In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
X=pd.read_csv(r"C:\Users\Admin\Downloads\alzheimers_disease_data.csv")
X
```

Out[1]:

	PatientID	Age	Gender	Ethnicity	EducationLevel	BMI	Smoking	AlcoholConsur
0	4751	73	0	0	2	22.927749	0	13.2
1	4752	89	0	0	0	26.827681	0	4.5
2	4753	73	0	3	1	17.795882	0	19.5
3	4754	74	1	0	1	33.800817	1	12.2
4	4755	89	0	0	0	20.716974	0	18.4
...	...	...	...	...	...	...	...	
2144	6895	61	0	0	1	39.121757	0	1.5
2145	6896	75	0	0	2	17.857903	0	18.7
2146	6897	77	0	0	1	15.476479	0	4.5
2147	6898	78	1	3	1	15.299911	0	8.6
2148	6899	72	0	0	2	33.289738	0	7.8

2149 rows × 35 columns

In [2]:
```
pd.set_option("Display.max_columns",100)
```

In [3]:
```
X
```

Out[3]:

	PatientID	Age	Gender	Ethnicity	EducationLevel	BMI	Smoking	AlcoholConsur
0	4751	73	0	0	2	22.927749	0	13.2
1	4752	89	0	0	0	26.827681	0	4.5
2	4753	73	0	3	1	17.795882	0	19.5
3	4754	74	1	0	1	33.800817	1	12.2
4	4755	89	0	0	0	20.716974	0	18.4
...	...	...	...	...	...	...	...	
2144	6895	61	0	0	1	39.121757	0	1.5
2145	6896	75	0	0	2	17.857903	0	18.7
2146	6897	77	0	0	1	15.476479	0	4.5
2147	6898	78	1	3	1	15.299911	0	8.6
2148	6899	72	0	0	2	33.289738	0	7.8

2149 rows × 35 columns

In [4]:
```
1 X.shape
```

Out[4]:
```
(2149, 35)
```

In [5]:
```
1 X.isnull().sum()
```

Out[5]:
```
PatientID                    0
Age                          0
Gender                       0
Ethnicity                    0
EducationLevel               0
BMI                          0
Smoking                      0
AlcoholConsumption           0
PhysicalActivity             0
DietQuality                  0
SleepQuality                 0
FamilyHistoryAlzheimers      0
CardiovascularDisease        0
Diabetes                     0
Depression                   0
HeadInjury                   0
Hypertension                 0
SystolicBP                   0
DiastolicBP                  0
CholesterolTotal             0
CholesterolLDL               0
CholesterolHDL               0
CholesterolTriglycerides     0
MMSE                         0
FunctionalAssessment         0
MemoryComplaints             0
BehavioralProblems           0
ADL                          0
Confusion                    0
Disorientation               0
PersonalityChanges           0
DifficultyCompletingTasks    0
Forgetfulness                0
Diagnosis                    0
DoctorInCharge               0
dtype: int64
```

In [6]:
```
1 X[['Age','Diagnosis']].value_counts()
```

Out[6]:
```
Age  Diagnosis
72   0            61
88   0            56
68   0            55
67   0            53
76   0            51
                  ..
64   1            19
74   1            18
69   1            17
79   1            16
86   1            15
Name: count, Length: 62, dtype: int64
```

In [7]:
```python
X[['Age','Diagnosis']].min()
```

Out[7]:
```
Age          60
Diagnosis     0
dtype: int64
```

In [8]:
```python
X[['Age','Diagnosis']].max()
```

Out[8]:
```
Age          90
Diagnosis     1
dtype: int64
```

In [9]:
```python
X1=X.loc[(X['BehavioralProblems']==1)&(X['Diagnosis']==1)]
X1
```

Out[9]:

	PatientID	Age	Gender	Ethnicity	EducationLevel	BMI	Smoking	AlcoholConsur
19	4770	68	0	0	3	20.041400	0	18.4
20	4771	82	1	0	0	36.223099	0	4.1
24	4775	64	1	0	1	15.457688	1	9.7
46	4797	71	0	1	1	15.648055	0	17.8
48	4799	87	0	1	2	33.476971	1	15.8
...	...	...	...	...	...	...	...	
2088	6839	64	1	0	2	39.218489	1	0.8
2126	6877	75	1	1	1	35.832125	0	13.0
2132	6883	65	1	0	1	18.517832	0	8.2
2136	6887	83	1	2	2	19.006926	0	13.9
2145	6896	75	0	0	2	17.857903	0	18.7

203 rows × 35 columns

In [10]:
```
X.loc[(X['DifficultyCompletingTasks']==1)&(X['Diagnosis']==1)]
```

Out[10]:

	PatientID	Age	Gender	Ethnicity	EducationLevel	BMI	Smoking	AlcoholConsur
45	4796	68	1	0	1	26.804164	0	19.6
58	4809	83	1	1	2	28.997201	0	14.8
65	4816	90	1	1	1	28.191097	0	10.4
72	4823	89	1	1	1	28.443777	0	4.3
78	4829	82	1	3	2	15.908275	0	16.3
...	...	...	...	...	...	...	...	
2130	6881	81	1	2	0	22.630945	1	2.3
2136	6887	83	1	2	2	19.006926	0	13.9
2138	6889	71	0	0	2	36.796170	0	16.1
2142	6893	88	0	0	0	20.097600	0	4.0
2143	6894	66	1	2	1	32.013806	1	9.3

124 rows × 35 columns

In [11]:
```
X2=X.loc[(X['ADL']<=5)&(X['Diagnosis']==1)]
X2
```

Out[11]:

	PatientID	Age	Gender	Ethnicity	EducationLevel	BMI	Smoking	AlcoholConsur
7	4758	75	0	0	1	18.776009	0	13.7
13	4764	78	1	0	1	28.870652	1	10.1
15	4766	69	0	0	1	18.045917	0	8.1
16	4767	63	1	1	2	22.822896	1	4.4
17	4768	65	1	0	1	16.333283	1	4.1
...	...	...	...	...	...	...	...	
2135	6886	60	1	0	2	15.435244	0	15.6
2138	6889	71	0	0	2	36.796170	0	16.1
2143	6894	66	1	2	1	32.013806	1	9.3
2144	6895	61	0	0	1	39.121757	0	1.5
2147	6898	78	1	3	1	15.299911	0	8.6

567 rows × 35 columns

In [12]:

```
X.loc[(X['FunctionalAssessment']<=5)&(X['Diagnosis']==1)]
```

Out[12]:

	PatientID	Age	Gender	Ethnicity	EducationLevel	BMI	Smoking	AlcoholConsur
7	4758	75	0	0	1	18.776009	0	13.7
13	4764	78	1	0	1	28.870652	1	10.1
15	4766	69	0	0	1	18.045917	0	8.1
17	4768	65	1	0	1	16.333283	1	4.1
19	4770	68	0	0	3	20.041400	0	18.4
...	...	...	...	...	...	...	...	
2133	6884	87	0	0	0	35.580967	0	17.1
2136	6887	83	1	2	2	19.006926	0	13.9
2143	6894	66	1	2	1	32.013806	1	9.3
2144	6895	61	0	0	1	39.121757	0	1.5
2146	6897	77	0	0	1	15.476479	0	4.5

584 rows × 35 columns

In [13]:

```
x=X.loc[X['Diagnosis']==1]
x
```

Out[13]:

	PatientID	Age	Gender	Ethnicity	EducationLevel	BMI	Smoking	AlcoholConsur
7	4758	75	0	0	1	18.776009	0	13.7
13	4764	78	1	0	1	28.870652	1	10.1
15	4766	69	0	0	1	18.045917	0	8.1
16	4767	63	1	1	2	22.822896	1	4.4
17	4768	65	1	0	1	16.333283	1	4.1
...	...	...	...	...	...	...	...	
2143	6894	66	1	2	1	32.013806	1	9.3
2144	6895	61	0	0	1	39.121757	0	1.5
2145	6896	75	0	0	2	17.857903	0	18.7
2146	6897	77	0	0	1	15.476479	0	4.5
2147	6898	78	1	3	1	15.299911	0	8.6

760 rows × 35 columns

In [14]:
```
x['Age'].value_counts().reset_index()
```

Out[14]:

	Age	count
0	61	32
1	90	31
2	71	31
3	76	30
4	78	29
5	68	29
6	84	29
7	63	28
8	88	28
9	70	27
10	89	27
11	83	27
12	60	26
13	75	26
14	66	26
15	62	25
16	73	24
17	80	24
18	67	24
19	87	24
20	65	24
21	77	23
22	72	21
23	85	21
24	81	20
25	82	19
26	64	19
27	74	18
28	69	17
29	79	16
30	86	15

In [15]:
```
X[['CholesterolTotal','CholesterolLDL','CholesterolHDL']].max()
```

Out[15]:
```
CholesterolTotal    299.993352
CholesterolLDL      199.965665
CholesterolHDL       99.980324
dtype: float64
```

In [16]:
```
X[['CholesterolTotal','CholesterolLDL','CholesterolHDL']].min()
```

Out[16]:
```
CholesterolTotal    150.093316
CholesterolLDL       50.230707
CholesterolHDL       20.003434
dtype: float64
```

In [17]:
```
X[['CholesterolTotal','CholesterolLDL','CholesterolHDL','Diagnosis']].r
```

Out[17]:

	index	0
0	CholesterolTotal	225.197519
1	CholesterolLDL	124.335944
2	CholesterolHDL	59.463533
3	Diagnosis	0.353653

In [18]:
```
X.groupby(['CholesterolTotal','CholesterolLDL','CholesterolHDL'])[['Di&
```

Out[18]:

			Diagnosis
CholesterolTotal	**CholesterolLDL**	**CholesterolHDL**	
150.093316	**69.067758**	**35.440518**	1
150.135572	**137.630223**	**37.583055**	1
150.192183	**105.857772**	**79.163699**	1
150.212650	**119.251101**	**83.997809**	1
150.287014	**163.424986**	**58.714098**	1
...	**...**	**...**	...
299.868482	**88.110915**	**81.983039**	1
299.873259	**164.452648**	**62.757367**	1
299.890133	**96.737082**	**97.516957**	1
299.959991	**120.918658**	**90.372018**	1
299.993352	**64.182129**	**51.410637**	1

2149 rows × 1 columns

In [19]:
```
A=X.loc[(X['CholesterolTotal']>=240)&(X['CholesterolLDL']>=160)&(X['Cho
A
```

Out[19]:

	PatientID	Age	Gender	Ethnicity	EducationLevel	BMI	Smoking	AlcoholConsur
14	4765	64	1	0	2	27.942863	0	2.1
120	4871	70	0	0	1	39.637153	0	3.6
131	4882	60	0	1	1	16.196569	0	4.7
248	4999	66	1	1	0	35.235847	0	17.5
325	5076	88	1	0	1	18.165057	1	18.4
344	5095	72	0	1	2	26.880676	0	4.4
398	5149	68	1	0	2	23.290559	0	5.2
435	5186	71	1	0	2	24.050077	0	1.3
450	5201	61	1	3	3	34.940999	0	10.2
513	5264	84	1	0	1	27.948006	0	0.9
524	5275	62	0	3	2	34.150320	0	19.7
739	5490	88	1	0	1	23.936827	0	16.8
807	5558	79	1	2	1	25.091602	0	4.5
832	5583	66	0	0	2	39.159497	0	9.9
883	5634	84	0	0	0	39.586924	1	18.6
951	5702	90	1	0	1	27.276350	0	3.2
1064	5815	78	1	0	1	22.268722	0	9.7
1096	5847	74	0	0	2	30.536551	1	10.8
1321	6072	82	0	0	2	26.191381	0	4.0
1414	6165	71	1	1	1	23.263395	1	11.7
1473	6224	72	1	0	0	23.094319	0	4.1
1501	6252	62	1	3	0	28.632493	0	1.0
1522	6273	77	0	0	1	26.828226	0	8.5
1583	6334	66	0	3	1	38.934002	1	5.7
1802	6553	67	1	0	0	34.274316	0	17.0
1853	6604	79	0	2	1	37.977139	0	18.2
2001	6752	65	0	1	1	30.588889	1	14.0
2072	6823	68	0	0	3	34.244492	1	3.4
2141	6892	72	0	0	2	21.600144	0	19.3

In [20]:
```
A['Diagnosis'].value_counts()
```

Out[20]:
```
Diagnosis
0    19
1    10
Name: count, dtype: int64
```

In [21]:
```
A1=A.loc[A['Diagnosis']==0]
A1
```

Out[21]:

	PatientID	Age	Gender	Ethnicity	EducationLevel	BMI	Smoking	AlcoholConsur
14	4765	64	1	0	2	27.942863	0	2.
120	4871	70	0	0	1	39.637153	0	3.6
131	4882	60	0	1	1	16.196569	0	4.7
248	4999	66	1	1	0	35.235847	0	17.5
325	5076	88	1	0	1	18.165057	1	18.4
344	5095	72	0	1	2	26.880676	0	4.4
398	5149	68	1	0	2	23.290559	0	5.2
450	5201	61	1	3	3	34.940999	0	10.2
513	5264	84	1	0	1	27.948006	0	0.9
524	5275	62	0	3	2	34.150320	0	19.7
807	5558	79	1	2	1	25.091602	0	4.5
951	5702	90	1	0	1	27.276350	0	3.2
1096	5847	74	0	0	2	30.536551	1	10.8
1414	6165	71	1	1	1	23.263395	1	11.7
1501	6252	62	1	3	0	28.632493	0	1.0
1583	6334	66	0	3	1	38.934002	1	5.7
1802	6553	67	1	0	0	34.274316	0	17.0
2072	6823	68	0	0	3	34.244492	1	3.4
2141	6892	72	0	0	2	21.600144	0	19.3

In [22]:
```
#with high Cholesterol levels many people are not diagnosed with Alzhe
```

In [23]:
```
A2=X.loc[(X['CholesterolTotal']<=200)&(X['CholesterolLDL']<100)&(X['Cho
A2
```

Out[23]:

	PatientID	Age	Gender	Ethnicity	EducationLevel	BMI	Smoking	AlcoholConsur
27	4778	71	0	0	2	18.434789	0	15.9
37	4788	60	1	0	2	31.568689	0	3.4
87	4838	75	0	0	2	31.820253	0	13.3
109	4860	60	1	0	2	24.974286	0	18.4
158	4909	68	1	3	3	15.794386	0	9.3
...	...	...	...	...	...	...	...	
2000	6751	90	0	2	1	22.161590	0	11.4
2007	6758	60	1	0	1	38.214532	1	8.7
2023	6774	71	0	2	3	27.332495	0	19.6
2094	6845	85	0	0	0	24.385193	0	2.8
2130	6881	81	1	2	0	22.630945	1	2.3

83 rows × 35 columns

In [24]:
```
A3=A2.loc[(A2['Gender']==0)&(A2['Diagnosis']==0)]
A3#with optimal level of chosesterol women are less likely to diagnosed
```

Out[24]:

	PatientID	Age	Gender	Ethnicity	EducationLevel	BMI	Smoking	AlcoholConsur
27	4778	71	0	0	2	18.434789	0	15.9
87	4838	75	0	0	2	31.820253	0	13.3
220	4971	70	0	1	1	21.780690	0	11.6
252	5003	79	0	2	1	26.798031	0	6.0
259	5010	76	0	1	1	23.950445	0	13.7
281	5032	68	0	0	3	33.789883	0	7.5
336	5087	79	0	0	1	20.546211	0	19.1
374	5125	80	0	1	3	17.095404	1	14.9
443	5194	65	0	2	2	29.584935	0	5.3
446	5197	80	0	0	1	33.398410	1	7.5
599	5350	79	0	0	1	31.261029	0	10.8
650	5401	87	0	3	1	21.631578	0	14.7
667	5418	64	0	0	0	32.362132	0	10.2
761	5512	86	0	0	1	21.900205	1	0.8
798	5549	82	0	1	2	22.602948	0	1.8
853	5604	75	0	1	0	33.326818	0	11.2
860	5611	71	0	2	2	16.080044	1	19.8
967	5718	80	0	0	1	22.031611	0	19.5
1160	5911	72	0	1	0	38.646316	1	15.4
1385	6136	68	0	0	3	33.603383	0	7.7
1589	6340	73	0	1	3	19.506076	0	12.5
1591	6342	81	0	1	0	38.186388	1	13.4
1618	6369	64	0	0	0	22.413218	0	19.1
1626	6377	69	0	3	1	36.555455	0	7.3
1698	6449	82	0	0	1	26.223176	1	11.9
1823	6574	88	0	1	1	25.260763	1	8.7
1925	6676	60	0	1	0	33.361610	0	8.1
1928	6679	69	0	0	2	20.944052	1	4.5
1946	6697	66	0	3	2	23.987902	0	16.3
2094	6845	85	0	0	0	24.385193	0	2.8

In [25]:
```
A3['Diagnosis'].value_counts()
```

Out[25]:
```
Diagnosis
0    30
Name: count, dtype: int64
```

In [26]:
```
X['HeadInjury'].unique()
```

Out[26]: `array([0, 1], dtype=int64)`

In [27]:
```
X.drop(columns='DoctorInCharge',inplace=True)
```

In [28]:
```
X
```

Out[28]:

	PatientID	Age	Gender	Ethnicity	EducationLevel	BMI	Smoking	AlcoholConsur
0	4751	73	0	0	2	22.927749	0	13.2
1	4752	89	0	0	0	26.827681	0	4.5
2	4753	73	0	3	1	17.795882	0	19.5
3	4754	74	1	0	1	33.800817	1	12.2
4	4755	89	0	0	0	20.716974	0	18.4
...	...	...	...	...	...	...	...	
2144	6895	61	0	0	1	39.121757	0	1.5
2145	6896	75	0	0	2	17.857903	0	18.7
2146	6897	77	0	0	1	15.476479	0	4.5
2147	6898	78	1	3	1	15.299911	0	8.6
2148	6899	72	0	0	2	33.289738	0	7.8

2149 rows × 34 columns

In [29]:
```
X4=X.loc[X['MMSE']<=15]
X4
```

Out[29]:

	PatientID	Age	Gender	Ethnicity	EducationLevel	BMI	Smoking	AlcoholConsur
2	4753	73	0	3	1	17.795882	0	19.5
3	4754	74	1	0	1	33.800817	1	12.2
4	4755	89	0	0	0	20.716974	0	18.4
6	4757	68	0	3	2	38.387622	1	0.6
7	4758	75	0	0	1	18.776009	0	13.7
...	...	...	...	...	...	...	...	
2143	6894	66	1	2	1	32.013806	1	9.3
2144	6895	61	0	0	1	39.121757	0	1.5
2145	6896	75	0	0	2	17.857903	0	18.7
2147	6898	78	1	3	1	15.299911	0	8.6
2148	6899	72	0	0	2	33.289738	0	7.8

1112 rows × 34 columns

In [30]:
```
x2=X.loc[(X['Forgetfulness']==1)&(X['DifficultyCompletingTasks']==1)&(X
x2
```

Out[30]:

	PatientID	Age	Gender	Ethnicity	EducationLevel	BMI	Smoking	AlcoholConsur
161	4912	63	1	0	2	39.793970	0	6.4
178	4929	74	1	1	1	32.148698	1	14.8
457	5208	68	0	0	3	26.634779	0	1.5
749	5500	74	1	1	3	15.014659	1	11.6
1147	5898	90	1	3	0	24.332593	1	12.0
1245	5996	86	1	0	0	16.656064	0	1.9
1476	6227	64	1	0	1	29.714024	0	4.0
1514	6265	67	0	2	2	28.493977	0	14.6
1675	6426	83	1	0	1	36.651524	1	0.4
1699	6450	75	0	0	0	19.626719	1	10.5
1707	6458	71	0	0	2	17.545005	1	3.7
1781	6532	87	0	0	3	25.173319	0	13.1
2008	6759	74	1	0	1	32.651525	0	8.6

In [31]:
```
x2['Age'].value_counts().reset_index() #patients above 70 havimg sympto
```

Out[31]:

	Age	count
0	74	3
1	63	1
2	68	1
3	90	1
4	86	1
5	64	1
6	67	1
7	83	1
8	75	1
9	71	1
10	87	1

In [32]:
```
X4['Diagnosis'].value_counts() #MMSE score less than 15 are more likely
```

Out[32]:
```
Diagnosis
0    618
1    494
Name: count, dtype: int64
```

In [33]:
```
x1=X.loc[:,['SystolicBP','DiastolicBP','CholesterolTotal','CholesterolL
```

In [34]:
```
x1
```

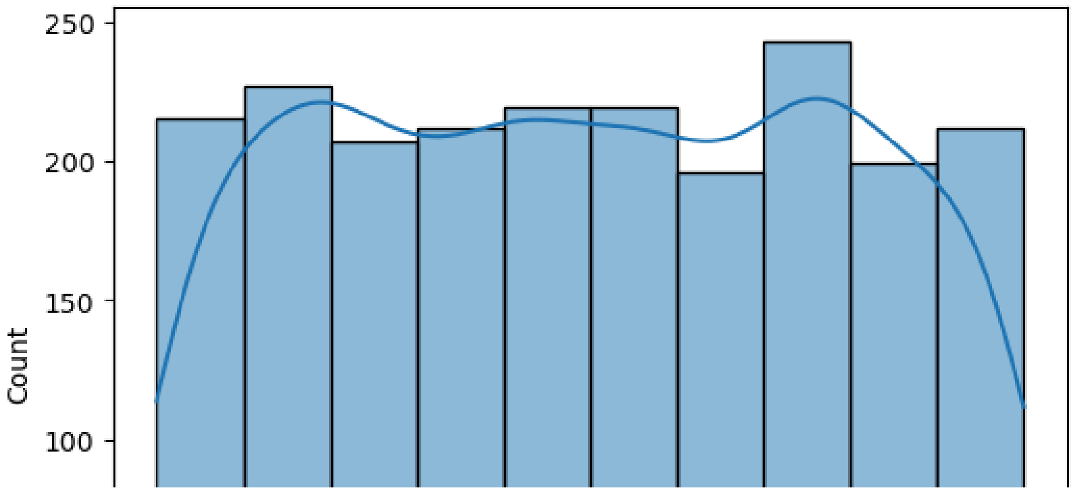
Out[34]:

	SystolicBP	DiastolicBP	CholesterolTotal	CholesterolLDL	CholesterolHDL	Cholesterol
0	142	72	242.366840	56.150897	33.682563	
1	115	64	231.162595	193.407996	79.028477	
2	99	116	284.181858	153.322762	69.772292	
3	118	115	159.582240	65.366637	68.457491	
4	94	117	237.602184	92.869700	56.874305	
...	...	...	...	...	...	
2144	122	101	280.476824	94.870490	60.943092	
2145	152	106	186.384436	95.410700	93.649735	
2146	115	118	237.024558	156.267294	99.678209	
2147	103	96	242.197192	52.482961	81.281111	
2148	166	78	283.396797	92.200064	81.920043	

2149 rows × 7 columns

In [35]:
```
import matplotlib.pyplot as plt
import seaborn as sns
for i in x1.columns:
    sns.histplot(x1[i],bins=10,kde=True)
    plt.show()
```

```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [36]:
```
X['CholesterolHDL'].mean()
```

Out[36]: 59.46353314116864

In [37]:
```
X['CholesterolHDL'].median()
```
Out[37]: 59.76823749767153

In [38]:
```
X['CholesterolHDL'].mode()[0]
```
Out[38]: 20.00343401498445

In [39]:
```
X['MMSE'].mean()
```
Out[39]: 14.7551319782826

In [40]:
```
X['MMSE'].median()
```
Out[40]: 14.441659908144548

In [41]:
```
X['MMSE'].mode()[0]
```
Out[41]: 0.0053121464417005

In [42]:
```
F=X.drop(columns='Diagnosis',axis=1)
T=X['Diagnosis']
```

In [43]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(F,T,train_size=0.90,ranc
```

In [44]:
```
from sklearn.preprocessing import MinMaxScaler
M=MinMaxScaler()
```

In [45]:
```
X_train[['PatientID','Age','BMI','SystolicBP','DiastolicBP','Cholesterc
X_train
```

Out[45]:

	PatientID	Age	Gender	Ethnicity	EducationLevel	BMI	Smoking	AlcoholCor
1611	0.750000	0.600000	1	0	1	0.278486	0	
1154	0.537244	0.066667	1	0	1	0.426405	0	
820	0.381750	0.500000	0	0	0	0.415574	0	
465	0.216480	0.533333	1	0	2	0.559979	0	
1675	0.779795	0.766667	1	0	1	0.866264	1	
...	...	...	...	...	...	...	...	
1208	0.562384	0.033333	1	0	1	0.522248	0	
172	0.080074	0.933333	1	1	2	0.652971	0	
983	0.457635	0.266667	0	0	0	0.757322	1	
1251	0.582402	0.233333	1	0	1	0.362522	1	
178	0.082868	0.466667	1	1	1	0.686035	1	

1934 rows × 33 columns

In [46]:

```
X_test[['PatientID','Age','BMI','SystolicBP','DiastolicBP','Cholesterol
X_test
```

Out[46]:

	PatientID	Age	Gender	Ethnicity	EducationLevel	BMI	Smoking	AlcoholCor
1953	0.918983	0.333333	0	1	0	0.159542	0	
65	0.029675	1.000000	1	1	1	0.531342	0	
475	0.222798	0.233333	0	0	2	0.564601	1	
1719	0.808761	0.200000	1	0	1	0.066571	0	
527	0.247292	0.533333	1	2	3	0.315526	0	
...	...	...	...	...	...	...	...	
1120	0.526613	0.633333	0	0	1	0.496167	0	
1519	0.714555	0.166667	0	0	1	0.894384	0	
590	0.276967	0.366667	0	0	1	0.251246	0	
2121	0.998116	0.033333	0	0	3	0.424122	0	
1432	0.673575	0.366667	1	0	3	0.536823	0	

215 rows × 33 columns

In [47]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
Log=LogisticRegression()
params={'C':[0.1,0.01,0.02,0.3],'penalty':['l1','l2']}
G=GridSearchCV(Log,param_grid=params,scoring="accuracy",cv=7)
```

In [48]:

```
G.fit(X_train,y_train)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_logist
ic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html (http
s://scikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression (https://scikit-learn.org/stable/modules/linear_model.html#l
ogistic-regression)
  n_iter_i = _check_optimize_result(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_logist
ic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
```

In [49]:
```
G.best_params_
```

Out[49]: {'C': 0.3, 'penalty': 'l2'}

In [50]:
```
O=G.best_estimator_
O
```

Out[50]: LogisticRegression(C=0.3)

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [51]:
```
Pr=O.predict(X_test)
Pr
```

Out[51]:
```
array([1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1], dtype=int64)
```

In [52]:
```
O.score(X_train,y_train)
```

Out[52]: 0.8500517063081696

In [53]:
```
O.score(X_test,y_test)
```

Out[53]: 0.8372093023255814

In [54]:
```
from sklearn.metrics import accuracy_score,classification_report,confus
```

In [55]:
```
accuracy_score(y_test,Pr)
```

Out[55]: 0.8372093023255814

In [56]:
```
print(classification_report(y_test,Pr))
```

```
              precision    recall  f1-score   support

           0       0.86      0.88      0.87       136
           1       0.79      0.76      0.77        79

    accuracy                           0.84       215
   macro avg       0.83      0.82      0.82       215
weighted avg       0.84      0.84      0.84       215
```

In [57]:
```
confusion_matrix(y_test,Pr)
```

Out[57]:
```
array([[120,  16],
       [ 19,  60]], dtype=int64)
```

In [58]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
KN=KNeighborsClassifier()
params={'n_neighbors':[4,3,8,10]}
V=GridSearchCV(KN,param_grid=params,scoring="accuracy",cv=7)
```

In [59]:
```
V.fit(X_train,y_train)
```

Out[59]:
```
GridSearchCV(cv=7, estimator=KNeighborsClassifier(),
             param_grid={'n_neighbors': [4, 3, 8, 10]}, scoring='accurac
y')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [60]:
```
V.best_params_
```

Out[60]:
```
{'n_neighbors': 8}
```

In [61]:
```
T=V.best_estimator_
T
```

Out[61]:
```
KNeighborsClassifier(n_neighbors=8)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [62]:
```
T.score(X_train,y_train)
```

Out[62]:
```
0.8252326783867632
```

In [63]:
```
T.score(X_test,y_test)
```

Out[63]:
```
0.7813953488372093
```

In [64]:
```
PDT=T.predict(X_test)
PDT
```

Out[64]:
```
array([1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0], dtype=int64)
```

In [65]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
s=SVC()
para={"gamma":[0.03,0.5,0.8,0.1],"kernel":["rbf"]}
g=GridSearchCV(s,param_grid=para,scoring="accuracy",cv=8)
```

In [66]:
```
g.fit(X_train,y_train)
```

Out[66]:
```
GridSearchCV(cv=8, estimator=SVC(),
             param_grid={'gamma': [0.03, 0.5, 0.8, 0.1], 'kernel': ['rb
f']},
             scoring='accuracy')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [67]:
```
g.best_params_
```

Out[67]:
```
{'gamma': 0.03, 'kernel': 'rbf'}
```

In [68]:
```
Model=g.best_estimator_
Model
```

Out[68]:
```
SVC(gamma=0.03)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [69]:
```
pre=Model.predict(X_test)
pre
```

Out[69]:
```
array([1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0,
       1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1,
       0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0], dtype=int64)
```

In [70]:
```
Model.score(X_train,y_train)
```

Out[70]:
```
0.9146845915201655
```

In [71]:
```
Model.score(X_test,y_test)
```

Out[71]:
```
0.8093023255813954
```

In [72]:
```
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import BernoulliNB
from sklearn.naive_bayes import ComplementNB
from sklearn.naive_bayes import CategoricalNB
N=GaussianNB()
B=BernoulliNB()
C=ComplementNB()
c=CategoricalNB()
```

In [73]:
```
N.fit(X_train,y_train)
```

Out[73]:
```
GaussianNB()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [74]:
```
N.score(X_train,y_train)
```

Out[74]:
```
0.8107549120992761
```

In [75]:
```
N.score(X_test,y_test)
```

Out[75]:
```
0.8418604651162791
```

In [76]:
```
B.fit(X_train,y_train)
```

Out[76]:
```
BernoulliNB()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [77]:
```
B.score(X_train,y_train)
```

Out[77]:
```
0.7145811789038262
```

In [78]:
```
B.score(X_test,y_test)
```

Out[78]:
```
0.7209302325581395
```

In [79]:
```
p=B.predict(X_test)
p
```

Out[79]:
```
array([0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0,
       0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1,
       1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0,
       0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1], dtype=int64)
```

In [80]:
```
C.fit(X_train,y_train)
```

Out[80]:
```
ComplementNB()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [81]:
```
C.score(X_train,y_train)
```

Out[81]:
```
0.7533609100310238
```

In [82]:
```
C.score(X_test,y_test)
```

Out[82]:
```
0.7255813953488373
```

In [83]:
```
c.fit(X_train,y_train)
```

Out[83]:
```
CategoricalNB()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [84]:
```
c.score(X_train,y_train)
```

Out[84]: 0.921923474663909

In [85]:
```
c.score(X_test,y_test)
```

Out[85]: 0.9069767441860465

In [86]:
```
cprd=c.predict(X_test)
cprd
```

Out[86]:
```
array([1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0,
       0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1,
       0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1], dtype=int64)
```

In [87]:
```
Y={"models":["Log","KN","s","N","B","C","c"],"train":[84.33,84.53,100,8
```

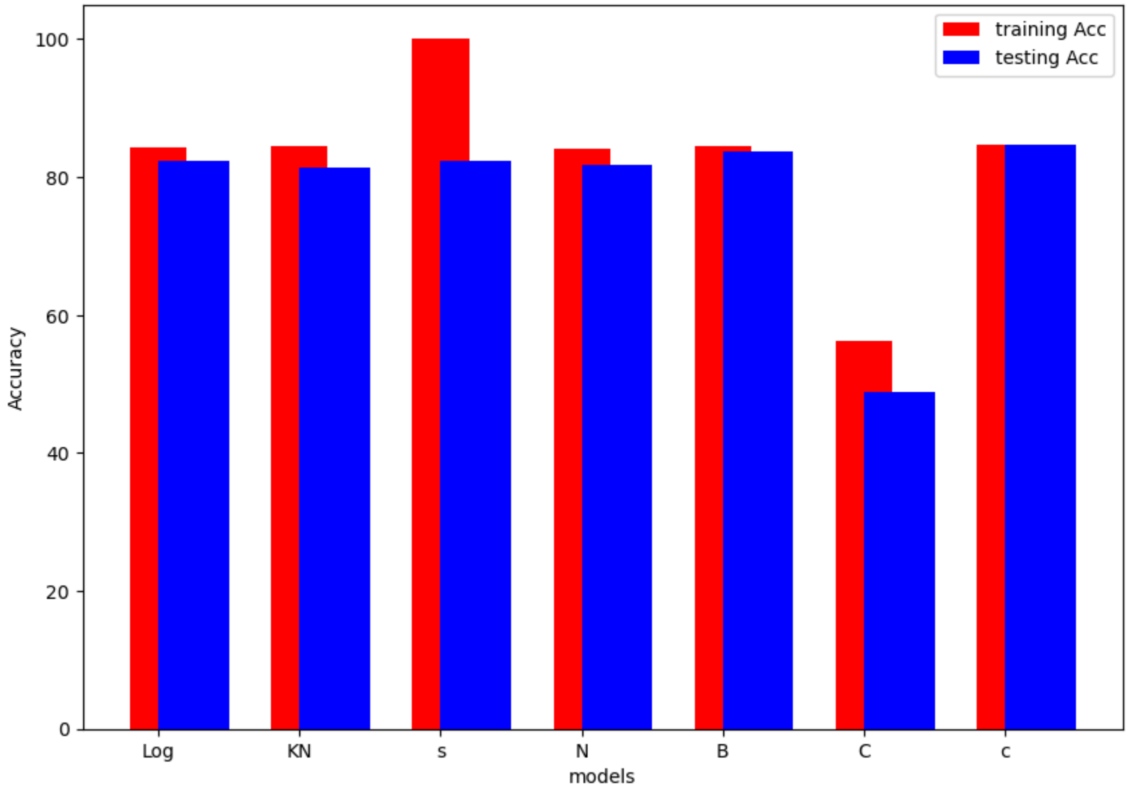
In [88]:
```
Y=pd.DataFrame(Y)
Y
```

Out[88]:

	models	train	test
0	Log	84.33	82.32
1	KN	84.53	81.39
2	s	100.00	82.32
3	N	84.02	81.86
4	B	84.48	83.72
5	C	56.25	48.83
6	c	84.69	84.69

In [89]:

```
plt.figure(figsize=(10,7))
plt.bar(Y['models'],Y['train'],color="red",width=0.4,align='center',lab
plt.bar(Y['models'],Y['test'],color="blue",width=0.5,align='edge',label
plt.legend()
plt.xlabel('models')
plt.ylabel('Accuracy')
plt.show()
```



In [90]:

```
#here CategoricalNB is working fine in comparision to others
```

#ANN

In [91]:

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Dropout
```

In [92]:
```
P=Sequential([
    Dense(140,input_shape=(X_train.shape[1],),activation="relu"),
    Dense(120,activation="relu"),
    Dense(100,activation="relu"),
    Dense(90,activation="relu"),
    Dense(60,activation="relu"),
    Dropout(0.2),
    Dense(60,activation="relu"),
    Dense(1,activation="sigmoid"),
])
```

```
C:\Users\Admin\AppData\Roaming\Python\Python311\site-packages\keras\src\la
yers\core\dense.py:87: UserWarning: Do not pass an `input_shape`/`input_di
m` argument to a layer. When using Sequential models, prefer using an `Inp
ut(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

In [93]:
```
P.compile(
loss="categorical_crossentropy",
optimizer="adam",
metrics=["accuracy"]
)
```

In [94]:
```
P.fit(X_train,y_train,epochs=60)
```

```
Epoch 1/60

C:\Users\Admin\AppData\Roaming\Python\Python311\site-packages\keras\src
\losses\losses.py:27: SyntaxWarning: In loss categorical_crossentropy,
expected y_pred.shape to be (batch_size, num_classes) with num_classes
> 1. Received: y_pred.shape=(None, 1). Consider using 'binary_crossentr
opy' if you only have 2 classes.
  return self.fn(y_true, y_pred, **self._fn_kwargs)

61/61 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.6180 - loss: 0.000
0e+00
Epoch 2/60
61/61 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.6396 - loss: 0.000
0e+00
Epoch 3/60
61/61 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.6490 - loss: 0.000
0e+00
Epoch 4/60
61/61 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.6562 - loss: 0.000
0e+00
```

In [95]:
```
Pred=P.predict(X_test)
Pred
```

```
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
```

Out[95]:
```
array([[0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
```

In [96]:
```
A=[]
for i in Pred:
    if i<=0.5:
        A.append(0)
    else:
        A.append(1)
```

In [97]:
```
A
```

Out[97]:
```
[0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
```

In [98]:
```
from sklearn.metrics import confusion_matrix,accuracy_score,classificat
```

In [99]:
```
confusion_matrix(y_test,Pred)
```

Out[99]:
```
array([[136,   0],
       [ 79,   0]], dtype=int64)
```

In [100]:
```
accuracy_score(y_test,Pred)
```

Out[100]:
```
0.6325581395348837
```

In [101]:
```
print(classification_report(y_test,Pred))
```

```
              precision    recall  f1-score   support

           0       0.63      1.00      0.77       136
           1       0.00      0.00      0.00        79

    accuracy                           0.63       215
   macro avg       0.32      0.50      0.39       215
weighted avg       0.40      0.63      0.49       215


C:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```

                                            PROJECT REPORT

Goal:
The objective of this project is to predict the likelihood of Alzheimer’s disease in patients based on features such as Forgetfulness, DifficultyCompletingTasks, BehavioralProblems, FunctionalAssessment, MMSE (Mini-Mental State Examination), age, weight, and other relevant factors. This helps in early diagnosis, enabling timely intervention and better management of the disease.

Discussion:
Insights:

#The analysis reveals that MMSE scores and FunctionalAssessment ratings are the most critical indicators of Alzheimer’s progression.
#Features like age and Forgetfulness also significantly contribute to the predictive power of the models, especially when combined with behavioral and task-completion assessments.
#Older patients with low MMSE scores, high difficulty completing tasks, and behavioral problems are more likely to exhibit Alzheimer’s symptoms.
```

```

Model Comparison:

Various machine learning models such as Random Forest, Support Vector Machines (SVM), and Gradient Boosting were compared to an Artificial Neural Network (ANN).

Conclusion:
#This project successfully developed predictive models for Alzheimer’s disease diagnosis.
#Using machine learning techniques and ANN, we demonstrated the importance of MMSE scores, behavioral patterns, and functional assessments in predicting the disease.
#The findings underscore the potential for personalized interventions and better patient care through early diagnosis.
```

In [ ]:

```

```