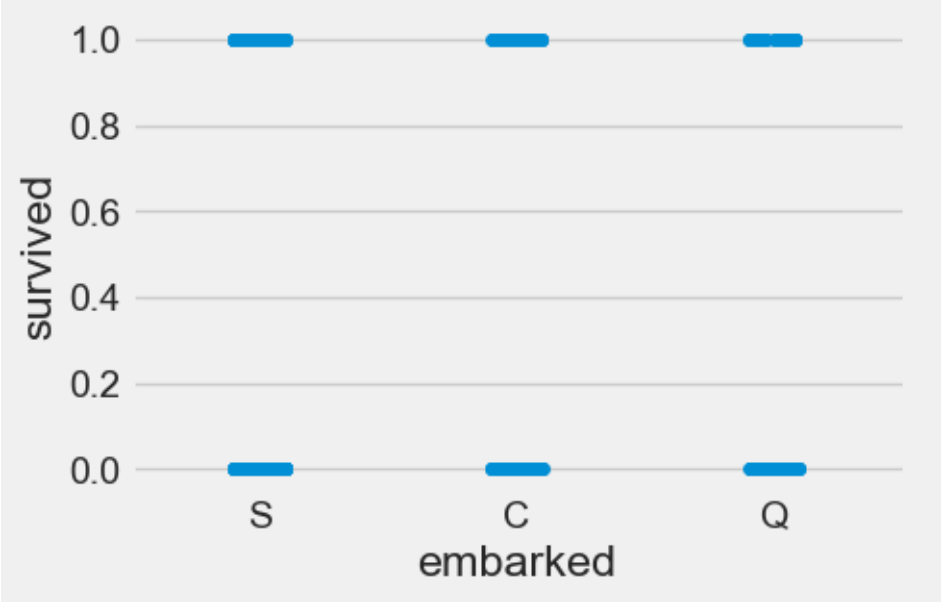
[444]:
```
sex     survived
female  0           127
        1           339
male    0           682
        1           161
Name: survived, dtype: int64
```

[445]:
```
pd.crosstab([df.sex,df.survived],df.pclass,margins=True).style.
  ↪background_gradient(cmap='summer_r')
```

[445]:
```
<pandas.io.formats.style.Styler at 0x2c1e5fe5690>
```

4 Embarked Feature

[446]:
```
sns.catplot(x='embarked',y='survived',data=df)
fig=plt.gcf()
fig.set_size_inches(5,3)
plt.show()
```



5 Explore Numerical Variables

[447]:
```
# find numerical variables
numerical = [var for var in df.columns if df[var].dtype!='O']
print('There are {} numerical variables\n'.format(len(numerical)))
print('The numerical variables are :', numerical)
```

```
There are 6 numerical variables

The numerical variables are : ['pclass', 'age', 'sibsp', 'parch', 'fare',
'survived']
```

[448]:
```
# view the numerical variables
df[numerical].head()
```

[448]:

	pclass	age	sibsp	parch	fare	survived
0	1	29.0000	0	0	211.3375	1
1	1	0.9167	1	2	151.5500	1
2	1	2.0000	1	2	151.5500	0
3	1	30.0000	1	2	151.5500	0
4	1	25.0000	1	2	151.5500	0

[449]:
```
# check missing values in numerical variables
df[numerical].isnull().sum()
```

[449]:
```
pclass        0
age         263
sibsp         0
parch         0
fare          1
survived      0
dtype: int64
```

[450]:
```
# Calculate the percentage of missing values in the numerical variables
missing_values = df[numerical].isnull().sum()
percentage_missing = ((missing_values / len(df)) * 100).
 ↪sort_values(ascending=False)[missing_values>0]
percentage_missing
```

[450]:
```
age     20.091673
fare     0.076394
dtype: float64
```

6 Analysing The Numerical Features

7 Pclass Feature

[451]:
```
pd.crosstab(df.pclass,df.survived,margins=True).style.
 ↪background_gradient(cmap='summer_r')
```

[451]:
```
<pandas.io.formats.style.Styler at 0x2c1ea83d010>
```

8 Age Feature

[452]:
```
print('Oldest Passenger was :',df['age'].max(),'Years')
print('Youngest Passenger was of:',df['age'].min(),'Years')
print('Average Age on the ship:',df['age'].mean(),'Years')
```

```
Oldest Passenger was : 80.0 Years
Youngest Passenger was of: 0.1667 Years
Average Age on the ship: 29.8811345124283 Years
```

9 SibSip Feature

[453]:
```
pd.crosstab([df.sibsp],df.survived).style.background_gradient(cmap='summer_r')
```

[453]:
```
<pandas.io.formats.style.Styler at 0x2c1ea45c650>
```

10 Parch Feature

[454]:
```
pd.crosstab(df.parch,df.pclass).style.background_gradient(cmap='summer_r')
```

[454]:
```
<pandas.io.formats.style.Styler at 0x2c1ea9b4510>
```

11 Date Preprocessing

12 Fill Null Values In Age Feature

[455]:
```
df['Initial']=0
for i in df:
    df['Initial']=df.name.str.extract('([A-Za-z]+)\.') #lets extract the␣
 ↪Salutations
```

[456]:
```
pd.crosstab(df.Initial,df.sex).T.style.background_gradient(cmap='summer_r')␣
 ↪#Checking the Initials with the Sex
```

[456]:
```
<pandas.io.formats.style.Styler at 0x2c1ea974450>
```

[457]:
```
df['Initial'].
 ↪replace(['Capt','Col','Countess','Don','Dona','Dr','Jonkheer','Lady','Major','Miss','Mlle',
```

[458]:
```
pd.crosstab(df.Initial,df.sex).T.style.background_gradient(cmap='summer_r')
```

[458]:
```
<pandas.io.formats.style.Styler at 0x2c1ea85b450>
```

[459]:
```
df.groupby('Initial')['age'].mean()
```

[459]:
```
Initial
Master     5.482704
Mr        32.545531
Mrs       37.046243
Ms        21.834502
Other     44.923077
Name: age, dtype: float64
```

[460]:
```
## Assigning the NaN Values with the Ceil values of the mean ages
df.loc[(df.age.isnull())&(df.Initial=='Mr'),'age']=33
df.loc[(df.age.isnull())&(df.Initial=='Mrs'),'age']=37
df.loc[(df.age.isnull())&(df.Initial=='Master'),'age']=5
df.loc[(df.age.isnull())&(df.Initial=='Ms'),'age']=22
df.loc[(df.age.isnull())&(df.Initial=='Other'),'age']=45
```

[461]:
```
df.age.isnull().any() #So no null values left finally
```

[461]: False

13 Filling Null Values In Embarked

[462]:
```
#maximum passengers boarded from Port S, we replace NaN with S
df['embarked'].fillna('S',inplace=True)
```

[463]:
```
df.embarked.isnull().any()
```

[463]: False

[464]:
```
df.isnull().sum()
```

[464]:
```
pclass         0
name           0
sex            0
age            0
sibsp          0
parch          0
ticket         0
fare           1
cabin       1014
embarked       0
survived       0
Initial        0
dtype: int64
```

[465]:
```
df.drop(['name','ticket','cabin'],axis=1,inplace=True)
df
```

[465]:

	pclass	sex	age	sibsp	parch	fare	embarked	survived	\
0	1	female	29.0000	0	0	211.3375	S	1	
1	1	male	0.9167	1	2	151.5500	S	1	
2	1	female	2.0000	1	2	151.5500	S	0	
3	1	male	30.0000	1	2	151.5500	S	0	
4	1	female	25.0000	1	2	151.5500	S	0	
…	…	…	…	…	…	…	…	…	
1304	3	female	14.5000	1	0	14.4542	C	0	
1305	3	female	22.0000	1	0	14.4542	C	0	
1306	3	male	26.5000	0	0	7.2250	C	0	
1307	3	male	27.0000	0	0	7.2250	C	0	
1308	3	male	29.0000	0	0	7.8750	S	0	

	Initial
0	Ms
1	Master

```
2          Ms
3          Mr
4         Mrs
…       …
1304       Ms
1305       Ms
1306       Mr
1307       Mr
1308       Mr

[1309 rows x 9 columns]
```

[466]:
```
# Drop rows with null values in the 'fare' column
df = df.dropna(subset=['fare'])
```

[467]:
```
df.isnull().sum()
```

[467]:
```
pclass      0
sex         0
age         0
sibsp       0
parch       0
fare        0
embarked    0
survived    0
Initial     0
dtype: int64
```

[468]:
```
# Calculate the correlation matrix
correlation_matrix = df.select_dtypes(include="number").corr()
correlation_matrix
```

[468]:

	pclass	age	sibsp	parch	fare	survived
pclass	1.000000	-0.375666	0.061162	0.018615	-0.558629	-0.312122
age	-0.375666	1.000000	-0.221380	-0.140208	0.167533	-0.068322
sibsp	0.061162	-0.221380	1.000000	0.373485	0.160238	-0.028122
parch	0.018615	-0.140208	0.373485	1.000000	0.221539	0.082418
fare	-0.558629	0.167533	0.160238	0.221539	1.000000	0.244265
survived	-0.312122	-0.068322	-0.028122	0.082418	0.244265	1.000000

14 Encode categorical variables

[469]:
```
df_encoded = pd.get_dummies(df, columns=['sex', 'embarked', 'Initial'])
df_encoded
```

[469]:
```
      pclass      age  sibsp  parch      fare  survived  sex_female  sex_male  \
0          1  29.0000      0      0  211.3375         1        True     False
1          1   0.9167      1      2  151.5500         1       False      True
2          1   2.0000      1      2  151.5500         0        True     False
3          1  30.0000      1      2  151.5500         0       False      True
4          1  25.0000      1      2  151.5500         0        True     False
…        …       …    …    …        …          …         …        …
1304       3  14.5000      1      0   14.4542         0        True     False
1305       3  22.0000      1      0   14.4542         0        True     False
1306       3  26.5000      0      0    7.2250         0       False      True
1307       3  27.0000      0      0    7.2250         0       False      True
1308       3  29.0000      0      0    7.8750         0       False      True

      embarked_C  embarked_Q  embarked_S  Initial_Master  Initial_Mr  \
0          False       False        True           False       False
1          False       False        True            True       False
2          False       False        True           False       False
3          False       False        True           False        True
4          False       False        True           False       False
…          …          …          …              …          …
1304        True       False       False           False       False
1305        True       False       False           False       False
1306        True       False       False           False        True
1307        True       False       False           False        True
1308       False       False        True           False        True

      Initial_Mrs  Initial_Ms  Initial_Other
0           False        True          False
1           False       False          False
2           False        True          False
3           False       False          False
4            True       False          False
…           …          …             …
1304        False        True          False
1305        False        True          False
1306        False       False          False
1307        False       False          False
1308        False       False          False

[1308 rows x 16 columns]
```

[470]:
```python
# Loop through the columns in df_encoded
for column in df_encoded.columns:
  if df_encoded[column].dtype == 'bool':
        # Convert boolean to int
        df_encoded[column] = df_encoded[column].astype(int)
```

```
print(df)
```

```
      pclass     sex      age  sibsp  parch      fare embarked  survived  \
0          1  female  29.0000      0      0  211.3375        S         1
1          1    male   0.9167      1      2  151.5500        S         1
2          1  female   2.0000      1      2  151.5500        S         0
3          1    male  30.0000      1      2  151.5500        S         0
4          1  female  25.0000      1      2  151.5500        S         0
…        …       …       …      …      …       …       …       …
1304       3  female  14.5000      1      0   14.4542        C         0
1305       3  female  22.0000      1      0   14.4542        C         0
1306       3    male  26.5000      0      0    7.2250        C         0
1307       3    male  27.0000      0      0    7.2250        C         0
1308       3    male  29.0000      0      0    7.8750        S         0

     Initial
0         Ms
1     Master
2         Ms
3         Mr
4        Mrs
…        …
1304      Ms
1305      Ms
1306      Mr
1307      Mr
1308      Mr

[1308 rows x 9 columns]
```

15 Remove Outliers Data

[471]:
```
from scipy.stats import zscore
# Calculate Z-Score for each numerical column
z_scores = np.abs(zscore(df.select_dtypes(include="number")))
# Set a threshold for Z-scores (e.g., 3 standard deviations)
threshold = 3
# Identify outliers
outliers_z = (z_scores > threshold).any(axis=1)
outliers_data = df_encoded[outliers_z]
outliers_data
```

[471]:
```
      pclass   age  sibsp  parch      fare  survived  sex_female  sex_male  \
0          1  29.0      0      0  211.3375         1           1         0
9          1  71.0      0      0   49.5042         0           0         1
10         1  47.0      1      0  227.5250         0           0         1
11         1  18.0      1      0  227.5250         1           1         0
```

```
14          1  80.0      0      0   30.0000         1           0         1
…       …   …    …    …       …       …          …       …
1179        3  33.0      1      9   69.5500         0           0         1
1180        3  37.0      1      9   69.5500         0           1         0
1210        3  40.0      1      4   27.9000         0           0         1
1211        3  45.0      1      4   27.9000         0           1         0
1235        3  74.0      0      0    7.7750         0           0         1

      embarked_C  embarked_Q  embarked_S  Initial_Master  Initial_Mr  \
0              0           0           1               0           0
9              1           0           0               0           1
10             1           0           0               0           0
11             1           0           0               0           0
14             0           0           1               0           1
…          …         …         …             …         …
1179           0           0           1               0           1
1180           0           0           1               0           0
1210           0           0           1               0           1
1211           0           0           1               0           0
1235           0           0           1               0           1

      Initial_Mrs  Initial_Ms  Initial_Other
0               0           1              0
9               0           0              0
10              0           0              1
11              1           0              0
14              0           0              0
…           …         …            …
1179            0           0              0
1180            1           0              0
1210            0           0              0
1211            1           0              0
1235            0           0              0

[103 rows x 16 columns]
```

[472]:
```
# Remove outliers from the DataFrame and create a new DataFrame without outliers
df_no_outliers = df_encoded[~outliers_z]
df_no_outliers
```

[472]:
```
      pclass      age  sibsp  parch      fare  survived  sex_female  sex_male  \
1          1   0.9167      1      2  151.5500         1           0         1
2          1   2.0000      1      2  151.5500         0           1         0
3          1  30.0000      1      2  151.5500         0           0         1
4          1  25.0000      1      2  151.5500         0           1         0
5          1  48.0000      0      0   26.5500         1           0         1
…      …       …    …    …       …       …          …       …
```

```
1304       3  14.5000      1      0   14.4542         0           1         0
1305       3  22.0000      1      0   14.4542         0           1         0
1306       3  26.5000      0      0    7.2250         0           0         1
1307       3  27.0000      0      0    7.2250         0           0         1
1308       3  29.0000      0      0    7.8750         0           0         1

      embarked_C  embarked_Q  embarked_S  Initial_Master  Initial_Mr  \
1              0           0           1               1           0
2              0           0           1               0           0
3              0           0           1               0           1
4              0           0           1               0           0
5              0           0           1               0           1
…            …           …           …               …           …
1304           1           0           0               0           0
1305           1           0           0               0           0
1306           1           0           0               0           1
1307           1           0           0               0           1
1308           0           0           1               0           1

      Initial_Mrs  Initial_Ms  Initial_Other
1               0           0              0
2               0           1              0
3               0           0              0
4               1           0              0
5               0           0              0
…             …           …              …
1304            0           1              0
1305            0           1              0
1306            0           0              0
1307            0           0              0
1308            0           0              0

[1205 rows x 16 columns]
```

[473]:
```
df_no_outliers.
  ↪drop(['sex_female','Initial_Master','Initial_Mr','Initial_Mrs','Initial_Ms','Initial_Other'
df_no_outliers
```

[473]:
```
      pclass      age  sibsp  parch      fare  survived  sex_male  embarked_C  \
1          1   0.9167      1      2  151.5500         1         1           0
2          1   2.0000      1      2  151.5500         0         0           0
3          1  30.0000      1      2  151.5500         0         1           0
4          1  25.0000      1      2  151.5500         0         0           0
5          1  48.0000      0      0   26.5500         1         1           0
…        …       …    …    …       …       …        …           …
1304       3  14.5000      1      0   14.4542         0         0           1
1305       3  22.0000      1      0   14.4542         0         0           1
```

```
1306       3  26.5000      0      0    7.2250         0         1           1
1307       3  27.0000      0      0    7.2250         0         1           1
1308       3  29.0000      0      0    7.8750         0         1           0

      embarked_Q  embarked_S
1               0           1
2               0           1
3               0           1
4               0           1
5               0           1
…             …           …
1304            0           0
1305            0           0
1306            0           0
1307            0           0
1308            0           1

[1205 rows x 10 columns]
```

[474]:
```
df_no_outliers.rename(columns={'embarked_C': 'C', 'embarked_Q':␣
↪'Q','embarked_S':'S','sex_male':'male','sibsp':'sibSp'}, inplace=True)
df_no_outliers
```

[474]:
```
      pclass      age  sibSp  parch      fare  survived  male  C  Q  S
1          1   0.9167      1      2  151.5500         1     1  0  0  1
2          1   2.0000      1      2  151.5500         0     0  0  0  1
3          1  30.0000      1      2  151.5500         0     1  0  0  1
4          1  25.0000      1      2  151.5500         0     0  0  0  1
5          1  48.0000      0      0   26.5500         1     1  0  0  1
…        …       …    …     …        …       …    … .. .. ..
1304       3  14.5000      1      0   14.4542         0     0  1  0  0
1305       3  22.0000      1      0   14.4542         0     0  1  0  0
1306       3  26.5000      0      0    7.2250         0     1  1  0  0
1307       3  27.0000      0      0    7.2250         0     1  1  0  0
1308       3  29.0000      0      0    7.8750         0     1  0  0  1

[1205 rows x 10 columns]
```

16 feature scaling

[475]:
```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

# Define the columns to be scaled
columns_to_scale = ['age', 'fare']
```

```
# Apply scaling to the selected columns only
df_no_outliers[columns_to_scale] = scaler.
 ↪fit_transform(df_no_outliers[columns_to_scale])
df_no_outliers
```

[475]:

	pclass	age	sibSp	parch	fare	survived	male	C	Q	S
1	1	0.011392	1	2	0.919227	1	1	0	0	1
2	1	0.027848	1	2	0.919227	0	0	0	0	1
3	1	0.453164	1	2	0.919227	0	1	0	0	1
4	1	0.377215	1	2	0.919227	0	0	0	0	1
5	1	0.726582	0	0	0.161039	1	1	0	0	1
…	…	…	…	…	…	…	…	..	..	..
1304	3	0.217721	1	0	0.087672	0	0	1	0	0
1305	3	0.331645	1	0	0.087672	0	0	1	0	0
1306	3	0.400000	0	0	0.043823	0	1	1	0	0
1307	3	0.407595	0	0	0.043823	0	1	1	0	0
1308	3	0.437974	0	0	0.047766	0	1	0	0	1

[1205 rows x 10 columns]

17 Test file preparations

[476]:
```
# Loading test dataset into Pandas Dataframes
test_df = pd.read_csv("test.csv")
test_df
```

[476]:

	pclass	name	sex	age	\
0	3	Kelly, Mr. James	male	34.5	
1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	
2	2	Myles, Mr. Thomas Francis	male	62.0	
3	3	Wirz, Mr. Albert	male	27.0	
4	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	
5	3	Svensson, Mr. Johan Cervin	male	14.0	
6	3	Connolly, Miss. Kate	female	30.0	
7	2	Caldwell, Mr. Albert Francis	male	26.0	
8	3	Abrahim, Mrs. Joseph (Sophie Halaut Easu)	female	18.0	
9	3	Davies, Mr. John Samuel	male	21.0	
10	3	Ilieff, Mr. Ylio	male	NaN	
11	1	Jones, Mr. Charles Cresson	male	46.0	
12	1	Snyder, Mrs. John Pillsbury (Nelle Stevenson)	female	23.0	
13	2	Howard, Mr. Benjamin	male	63.0	
14	1	Chaffee, Mrs. Herbert Fuller (Carrie Constance…	female	47.0	
15	2	del Carlo, Mrs. Sebastiano (Argenia Genovesi)	female	24.0	
16	2	Keane, Mr. Daniel	male	35.0	
17	3	Assaf, Mr. Gerios	male	21.0	

```
18       3                      Ilmakangas, Miss. Ida Livija  female  27.0

    sibSp  parch            ticket     fare cabin embarked
0       0      0            330911   7.8292   NaN        Q
1       1      0            363272   7.0000   NaN        S
2       0      0            240276   9.6875   NaN        Q
3       0      0            315154   8.6625   NaN        S
4       1      1           3101298  12.2875   NaN        S
5       0      0              7538   9.2250   NaN        S
6       0      0            330972   7.6292   NaN        Q
7       1      1            248738  29.0000   NaN        S
8       0      0              2657   7.2292   NaN        C
9       2      0         A/4 48871  24.1500   NaN        S
10      0      0            349220   7.8958   NaN        S
11      0      0               694  26.0000   NaN        S
12      1      0             21228  82.2667   B45        S
13      1      0             24065  26.0000   NaN        S
14      1      0       W.E.P. 5734  61.1750   E31        S
15      1      0     SC/PARIS 2167  27.7208   NaN        C
16      0      0            233734  12.3500   NaN        Q
17      0      0              2692   7.2250   NaN        C
18      1      0  STON/O2. 3101270   7.9250   NaN        S
```

18 preprocess the test file

[477]:
```
test_df.isnull().sum()
```

[477]:
```
pclass       0
name         0
sex          0
age          1
sibSp        0
parch        0
ticket       0
fare         0
cabin       17
embarked     0
dtype: int64
```

[478]:
```
test_df.iloc[10]
```

[478]:
```
pclass                    3
name        Ilieff, Mr. Ylio
sex                     male
age                      NaN
sibSp                      0
```

```
parch                     0
ticket               349220
fare                 7.8958
cabin                   NaN
embarked                  S
Name: 10, dtype: object
```

[479]:
```
# replace the missing value in row #10 with 33 years as he is 'Mr'
test_df['age'].fillna(33,inplace=True)
```

[480]:
```
sex = pd.get_dummies(test_df['sex'],drop_first=True)
embark = pd.get_dummies(test_df['embarked'])
```

[481]:
```
test_df = pd.concat([test_df,sex,embark],axis=1)
test_df = test_df.replace({True: 1, False: 0})
test_df
```

[481]:

	pclass	name	sex	age	sibSp	parch	ticket	fare	cabin	embarked	male	C	Q	S
0	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q	1	0	1	0
1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S	0	0	0	1
2	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q	1	0	1	0
3	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S	1	0	0	1
4	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S	0	0	0	1
5	3	Svensson, Mr. Johan Cervin	male	14.0	0	0	7538	9.2250	NaN	S	1	0	0	1
6	3	Connolly, Miss. Kate	female	30.0	0	0	330972	7.6292	NaN	Q	0	0	1	0
7	2	Caldwell, Mr. Albert Francis	male	26.0	1	1	248738	29.0000	NaN	S	1	0	0	1
8	3	Abrahim, Mrs. Joseph (Sophie Halaut Easu)	female	18.0	0	0	2657	7.2292	NaN	C	0	1	0	0
9	3	Davies, Mr. John Samuel	male	21.0	2	0	A/4 48871	24.1500	NaN	S	1	0	0	1
10	3	Ilieff, Mr. Ylio	male	33.0	0	0	349220	7.8958	NaN	S	1	0	0	1
11	1	Jones, Mr. Charles Cresson	male	46.0	0	0	694	26.0000	NaN	S	1	0	0	1
12	1	Snyder, Mrs. John Pillsbury (Nelle Stevenson)	female	23.0	1	0	21228	82.2667	B45	S	0	0	0	1
13	2	Howard, Mr. Benjamin	male	63.0	1	0	24065	26.0000	NaN	S	1	0	0	1
14	1	Chaffee, Mrs. Herbert Fuller (Carrie Constance…	female	47.0	1	0	W.E.P. 5734	61.1750	E31	S	0	0	0	1
15	2	del Carlo, Mrs. Sebastiano (Argenia Genovesi)	female	24.0	1	0	SC/PARIS 2167	27.7208	NaN	C	0	1	0	0
16	2	Keane, Mr. Daniel	male	35.0	0	0	233734	12.3500	NaN	Q	1	0	1	0
17	3	Assaf, Mr. Gerios	male	21.0	0	0	2692	7.2250	NaN	C	1	1	0	0
18	3	Ilmakangas, Miss. Ida Livija	female	27.0	1	0	STON/O2. 3101270	7.9250	NaN	S	0	0	0	1

8	0	0	2657	7.2292	NaN	C	0	1	0	0
9	2	0	A/4 48871	24.1500	NaN	S	1	0	0	1
10	0	0	349220	7.8958	NaN	S	1	0	0	1
11	0	0	694	26.0000	NaN	S	1	0	0	1
12	1	0	21228	82.2667	B45	S	0	0	0	1
13	1	0	24065	26.0000	NaN	S	1	0	0	1
14	1	0	W.E.P. 5734	61.1750	E31	S	0	0	0	1
15	1	0	SC/PARIS 2167	27.7208	NaN	C	0	1	0	0
16	0	0	233734	12.3500	NaN	Q	1	0	1	0
17	0	0	2692	7.2250	NaN	C	1	1	0	0
18	1	0	STON/O2. 3101270	7.9250	NaN	S	0	0	0	1

[482]:
```
test_df.drop(['name','ticket','cabin','sex','embarked'],axis=1,inplace=True)
test_df
```

[482]:

	pclass	age	sibSp	parch	fare	male	C	Q	S
0	3	34.5	0	0	7.8292	1	0	1	0
1	3	47.0	1	0	7.0000	0	0	0	1
2	2	62.0	0	0	9.6875	1	0	1	0
3	3	27.0	0	0	8.6625	1	0	0	1
4	3	22.0	1	1	12.2875	0	0	0	1
5	3	14.0	0	0	9.2250	1	0	0	1
6	3	30.0	0	0	7.6292	0	0	1	0
7	2	26.0	1	1	29.0000	1	0	0	1
8	3	18.0	0	0	7.2292	0	1	0	0
9	3	21.0	2	0	24.1500	1	0	0	1
10	3	33.0	0	0	7.8958	1	0	0	1
11	1	46.0	0	0	26.0000	1	0	0	1
12	1	23.0	1	0	82.2667	0	0	0	1
13	2	63.0	1	0	26.0000	1	0	0	1
14	1	47.0	1	0	61.1750	0	0	0	1
15	2	24.0	1	0	27.7208	0	1	0	0
16	2	35.0	0	0	12.3500	1	0	1	0
17	3	21.0	0	0	7.2250	1	1	0	0
18	3	27.0	1	0	7.9250	0	0	0	1

[483]:
```
# scaling
scaler = MinMaxScaler()

# Define the columns to be scaled
columns_to_scale = ['age', 'fare']

# Apply scaling to the selected columns only
test_df[columns_to_scale] = scaler.fit_transform(test_df[columns_to_scale])
test_df
```

[483]:

	pclass	age	sibSp	parch	fare	male	C	Q	S
0	3	0.418367	0	0	0.011017	1	0	1	0
1	3	0.673469	1	0	0.000000	0	0	0	1
2	2	0.979592	0	0	0.035706	1	0	1	0
3	3	0.265306	0	0	0.022088	1	0	0	1
4	3	0.163265	1	1	0.070250	0	0	0	1
5	3	0.000000	0	0	0.029562	1	0	0	1
6	3	0.326531	0	0	0.008360	0	0	1	0
7	2	0.244898	1	1	0.292294	1	0	0	1
8	3	0.081633	0	0	0.003045	0	1	0	0
9	3	0.142857	2	0	0.227856	1	0	0	1
10	3	0.387755	0	0	0.011902	1	0	0	1
11	1	0.653061	0	0	0.252436	1	0	0	1
12	1	0.183673	1	0	1.000000	0	0	0	1
13	2	1.000000	1	0	0.252436	1	0	0	1
14	1	0.673469	1	0	0.719774	0	0	0	1
15	2	0.204082	1	0	0.275298	0	1	0	0
16	2	0.428571	0	0	0.071081	1	0	1	0
17	3	0.142857	0	0	0.002989	1	1	0	0
18	3	0.265306	1	0	0.012290	0	0	0	1

19 ----------

20 Start to build our models!

21 Split the data into features (X) and target variable (y)

[484]:
```
X = df_no_outliers.drop('survived', axis=1)
y = df_no_outliers['survived']
```

22 Splitting the data into training and testing sets

[485]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3,␣
 ↪random_state = 0)
X_train.shape, X_test.shape
```

[485]: ((843, 9), (362, 9))

23 1)Naive Bayes

[486]:
```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)
```

[486]:
```
GaussianNB()
```

[487]:
```
y_pred = gnb.predict(X_test)
y_pred
```

[487]:
```
array([0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
       0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0,
       0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0,
       1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0,
       0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0,
       0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0,
       1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1,
       0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0,
       0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1,
       1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 1, 0, 0, 1, 0, 0, 0, 1], dtype=int64)
```

[488]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print('Confusion matrix\n\n', cm)
```

```
Confusion matrix

 [[188  45]
 [ 43  86]]
```

[489]:
```
TP = cm[0,0]
TN = cm[1,1]
FP = cm[0,1]
FN = cm[1,0]
print('\nTrue Positives(TP) = ', TP)
print('\nTrue Negatives(TN) = ', TN)
print('\nFalse Positives(FP) = ', FP)
print('\nFalse Negatives(FN) = ', FN)
```

```
True Positives(TP) =  188

True Negatives(TN) =  86

False Positives(FP) =  45

False Negatives(FN) =  43
```

[490]:
```
from sklearn.metrics import accuracy_score, classification_report
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
# Calculate precision, recall, and F1-score
report = classification_report(y_test, y_pred)
print("Classification Report:")
print(report)
```

```
Accuracy: 0.7569060773480663
Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.81      0.81       233
           1       0.66      0.67      0.66       129

    accuracy                           0.76       362
   macro avg       0.74      0.74      0.74       362
weighted avg       0.76      0.76      0.76       362
```

23.1 2)KNN

[491]:
```
# import KNeighbors ClaSSifier from sklearn
from sklearn.neighbors import KNeighborsClassifier


# instantiate the model
knn = KNeighborsClassifier(n_neighbors=9)


# fit the model to the training set
knn.fit(X_train, y_train)
```

[491]:
```
KNeighborsClassifier(n_neighbors=9)
```

[492]:
```
y_pred = knn.predict(X_test)

y_pred
```

[492]:
```
array([0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0,
       0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,
       0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0,
       1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0,
       0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0,
       1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1,
       0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 1, 1, 1, 0, 0, 1], dtype=int64)
```

[493]:
```
y_pred_train = knn.predict(X_train)
```

24 calculate Accuracy

[494]:
```
from sklearn.metrics import accuracy_score

print('Model accuracy score: {0:0.4f}'. format(accuracy_score(y_test, y_pred)))
```

```
Model accuracy score: 0.7956
```

25 Check for overfitting and underfitting

[495]:
```
# print the scores on training and test set

print('Training set score: {:.4f}'.format(knn.score(X_train, y_train)))

print('Test set score: {:.4f}'.format(knn.score(X_test, y_test)))
```

```
Training set score: 0.8066
Test set score: 0.7956
```

26 print confusion matrix

[496]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print('Confusion matrix\n\n', cm)
```

```
Confusion matrix

 [[199  34]
 [ 40  89]]
```

[497]:
```
TP = cm[0,0]
TN = cm[1,1]
FP = cm[0,1]
FN = cm[1,0]
print('\nTrue Positives(TP) = ', TP)
print('\nTrue Negatives(TN) = ', TN)
print('\nFalse Positives(FP) = ', FP)
print('\nFalse Negatives(FN) = ', FN)
```

```
True Positives(TP) =  199

True Negatives(TN) =  89

False Positives(FP) =  34

False Negatives(FN) =  40
```

[498]:
```
from sklearn.metrics import accuracy_score, classification_report
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
# Calculate precision, recall, and F1-score
report = classification_report(y_test, y_pred)
print("Classification Report:")
print(report)
```

```
Accuracy: 0.7955801104972375
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.85      0.84       233
           1       0.72      0.69      0.71       129

    accuracy                           0.80       362
   macro avg       0.78      0.77      0.77       362
weighted avg       0.79      0.80      0.79       362
```

[499]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# Define a range of k values to search over
k_values = range(1, 7, 1)  # values of k from 1 to 7
```

```
# Initialize lists to store accuracy scores
accuracy_scores = []

# Iterate over each k value
for k in k_values:
    # Initialize the KNN classifier with the current k value
    knn = KNeighborsClassifier(n_neighbors=k)

    # Train the KNN classifier
    knn.fit(X_train, y_train)

    # Make predictions on the test set
    y_pred = knn.predict(X_test)

    # Calculate accuracy and store it
    accuracy = accuracy_score(y_test, y_pred)
    accuracy_scores.append(accuracy)



# Find the k value with the highest accuracy
best_k = k_values[np.argmax(accuracy_scores)]
print("Best k (Elbow Method):", best_k)
```

```
Best k (Elbow Method): 4
```

27 3)SVM

[500]:
```
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
classifier = SVC(probability=True, random_state=0)
classifier.fit(X_train, y_train)
```

[500]:
```
SVC(probability=True, random_state=0)
```

[501]:
```
classifier.score(X_test, y_test)
```

[501]:
```
0.8204419889502762
```

[502]:
```
classifier = SVC(kernel='linear', probability=True, random_state=0)
classifier.fit(X_train, y_train)
classifier.score(X_test, y_test)
```

[502]:
```
0.8204419889502762
```

[503]:
```
from sklearn.model_selection import cross_validate
scores = cross_validate(classifier, X, y, cv=5)
print(scores['test_score'].mean())
```

```
0.7568464730290457
```

[504]:
```
from sklearn.model_selection import GridSearchCV

classifier= SVC(probability=True, random_state=0)

param_grid = {
    'C': [0.1, 0.5, 1, 5, 10],
    'gamma': [0.1, 0.5, 1, 5, 10],
    'kernel': ['linear', 'rbf', 'sigmoid']
}

grid_search = GridSearchCV(estimator=classifier, param_grid=param_grid, cv=5,␣
 ↪verbose=2)
grid_search.fit(X, y) # Train the model 75 times with 75 different parameter␣
 ↪combinations

best_model = grid_search.best_estimator_
```

```
Fitting 5 folds for each of 75 candidates, totalling 375 fits
[CV] END …C=0.1, gamma=0.1, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=0.1, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=0.1, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=0.1, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=0.1, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=0.1, kernel=rbf; total time=   0.4s
[CV] END …C=0.1, gamma=0.1, kernel=rbf; total time=   0.3s
[CV] END …C=0.1, gamma=0.1, kernel=rbf; total time=   0.4s
[CV] END …C=0.1, gamma=0.1, kernel=rbf; total time=   0.3s
[CV] END …C=0.1, gamma=0.1, kernel=rbf; total time=   0.3s
[CV] END …C=0.1, gamma=0.1, kernel=sigmoid; total time=   0.1s
[CV] END …C=0.1, gamma=0.1, kernel=sigmoid; total time=   0.2s
[CV] END …C=0.1, gamma=0.1, kernel=sigmoid; total time=   0.1s
[CV] END …C=0.1, gamma=0.1, kernel=sigmoid; total time=   0.1s
[CV] END …C=0.1, gamma=0.1, kernel=sigmoid; total time=   0.3s
[CV] END …C=0.1, gamma=0.5, kernel=linear; total time=   0.1s
[CV] END …C=0.1, gamma=0.5, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=0.5, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=0.5, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=0.5, kernel=linear; total time=   0.1s
[CV] END …C=0.1, gamma=0.5, kernel=rbf; total time=   0.3s
[CV] END …C=0.1, gamma=0.5, kernel=rbf; total time=   0.2s
[CV] END …C=0.1, gamma=0.5, kernel=rbf; total time=   0.7s
```

```
[CV] END …C=0.1, gamma=0.5, kernel=rbf; total time=   0.4s
[CV] END …C=0.1, gamma=0.5, kernel=rbf; total time=   0.1s
[CV] END …C=0.1, gamma=0.5, kernel=sigmoid; total time=   0.2s
[CV] END …C=0.1, gamma=0.5, kernel=sigmoid; total time=   0.2s
[CV] END …C=0.1, gamma=0.5, kernel=sigmoid; total time=   0.2s
[CV] END …C=0.1, gamma=0.5, kernel=sigmoid; total time=   0.2s
[CV] END …C=0.1, gamma=0.5, kernel=sigmoid; total time=   0.2s
[CV] END …C=0.1, gamma=1, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=1, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=1, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=1, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=1, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=1, kernel=rbf; total time=   0.3s
[CV] END …C=0.1, gamma=1, kernel=rbf; total time=   0.3s
[CV] END …C=0.1, gamma=1, kernel=rbf; total time=   0.2s
[CV] END …C=0.1, gamma=1, kernel=rbf; total time=   0.1s
[CV] END …C=0.1, gamma=1, kernel=rbf; total time=   0.1s
[CV] END …C=0.1, gamma=1, kernel=sigmoid; total time=   0.2s
[CV] END …C=0.1, gamma=1, kernel=sigmoid; total time=   0.2s
[CV] END …C=0.1, gamma=1, kernel=sigmoid; total time=   0.2s
[CV] END …C=0.1, gamma=1, kernel=sigmoid; total time=   0.2s
[CV] END …C=0.1, gamma=1, kernel=sigmoid; total time=   0.2s
[CV] END …C=0.1, gamma=5, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=5, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=5, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=5, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=5, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=5, kernel=rbf; total time=   0.2s
[CV] END …C=0.1, gamma=5, kernel=rbf; total time=   0.2s
[CV] END …C=0.1, gamma=5, kernel=rbf; total time=   0.2s
[CV] END …C=0.1, gamma=5, kernel=rbf; total time=   0.2s
[CV] END …C=0.1, gamma=5, kernel=rbf; total time=   0.2s
[CV] END …C=0.1, gamma=5, kernel=sigmoid; total time=   0.0s
[CV] END …C=0.1, gamma=5, kernel=sigmoid; total time=   0.0s
[CV] END …C=0.1, gamma=5, kernel=sigmoid; total time=   0.0s
[CV] END …C=0.1, gamma=5, kernel=sigmoid; total time=   0.0s
[CV] END …C=0.1, gamma=5, kernel=sigmoid; total time=   0.0s
[CV] END …C=0.1, gamma=10, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=10, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=10, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=10, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=10, kernel=linear; total time=   0.0s
[CV] END …C=0.1, gamma=10, kernel=rbf; total time=   0.2s
[CV] END …C=0.1, gamma=10, kernel=rbf; total time=   0.2s
[CV] END …C=0.1, gamma=10, kernel=rbf; total time=   0.2s
[CV] END …C=0.1, gamma=10, kernel=rbf; total time=   0.2s
[CV] END …C=0.1, gamma=10, kernel=rbf; total time=   0.3s
[CV] END …C=0.1, gamma=10, kernel=sigmoid; total time=   0.0s
```

```
[CV] END …C=0.1, gamma=10, kernel=sigmoid; total time=   0.0s
[CV] END …C=0.1, gamma=10, kernel=sigmoid; total time=   0.0s
[CV] END …C=0.1, gamma=10, kernel=sigmoid; total time=   0.0s
[CV] END …C=0.1, gamma=10, kernel=sigmoid; total time=   0.0s
[CV] END …C=0.5, gamma=0.1, kernel=linear; total time=   0.0s
[CV] END …C=0.5, gamma=0.1, kernel=linear; total time=   0.0s
[CV] END …C=0.5, gamma=0.1, kernel=linear; total time=   0.0s
[CV] END …C=0.5, gamma=0.1, kernel=linear; total time=   0.0s
[CV] END …C=0.5, gamma=0.1, kernel=linear; total time=   0.0s
[CV] END …C=0.5, gamma=0.1, kernel=rbf; total time=   0.1s
[CV] END …C=0.5, gamma=0.1, kernel=rbf; total time=   0.1s
[CV] END …C=0.5, gamma=0.1, kernel=rbf; total time=   0.1s
[CV] END …C=0.5, gamma=0.1, kernel=rbf; total time=   0.1s
[CV] END …C=0.5, gamma=0.1, kernel=rbf; total time=   0.1s
[CV] END …C=0.5, gamma=0.1, kernel=sigmoid; total time=   0.1s
[CV] END …C=0.5, gamma=0.1, kernel=sigmoid; total time=   0.1s
[CV] END …C=0.5, gamma=0.1, kernel=sigmoid; total time=   0.1s
[CV] END …C=0.5, gamma=0.1, kernel=sigmoid; total time=   0.0s
[CV] END …C=0.5, gamma=0.1, kernel=sigmoid; total time=   0.1s
[CV] END …C=0.5, gamma=0.5, kernel=linear; total time=   0.1s
[CV] END …C=0.5, gamma=0.5, kernel=linear; total time=   0.0s
[CV] END …C=0.5, gamma=0.5, kernel=linear; total time=   0.0s
[CV] END …C=0.5, gamma=0.5, kernel=linear; total time=   0.0s
[CV] END …C=0.5, gamma=0.5, kernel=linear; total time=   0.0s
[CV] END …C=0.5, gamma=0.5, kernel=rbf; total time=   0.2s
[CV] END …C=0.5, gamma=0.5, kernel=rbf; total time=   0.2s
[CV] END …C=0.5, gamma=0.5, kernel=rbf; total time=   0.1s
[CV] END …C=0.5, gamma=0.5, kernel=rbf; total time=   0.1s
[CV] END …C=0.5, gamma=0.5, kernel=rbf; total time=   0.1s
[CV] END …C=0.5, gamma=0.5, kernel=sigmoid; total time=   0.2s
[CV] END …C=0.5, gamma=0.5, kernel=sigmoid; total time=   0.2s
[CV] END …C=0.5, gamma=0.5, kernel=sigmoid; total time=   0.1s
[CV] END …C=0.5, gamma=0.5, kernel=sigmoid; total time=   0.2s
[CV] END …C=0.5, gamma=0.5, kernel=sigmoid; total time=   0.2s
[CV] END …C=0.5, gamma=1, kernel=linear; total time=   0.0s
[CV] END …C=0.5, gamma=1, kernel=linear; total time=   0.0s
[CV] END …C=0.5, gamma=1, kernel=linear; total time=   0.0s
[CV] END …C=0.5, gamma=1, kernel=linear; total time=   0.0s
[CV] END …C=0.5, gamma=1, kernel=linear; total time=   0.0s
[CV] END …C=0.5, gamma=1, kernel=rbf; total time=   0.1s
[CV] END …C=0.5, gamma=1, kernel=rbf; total time=   0.1s
[CV] END …C=0.5, gamma=1, kernel=rbf; total time=   0.2s
[CV] END …C=0.5, gamma=1, kernel=rbf; total time=   0.1s
[CV] END …C=0.5, gamma=1, kernel=rbf; total time=   0.1s
[CV] END …C=0.5, gamma=1, kernel=sigmoid; total time=   0.1s
[CV] END …C=0.5, gamma=1, kernel=sigmoid; total time=   0.2s
[CV] END …C=0.5, gamma=1, kernel=sigmoid; total time=   0.2s
[CV] END …C=0.5, gamma=1, kernel=sigmoid; total time=   0.3s
```

```
[CV] END …C=0.5, gamma=1, kernel=sigmoid; total time=   0.2s
[CV] END …C=0.5, gamma=5, kernel=linear; total time=   0.0s
[CV] END …C=0.5, gamma=5, kernel=linear; total time=   0.0s
[CV] END …C=0.5, gamma=5, kernel=linear; total time=   0.0s
[CV] END …C=0.5, gamma=5, kernel=linear; total time=   0.0s
[CV] END …C=0.5, gamma=5, kernel=linear; total time=   0.0s
[CV] END …C=0.5, gamma=5, kernel=rbf; total time=   0.2s
[CV] END …C=0.5, gamma=5, kernel=rbf; total time=   0.2s
[CV] END …C=0.5, gamma=5, kernel=rbf; total time=   0.2s
[CV] END …C=0.5, gamma=5, kernel=rbf; total time=   0.3s
[CV] END …C=0.5, gamma=5, kernel=rbf; total time=   0.4s
[CV] END …C=0.5, gamma=5, kernel=sigmoid; total time=   0.0s
[CV] END …C=0.5, gamma=5, kernel=sigmoid; total time=   0.0s
[CV] END …C=0.5, gamma=5, kernel=sigmoid; total time=   0.0s
[CV] END …C=0.5, gamma=5, kernel=sigmoid; total time=   0.0s
[CV] END …C=0.5, gamma=5, kernel=sigmoid; total time=   0.1s
[CV] END …C=0.5, gamma=10, kernel=linear; total time=   0.0s
[CV] END …C=0.5, gamma=10, kernel=linear; total time=   0.0s
[CV] END …C=0.5, gamma=10, kernel=linear; total time=   0.0s
[CV] END …C=0.5, gamma=10, kernel=linear; total time=   0.0s
[CV] END …C=0.5, gamma=10, kernel=linear; total time=   0.0s
[CV] END …C=0.5, gamma=10, kernel=rbf; total time=   0.2s
[CV] END …C=0.5, gamma=10, kernel=rbf; total time=   0.2s
[CV] END …C=0.5, gamma=10, kernel=rbf; total time=   0.2s
[CV] END …C=0.5, gamma=10, kernel=rbf; total time=   0.2s
[CV] END …C=0.5, gamma=10, kernel=rbf; total time=   0.2s
[CV] END …C=0.5, gamma=10, kernel=sigmoid; total time=   0.0s
[CV] END …C=0.5, gamma=10, kernel=sigmoid; total time=   0.1s
[CV] END …C=0.5, gamma=10, kernel=sigmoid; total time=   0.1s
[CV] END …C=0.5, gamma=10, kernel=sigmoid; total time=   0.2s
[CV] END …C=0.5, gamma=10, kernel=sigmoid; total time=   0.3s
[CV] END …C=1, gamma=0.1, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=0.1, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=0.1, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=0.1, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=0.1, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=0.1, kernel=rbf; total time=   0.1s
[CV] END …C=1, gamma=0.1, kernel=rbf; total time=   0.1s
[CV] END …C=1, gamma=0.1, kernel=rbf; total time=   0.1s
[CV] END …C=1, gamma=0.1, kernel=rbf; total time=   0.1s
[CV] END …C=1, gamma=0.1, kernel=rbf; total time=   0.1s
[CV] END …C=1, gamma=0.1, kernel=sigmoid; total time=   0.1s
[CV] END …C=1, gamma=0.1, kernel=sigmoid; total time=   0.2s
[CV] END …C=1, gamma=0.1, kernel=sigmoid; total time=   0.1s
[CV] END …C=1, gamma=0.1, kernel=sigmoid; total time=   0.0s
[CV] END …C=1, gamma=0.1, kernel=sigmoid; total time=   0.1s
[CV] END …C=1, gamma=0.5, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=0.5, kernel=linear; total time=   0.0s
```

```
[CV] END …C=1, gamma=0.5, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=0.5, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=0.5, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=0.5, kernel=rbf; total time=   0.1s
[CV] END …C=1, gamma=0.5, kernel=rbf; total time=   0.1s
[CV] END …C=1, gamma=0.5, kernel=rbf; total time=   0.1s
[CV] END …C=1, gamma=0.5, kernel=rbf; total time=   0.1s
[CV] END …C=1, gamma=0.5, kernel=rbf; total time=   0.1s
[CV] END …C=1, gamma=0.5, kernel=sigmoid; total time=   0.2s
[CV] END …C=1, gamma=0.5, kernel=sigmoid; total time=   0.1s
[CV] END …C=1, gamma=0.5, kernel=sigmoid; total time=   0.1s
[CV] END …C=1, gamma=0.5, kernel=sigmoid; total time=   0.1s
[CV] END …C=1, gamma=0.5, kernel=sigmoid; total time=   0.1s
[CV] END …C=1, gamma=1, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=1, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=1, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=1, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=1, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=1, kernel=rbf; total time=   0.1s
[CV] END …C=1, gamma=1, kernel=rbf; total time=   0.1s
[CV] END …C=1, gamma=1, kernel=rbf; total time=   0.2s
[CV] END …C=1, gamma=1, kernel=rbf; total time=   0.1s
[CV] END …C=1, gamma=1, kernel=rbf; total time=   0.1s
[CV] END …C=1, gamma=1, kernel=sigmoid; total time=   0.1s
[CV] END …C=1, gamma=1, kernel=sigmoid; total time=   0.1s
[CV] END …C=1, gamma=1, kernel=sigmoid; total time=   0.1s
[CV] END …C=1, gamma=1, kernel=sigmoid; total time=   0.1s
[CV] END …C=1, gamma=1, kernel=sigmoid; total time=   0.2s
[CV] END …C=1, gamma=5, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=5, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=5, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=5, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=5, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=5, kernel=rbf; total time=   0.1s
[CV] END …C=1, gamma=5, kernel=rbf; total time=   0.1s
[CV] END …C=1, gamma=5, kernel=rbf; total time=   0.1s
[CV] END …C=1, gamma=5, kernel=rbf; total time=   0.1s
[CV] END …C=1, gamma=5, kernel=rbf; total time=   0.1s
[CV] END …C=1, gamma=5, kernel=sigmoid; total time=   0.0s
[CV] END …C=1, gamma=5, kernel=sigmoid; total time=   0.0s
[CV] END …C=1, gamma=5, kernel=sigmoid; total time=   0.1s
[CV] END …C=1, gamma=5, kernel=sigmoid; total time=   0.1s
[CV] END …C=1, gamma=5, kernel=sigmoid; total time=   0.1s
[CV] END …C=1, gamma=10, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=10, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=10, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=10, kernel=linear; total time=   0.0s
[CV] END …C=1, gamma=10, kernel=linear; total time=   0.0s
```

```
[CV] END …C=1, gamma=10, kernel=rbf; total time=   0.3s
[CV] END …C=1, gamma=10, kernel=rbf; total time=   0.2s
[CV] END …C=1, gamma=10, kernel=rbf; total time=   0.3s
[CV] END …C=1, gamma=10, kernel=rbf; total time=   0.1s
[CV] END …C=1, gamma=10, kernel=rbf; total time=   0.2s
[CV] END …C=1, gamma=10, kernel=sigmoid; total time=   0.0s
[CV] END …C=1, gamma=10, kernel=sigmoid; total time=   0.1s
[CV] END …C=1, gamma=10, kernel=sigmoid; total time=   0.0s
[CV] END …C=1, gamma=10, kernel=sigmoid; total time=   0.0s
[CV] END …C=1, gamma=10, kernel=sigmoid; total time=   0.1s
[CV] END …C=5, gamma=0.1, kernel=linear; total time=   0.2s
[CV] END …C=5, gamma=0.1, kernel=linear; total time=   0.1s
[CV] END …C=5, gamma=0.1, kernel=linear; total time=   0.1s
[CV] END …C=5, gamma=0.1, kernel=linear; total time=   0.0s
[CV] END …C=5, gamma=0.1, kernel=linear; total time=   0.0s
[CV] END …C=5, gamma=0.1, kernel=rbf; total time=   0.2s
[CV] END …C=5, gamma=0.1, kernel=rbf; total time=   0.4s
[CV] END …C=5, gamma=0.1, kernel=rbf; total time=   0.2s
[CV] END …C=5, gamma=0.1, kernel=rbf; total time=   0.3s
[CV] END …C=5, gamma=0.1, kernel=rbf; total time=   0.1s
[CV] END …C=5, gamma=0.1, kernel=sigmoid; total time=   0.0s
[CV] END …C=5, gamma=0.1, kernel=sigmoid; total time=   0.0s
[CV] END …C=5, gamma=0.1, kernel=sigmoid; total time=   0.1s
[CV] END …C=5, gamma=0.1, kernel=sigmoid; total time=   0.0s
[CV] END …C=5, gamma=0.1, kernel=sigmoid; total time=   0.0s
[CV] END …C=5, gamma=0.5, kernel=linear; total time=   0.1s
[CV] END …C=5, gamma=0.5, kernel=linear; total time=   0.0s
[CV] END …C=5, gamma=0.5, kernel=linear; total time=   0.0s
[CV] END …C=5, gamma=0.5, kernel=linear; total time=   0.0s
[CV] END …C=5, gamma=0.5, kernel=linear; total time=   0.0s
[CV] END …C=5, gamma=0.5, kernel=rbf; total time=   0.2s
[CV] END …C=5, gamma=0.5, kernel=rbf; total time=   0.2s
[CV] END …C=5, gamma=0.5, kernel=rbf; total time=   0.4s
[CV] END …C=5, gamma=0.5, kernel=rbf; total time=   0.2s
[CV] END …C=5, gamma=0.5, kernel=rbf; total time=   0.1s
[CV] END …C=5, gamma=0.5, kernel=sigmoid; total time=   0.2s
[CV] END …C=5, gamma=0.5, kernel=sigmoid; total time=   0.1s
[CV] END …C=5, gamma=0.5, kernel=sigmoid; total time=   0.1s
[CV] END …C=5, gamma=0.5, kernel=sigmoid; total time=   0.1s
[CV] END …C=5, gamma=0.5, kernel=sigmoid; total time=   0.1s
[CV] END …C=5, gamma=1, kernel=linear; total time=   0.1s
[CV] END …C=5, gamma=1, kernel=linear; total time=   0.1s
[CV] END …C=5, gamma=1, kernel=linear; total time=   0.1s
[CV] END …C=5, gamma=1, kernel=linear; total time=   0.0s
[CV] END …C=5, gamma=1, kernel=linear; total time=   0.0s
[CV] END …C=5, gamma=1, kernel=rbf; total time=   0.1s
[CV] END …C=5, gamma=1, kernel=rbf; total time=   0.2s
[CV] END …C=5, gamma=1, kernel=rbf; total time=   0.3s
```

```
[CV] END …C=5, gamma=1, kernel=rbf; total time=   0.1s
[CV] END …C=5, gamma=1, kernel=rbf; total time=   0.2s
[CV] END …C=5, gamma=1, kernel=sigmoid; total time=   0.2s
[CV] END …C=5, gamma=1, kernel=sigmoid; total time=   0.1s
[CV] END …C=5, gamma=1, kernel=sigmoid; total time=   0.1s
[CV] END …C=5, gamma=1, kernel=sigmoid; total time=   0.3s
[CV] END …C=5, gamma=1, kernel=sigmoid; total time=   0.2s
[CV] END …C=5, gamma=5, kernel=linear; total time=   0.3s
[CV] END …C=5, gamma=5, kernel=linear; total time=   0.1s
[CV] END …C=5, gamma=5, kernel=linear; total time=   0.0s
[CV] END …C=5, gamma=5, kernel=linear; total time=   0.2s
[CV] END …C=5, gamma=5, kernel=linear; total time=   0.2s
[CV] END …C=5, gamma=5, kernel=rbf; total time=   0.2s
[CV] END …C=5, gamma=5, kernel=rbf; total time=   0.3s
[CV] END …C=5, gamma=5, kernel=rbf; total time=   0.3s
[CV] END …C=5, gamma=5, kernel=rbf; total time=   0.2s
[CV] END …C=5, gamma=5, kernel=rbf; total time=   0.2s
[CV] END …C=5, gamma=5, kernel=sigmoid; total time=   0.0s
[CV] END …C=5, gamma=5, kernel=sigmoid; total time=   0.0s
[CV] END …C=5, gamma=5, kernel=sigmoid; total time=   0.0s
[CV] END …C=5, gamma=5, kernel=sigmoid; total time=   0.1s
[CV] END …C=5, gamma=5, kernel=sigmoid; total time=   0.1s
[CV] END …C=5, gamma=10, kernel=linear; total time=   0.1s
[CV] END …C=5, gamma=10, kernel=linear; total time=   0.0s
[CV] END …C=5, gamma=10, kernel=linear; total time=   0.0s
[CV] END …C=5, gamma=10, kernel=linear; total time=   0.0s
[CV] END …C=5, gamma=10, kernel=linear; total time=   0.0s
[CV] END …C=5, gamma=10, kernel=rbf; total time=   0.3s
[CV] END …C=5, gamma=10, kernel=rbf; total time=   0.3s
[CV] END …C=5, gamma=10, kernel=rbf; total time=   0.2s
[CV] END …C=5, gamma=10, kernel=rbf; total time=   0.2s
[CV] END …C=5, gamma=10, kernel=rbf; total time=   0.2s
[CV] END …C=5, gamma=10, kernel=sigmoid; total time=   0.0s
[CV] END …C=5, gamma=10, kernel=sigmoid; total time=   0.0s
[CV] END …C=5, gamma=10, kernel=sigmoid; total time=   0.0s
[CV] END …C=5, gamma=10, kernel=sigmoid; total time=   0.0s
[CV] END …C=5, gamma=10, kernel=sigmoid; total time=   0.0s
[CV] END …C=10, gamma=0.1, kernel=linear; total time=   0.1s
[CV] END …C=10, gamma=0.1, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=0.1, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=0.1, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=0.1, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=0.1, kernel=rbf; total time=   0.2s
[CV] END …C=10, gamma=0.1, kernel=rbf; total time=   0.1s
[CV] END …C=10, gamma=0.1, kernel=rbf; total time=   0.1s
[CV] END …C=10, gamma=0.1, kernel=rbf; total time=   0.1s
[CV] END …C=10, gamma=0.1, kernel=rbf; total time=   0.1s
[CV] END …C=10, gamma=0.1, kernel=sigmoid; total time=   0.0s
```

```
[CV] END …C=10, gamma=0.1, kernel=sigmoid; total time=   0.0s
[CV] END …C=10, gamma=0.1, kernel=sigmoid; total time=   0.0s
[CV] END …C=10, gamma=0.1, kernel=sigmoid; total time=   0.0s
[CV] END …C=10, gamma=0.1, kernel=sigmoid; total time=   0.0s
[CV] END …C=10, gamma=0.5, kernel=linear; total time=   0.1s
[CV] END …C=10, gamma=0.5, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=0.5, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=0.5, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=0.5, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=0.5, kernel=rbf; total time=   0.1s
[CV] END …C=10, gamma=0.5, kernel=rbf; total time=   0.1s
[CV] END …C=10, gamma=0.5, kernel=rbf; total time=   0.1s
[CV] END …C=10, gamma=0.5, kernel=rbf; total time=   0.1s
[CV] END …C=10, gamma=0.5, kernel=rbf; total time=   0.1s
[CV] END …C=10, gamma=0.5, kernel=sigmoid; total time=   0.1s
[CV] END …C=10, gamma=0.5, kernel=sigmoid; total time=   0.1s
[CV] END …C=10, gamma=0.5, kernel=sigmoid; total time=   0.1s
[CV] END …C=10, gamma=0.5, kernel=sigmoid; total time=   0.1s
[CV] END …C=10, gamma=0.5, kernel=sigmoid; total time=   0.2s
[CV] END …C=10, gamma=1, kernel=linear; total time=   0.1s
[CV] END …C=10, gamma=1, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=1, kernel=linear; total time=   0.1s
[CV] END …C=10, gamma=1, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=1, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=1, kernel=rbf; total time=   0.2s
[CV] END …C=10, gamma=1, kernel=rbf; total time=   0.2s
[CV] END …C=10, gamma=1, kernel=rbf; total time=   0.2s
[CV] END …C=10, gamma=1, kernel=rbf; total time=   0.2s
[CV] END …C=10, gamma=1, kernel=rbf; total time=   0.2s
[CV] END …C=10, gamma=1, kernel=sigmoid; total time=   0.2s
[CV] END …C=10, gamma=1, kernel=sigmoid; total time=   0.1s
[CV] END …C=10, gamma=1, kernel=sigmoid; total time=   0.1s
[CV] END …C=10, gamma=1, kernel=sigmoid; total time=   0.1s
[CV] END …C=10, gamma=1, kernel=sigmoid; total time=   0.1s
[CV] END …C=10, gamma=5, kernel=linear; total time=   0.1s
[CV] END …C=10, gamma=5, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=5, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=5, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=5, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=5, kernel=rbf; total time=   0.2s
[CV] END …C=10, gamma=5, kernel=rbf; total time=   0.2s
[CV] END …C=10, gamma=5, kernel=rbf; total time=   0.2s
[CV] END …C=10, gamma=5, kernel=rbf; total time=   0.2s
[CV] END …C=10, gamma=5, kernel=rbf; total time=   0.2s
[CV] END …C=10, gamma=5, kernel=sigmoid; total time=   0.0s
[CV] END …C=10, gamma=5, kernel=sigmoid; total time=   0.0s
[CV] END …C=10, gamma=5, kernel=sigmoid; total time=   0.1s
[CV] END …C=10, gamma=5, kernel=sigmoid; total time=   0.0s
```

```
[CV] END …C=10, gamma=5, kernel=sigmoid; total time=   0.0s
[CV] END …C=10, gamma=10, kernel=linear; total time=   0.1s
[CV] END …C=10, gamma=10, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=10, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=10, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=10, kernel=linear; total time=   0.0s
[CV] END …C=10, gamma=10, kernel=rbf; total time=   0.2s
[CV] END …C=10, gamma=10, kernel=rbf; total time=   0.2s
[CV] END …C=10, gamma=10, kernel=rbf; total time=   0.2s
[CV] END …C=10, gamma=10, kernel=rbf; total time=   0.3s
[CV] END …C=10, gamma=10, kernel=rbf; total time=   0.2s
[CV] END …C=10, gamma=10, kernel=sigmoid; total time=   0.0s
[CV] END …C=10, gamma=10, kernel=sigmoid; total time=   0.0s
[CV] END …C=10, gamma=10, kernel=sigmoid; total time=   0.0s
[CV] END …C=10, gamma=10, kernel=sigmoid; total time=   0.0s
[CV] END …C=10, gamma=10, kernel=sigmoid; total time=   0.0s
```

[505]:
```
print(grid_search.best_params_)
```

```
{'C': 0.1, 'gamma': 0.1, 'kernel': 'linear'}
```

[506]:
```
scores = cross_validate(best_model, X, y, cv=5)
print(scores['test_score'].mean())
```

```
0.7825726141078839
```

[507]:
```
model = SVC(kernel='linear', probability=True, random_state=0)

param_grid = {
    'C': [0.1,0.2, 0.5, 1, 1.5],
    'gamma': [0.001, 0.009, 0.098, 0.01, 0.1]
}

grid_search = GridSearchCV(estimator=classifier, param_grid=param_grid, cv=5,␣
 ↪verbose=2)
grid_search.fit(X, y)

best_model = grid_search.best_estimator_
```

```
Fitting 5 folds for each of 25 candidates, totalling 125 fits
[CV] END …C=0.1, gamma=0.001; total time=   0.2s
[CV] END …C=0.1, gamma=0.001; total time=   0.2s
[CV] END …C=0.1, gamma=0.001; total time=   0.2s
[CV] END …C=0.1, gamma=0.001; total time=   0.2s
[CV] END …C=0.1, gamma=0.001; total time=   0.2s
[CV] END …C=0.1, gamma=0.009; total time=   0.2s
[CV] END …C=0.1, gamma=0.009; total time=   0.3s
[CV] END …C=0.1, gamma=0.009; total time=   0.2s
```

```
[CV] END …C=0.1, gamma=0.009; total time=   0.2s
[CV] END …C=0.1, gamma=0.009; total time=   0.2s
[CV] END …C=0.1, gamma=0.098; total time=   0.2s
[CV] END …C=0.1, gamma=0.098; total time=   0.2s
[CV] END …C=0.1, gamma=0.098; total time=   0.2s
[CV] END …C=0.1, gamma=0.098; total time=   0.2s
[CV] END …C=0.1, gamma=0.098; total time=   0.2s
[CV] END …C=0.1, gamma=0.01; total time=   0.2s
[CV] END …C=0.1, gamma=0.01; total time=   0.2s
[CV] END …C=0.1, gamma=0.01; total time=   0.2s
[CV] END …C=0.1, gamma=0.01; total time=   0.2s
[CV] END …C=0.1, gamma=0.01; total time=   0.2s
[CV] END …C=0.1, gamma=0.1; total time=   0.1s
[CV] END …C=0.1, gamma=0.1; total time=   0.2s
[CV] END …C=0.1, gamma=0.1; total time=   0.2s
[CV] END …C=0.1, gamma=0.1; total time=   0.2s
[CV] END …C=0.1, gamma=0.1; total time=   0.2s
[CV] END …C=0.2, gamma=0.001; total time=   0.3s
[CV] END …C=0.2, gamma=0.001; total time=   0.2s
[CV] END …C=0.2, gamma=0.001; total time=   0.6s
[CV] END …C=0.2, gamma=0.001; total time=   0.2s
[CV] END …C=0.2, gamma=0.001; total time=   0.2s
[CV] END …C=0.2, gamma=0.009; total time=   0.2s
[CV] END …C=0.2, gamma=0.009; total time=   0.2s
[CV] END …C=0.2, gamma=0.009; total time=   0.2s
[CV] END …C=0.2, gamma=0.009; total time=   0.2s
[CV] END …C=0.2, gamma=0.009; total time=   0.2s
[CV] END …C=0.2, gamma=0.098; total time=   0.1s
[CV] END …C=0.2, gamma=0.098; total time=   0.1s
[CV] END …C=0.2, gamma=0.098; total time=   0.1s
[CV] END …C=0.2, gamma=0.098; total time=   0.1s
[CV] END …C=0.2, gamma=0.098; total time=   0.1s
[CV] END …C=0.2, gamma=0.01; total time=   0.2s
[CV] END …C=0.2, gamma=0.01; total time=   0.2s
[CV] END …C=0.2, gamma=0.01; total time=   0.2s
[CV] END …C=0.2, gamma=0.01; total time=   0.2s
[CV] END …C=0.2, gamma=0.01; total time=   0.2s
[CV] END …C=0.2, gamma=0.1; total time=   0.1s
[CV] END …C=0.2, gamma=0.1; total time=   0.2s
[CV] END …C=0.2, gamma=0.1; total time=   0.1s
[CV] END …C=0.2, gamma=0.1; total time=   0.1s
[CV] END …C=0.2, gamma=0.1; total time=   0.1s
[CV] END …C=0.5, gamma=0.001; total time=   0.2s
[CV] END …C=0.5, gamma=0.001; total time=   0.2s
[CV] END …C=0.5, gamma=0.001; total time=   0.2s
[CV] END …C=0.5, gamma=0.001; total time=   0.2s
[CV] END …C=0.5, gamma=0.001; total time=   0.2s
[CV] END …C=0.5, gamma=0.009; total time=   0.2s
```

```
[CV] END …C=0.5, gamma=0.009; total time=   0.2s
[CV] END …C=0.5, gamma=0.009; total time=   0.2s
[CV] END …C=0.5, gamma=0.009; total time=   0.2s
[CV] END …C=0.5, gamma=0.009; total time=   0.2s
[CV] END …C=0.5, gamma=0.098; total time=   0.1s
[CV] END …C=0.5, gamma=0.098; total time=   0.1s
[CV] END …C=0.5, gamma=0.098; total time=   0.1s
[CV] END …C=0.5, gamma=0.098; total time=   0.1s
[CV] END …C=0.5, gamma=0.098; total time=   0.1s
[CV] END …C=0.5, gamma=0.01; total time=   0.2s
[CV] END …C=0.5, gamma=0.01; total time=   0.2s
[CV] END …C=0.5, gamma=0.01; total time=   0.2s
[CV] END …C=0.5, gamma=0.01; total time=   0.2s
[CV] END …C=0.5, gamma=0.01; total time=   0.2s
[CV] END …C=0.5, gamma=0.1; total time=   0.1s
[CV] END …C=0.5, gamma=0.1; total time=   0.2s
[CV] END …C=0.5, gamma=0.1; total time=   0.2s
[CV] END …C=0.5, gamma=0.1; total time=   0.1s
[CV] END …C=0.5, gamma=0.1; total time=   0.1s
[CV] END …C=1, gamma=0.001; total time=   0.2s
[CV] END …C=1, gamma=0.001; total time=   0.3s
[CV] END …C=1, gamma=0.001; total time=   0.3s
[CV] END …C=1, gamma=0.001; total time=   0.2s
[CV] END …C=1, gamma=0.001; total time=   0.2s
[CV] END …C=1, gamma=0.009; total time=   0.2s
[CV] END …C=1, gamma=0.009; total time=   0.2s
[CV] END …C=1, gamma=0.009; total time=   0.2s
[CV] END …C=1, gamma=0.009; total time=   0.1s
[CV] END …C=1, gamma=0.009; total time=   0.2s
[CV] END …C=1, gamma=0.098; total time=   0.1s
[CV] END …C=1, gamma=0.098; total time=   0.1s
[CV] END …C=1, gamma=0.098; total time=   0.1s
[CV] END …C=1, gamma=0.098; total time=   0.1s
[CV] END …C=1, gamma=0.098; total time=   0.2s
[CV] END …C=1, gamma=0.01; total time=   0.3s
[CV] END …C=1, gamma=0.01; total time=   0.2s
[CV] END …C=1, gamma=0.01; total time=   0.2s
[CV] END …C=1, gamma=0.01; total time=   0.2s
[CV] END …C=1, gamma=0.01; total time=   0.1s
[CV] END …C=1, gamma=0.1; total time=   0.1s
[CV] END …C=1, gamma=0.1; total time=   0.1s
[CV] END …C=1, gamma=0.1; total time=   0.1s
[CV] END …C=1, gamma=0.1; total time=   0.1s
[CV] END …C=1, gamma=0.1; total time=   0.1s
[CV] END …C=1.5, gamma=0.001; total time=   0.3s
[CV] END …C=1.5, gamma=0.001; total time=   0.2s
[CV] END …C=1.5, gamma=0.001; total time=   0.2s
[CV] END …C=1.5, gamma=0.001; total time=   0.2s
```

```
[CV] END …C=1.5, gamma=0.001; total time=   0.2s
[CV] END …C=1.5, gamma=0.009; total time=   0.2s
[CV] END …C=1.5, gamma=0.009; total time=   0.2s
[CV] END …C=1.5, gamma=0.009; total time=   0.2s
[CV] END …C=1.5, gamma=0.009; total time=   0.2s
[CV] END …C=1.5, gamma=0.009; total time=   0.1s
[CV] END …C=1.5, gamma=0.098; total time=   0.1s
[CV] END …C=1.5, gamma=0.098; total time=   0.1s
[CV] END …C=1.5, gamma=0.098; total time=   0.1s
[CV] END …C=1.5, gamma=0.098; total time=   0.1s
[CV] END …C=1.5, gamma=0.098; total time=   0.1s
[CV] END …C=1.5, gamma=0.01; total time=   0.1s
[CV] END …C=1.5, gamma=0.01; total time=   0.2s
[CV] END …C=1.5, gamma=0.01; total time=   0.2s
[CV] END …C=1.5, gamma=0.01; total time=   0.2s
[CV] END …C=1.5, gamma=0.01; total time=   0.1s
[CV] END …C=1.5, gamma=0.1; total time=   0.1s
[CV] END …C=1.5, gamma=0.1; total time=   0.2s
[CV] END …C=1.5, gamma=0.1; total time=   0.1s
[CV] END …C=1.5, gamma=0.1; total time=   0.1s
[CV] END …C=1.5, gamma=0.1; total time=   0.1s
```

[508]:
```
print(grid_search.best_params_)
```

```
{'C': 1.5, 'gamma': 0.009}
```

[509]:
```
scores = cross_validate(best_model, X, y, cv=5)
print(scores['test_score'].mean())
```

```
0.7850622406639005
```

[510]:
```
classifier = best_model
```

[511]:
```
from sklearn.metrics import classification_report

print(classification_report(y_test, classifier.predict(X_test)))
```

```
              precision    recall  f1-score   support

           0       0.85      0.87      0.86       233
           1       0.76      0.73      0.74       129

    accuracy                           0.82       362
   macro avg       0.81      0.80      0.80       362
weighted avg       0.82      0.82      0.82       362
```

28 (ANN)

[512]:
```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.metrics import accuracy_score, confusion_matrix,␣
 ↪classification_report
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[512], line 1
----> 1 import tensorflow as tf
      2 from tensorflow.keras.models import Sequential
      3 from tensorflow.keras.layers import Dense

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.
 ↪11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-packages\tensorflow\__init__.
 ↪py:37
     34 import site as _site
     35 import sys as _sys
---> 37 from tensorflow.python.tools import module_util as _module_util
     38 from tensorflow.python.util.lazy_loader import KerasLazyLoader as␣
 ↪_KerasLazyLoader
     40 # Make sure code inside the TensorFlow codebase can use tf2.enabled() at␣
 ↪import.

ModuleNotFoundError: No module named 'tensorflow.python'
```

[ ]:
```
model = tf.keras.models.Sequential()
model.add(Dense(32, activation = 'relu', input_shape = (9, )))
model.add(Dense(32,activation='relu'))
model.add(Dense(16,activation='relu'))
model.add(Dense(8,activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

28.1 Training the ANN

[ ]:
```
from keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import Adam
adam = Adam(0.1)
# Define the EarlyStopping callback to prevent overfitting.
early_stopping_monitor = EarlyStopping(min_delta=0.001, patience=10,␣
 ↪monitor='val_loss', restore_best_weights=True)
```

[ ]:
```
model.compile(optimizer='adam', loss = 'binary_crossentropy',metrics =␣
 ↪['accuracy'])
```

[ ]:
```
# Training the ANN on the Training set
X_train = np.asarray(X_train, dtype=np.float32)
Y_train = np.asarray(y_train, dtype=np.float32)
test_file = np.asarray(test_df, dtype=np.float32)
history = model.fit(x = X_train, y = Y_train,
                    validation_data = (X_test, y_test),
                    callbacks=[early_stopping_monitor],
                    epochs=100)
```

```
Epoch 1/100
27/27             0s 4ms/step -
accuracy: 0.7684 - loss: 0.4825 - val_accuracy: 0.8039 - val_loss: 0.4324
Epoch 2/100
27/27             0s 3ms/step -
accuracy: 0.8085 - loss: 0.4244 - val_accuracy: 0.8039 - val_loss: 0.4416
Epoch 3/100
27/27             0s 7ms/step -
accuracy: 0.7844 - loss: 0.4520 - val_accuracy: 0.8066 - val_loss: 0.4344
Epoch 4/100
27/27             0s 3ms/step -
accuracy: 0.7935 - loss: 0.4568 - val_accuracy: 0.8039 - val_loss: 0.4397
Epoch 5/100
27/27             0s 3ms/step -
accuracy: 0.7946 - loss: 0.4502 - val_accuracy: 0.7901 - val_loss: 0.4539
Epoch 6/100
27/27             0s 3ms/step -
accuracy: 0.7901 - loss: 0.4575 - val_accuracy: 0.8066 - val_loss: 0.4361
Epoch 7/100
27/27             0s 4ms/step -
accuracy: 0.7707 - loss: 0.4751 - val_accuracy: 0.8094 - val_loss: 0.4341
Epoch 8/100
27/27             0s 4ms/step -
accuracy: 0.7886 - loss: 0.4555 - val_accuracy: 0.8177 - val_loss: 0.4306
Epoch 9/100
27/27             0s 4ms/step -
accuracy: 0.7800 - loss: 0.4501 - val_accuracy: 0.8232 - val_loss: 0.4325
Epoch 10/100
27/27             0s 4ms/step -
accuracy: 0.7822 - loss: 0.4833 - val_accuracy: 0.8204 - val_loss: 0.4360
Epoch 11/100
27/27             0s 6ms/step -
accuracy: 0.7597 - loss: 0.4826 - val_accuracy: 0.8122 - val_loss: 0.4312
Epoch 12/100
27/27             0s 3ms/step -
accuracy: 0.7865 - loss: 0.4573 - val_accuracy: 0.8066 - val_loss: 0.4329
Epoch 13/100
27/27             0s 3ms/step -
```

```
accuracy: 0.8029 - loss: 0.4475 - val_accuracy: 0.8232 - val_loss: 0.4289
Epoch 14/100
27/27             0s 3ms/step -
accuracy: 0.8123 - loss: 0.4361 - val_accuracy: 0.7901 - val_loss: 0.4410
Epoch 15/100
27/27             0s 4ms/step -
accuracy: 0.8052 - loss: 0.4365 - val_accuracy: 0.7956 - val_loss: 0.4376
Epoch 16/100
27/27             0s 4ms/step -
accuracy: 0.8080 - loss: 0.4376 - val_accuracy: 0.7762 - val_loss: 0.4726
Epoch 17/100
27/27             0s 4ms/step -
accuracy: 0.7985 - loss: 0.4432 - val_accuracy: 0.7901 - val_loss: 0.4514
Epoch 18/100
27/27             0s 4ms/step -
accuracy: 0.7919 - loss: 0.4668 - val_accuracy: 0.8011 - val_loss: 0.4313
Epoch 19/100
27/27             0s 4ms/step -
accuracy: 0.8242 - loss: 0.4209 - val_accuracy: 0.7873 - val_loss: 0.4430
Epoch 20/100
27/27             0s 4ms/step -
accuracy: 0.8025 - loss: 0.4336 - val_accuracy: 0.8149 - val_loss: 0.4248
Epoch 21/100
27/27             0s 5ms/step -
accuracy: 0.7884 - loss: 0.4528 - val_accuracy: 0.8149 - val_loss: 0.4321
Epoch 22/100
27/27             0s 5ms/step -
accuracy: 0.8247 - loss: 0.4307 - val_accuracy: 0.8122 - val_loss: 0.4266
Epoch 23/100
27/27             0s 3ms/step -
accuracy: 0.8166 - loss: 0.4305 - val_accuracy: 0.8094 - val_loss: 0.4294
Epoch 24/100
27/27             0s 3ms/step -
accuracy: 0.8193 - loss: 0.4161 - val_accuracy: 0.8039 - val_loss: 0.4372
Epoch 25/100
27/27             0s 3ms/step -
accuracy: 0.8042 - loss: 0.4393 - val_accuracy: 0.8066 - val_loss: 0.4314
Epoch 26/100
27/27             0s 4ms/step -
accuracy: 0.7864 - loss: 0.4375 - val_accuracy: 0.8011 - val_loss: 0.4363
Epoch 27/100
27/27             0s 3ms/step -
accuracy: 0.8050 - loss: 0.4302 - val_accuracy: 0.8260 - val_loss: 0.4249
Epoch 28/100
27/27             0s 3ms/step -
accuracy: 0.7862 - loss: 0.4552 - val_accuracy: 0.8094 - val_loss: 0.4311
Epoch 29/100
27/27             0s 3ms/step -
```

```
accuracy: 0.8258 - loss: 0.4232 - val_accuracy: 0.8066 - val_loss: 0.4278
Epoch 30/100
27/27             0s 3ms/step -
accuracy: 0.8237 - loss: 0.4129 - val_accuracy: 0.8011 - val_loss: 0.4410
```

28.2 Making the predictions and evaluating the model

[ ]:
```
y_pred = model.predict(X_test)
y_pred = (y_pred > 0.5)
```

```
12/12             0s 0s/step
```

28.2.1 confusion matrix

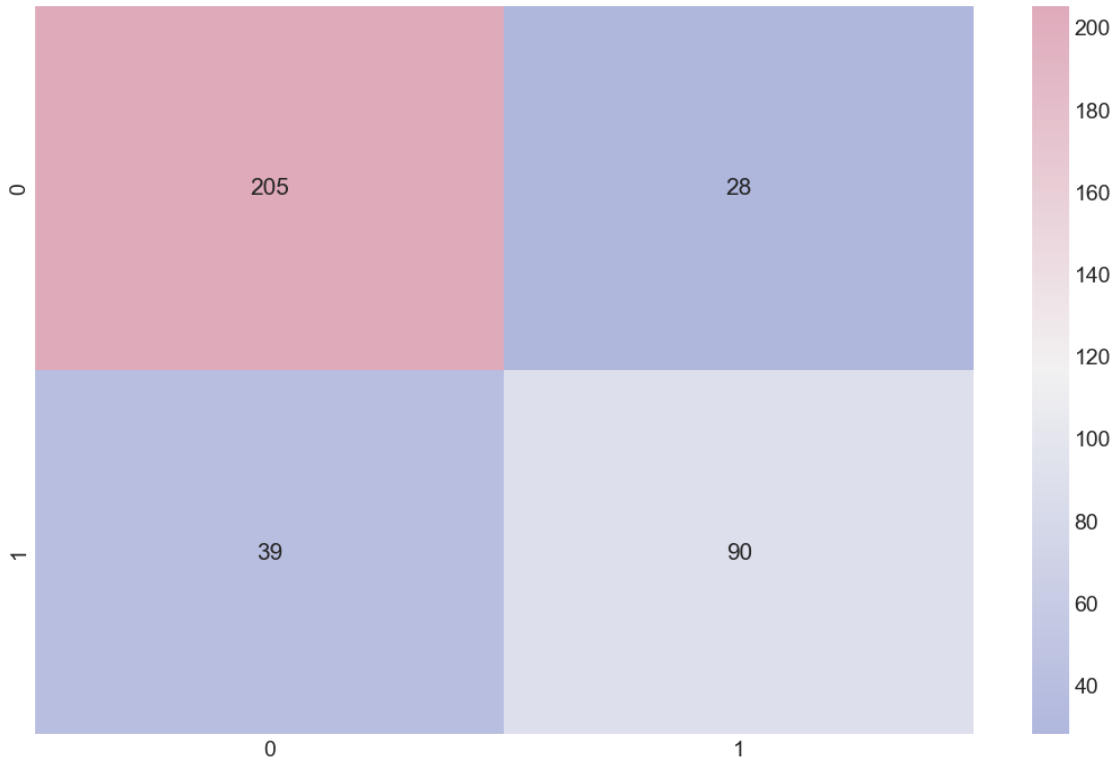
[ ]:
```
cf_matrix = confusion_matrix(y_test, y_pred)
print(cf_matrix)
accuracy_score(y_test,y_pred)
```

```
[[205  28]
 [ 39  90]]
```

[ ]:
```
0.8149171270718232
```

[ ]:
```
cmap1 = sns.diverging_palette(260, -10, s=50, l=75, n=5, as_cmap=True)
plt.subplots(figsize=(12, 8))
cf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(cf_matrix, cmap=cmap1, annot=True, fmt='.0f', annot_kws={'size':␣
↪15})
```

[ ]:
```
<Axes: >
```



28.2.2 predictions on testfile

[ ]:
```
predictions = model.predict(test_df)
predictions = (predictions > 0.5).astype(int)


df_predictions = pd.DataFrame(predictions, columns=["Survive"])

print(df_predictions)
```

```
1/1             0s 62ms/step
    Survive
0         0
1         0
2         0
3         0
4         0
5         0
6         1
7         1
8         1
9         0
10        0
```

11	0
12	1
13	0
14	1
15	1
16	0
17	0
18	0