Mini Project
UCS-538: Data Science Fundamentals

Title

HEART STROKE PREDICTION

Submitted by: SRISHTI MITTAL

Roll No.: 102165019

Submitted to: Dr. SAURABH SHARMA

Department of ECE

THAPAR INSTITUTE OF ENGINEERING & TECHNOLOGY, PATIALA, PUNJAB

29th November 2023

1. Introduction

1.1 Overview

The Heart Stroke Prediction Project was undertaken to develop a predictive model for early detection of heart strokes utilizing data science methodologies. This report encapsulates the comprehensive process, from data preparation to model evaluation, and presents the key findings and insights gained through this analysis.

1.2 Problem Definition and Scope

Heart stroke, also known as a cerebrovascular accident (CVA) or brain attack, is a medical emergency that occurs when blood flow to the brain is disrupted, leading to damage or death of brain cells. It can have severe consequences, including long-term disability or even death. Early detection and prediction of individuals at risk of a heart stroke are crucial for preventive healthcare.

The problem at hand is to develop a predictive model that can assess the likelihood of an individual experiencing a heart stroke based on their health-related attributes and historical data. The goal is to create a tool that can aid healthcare professionals in identifying individuals who may be at an increased risk of a heart stroke, allowing for timely intervention and preventive measures.

2. Data Preparation

2.1 Dataset Used

1	age	anaemia	creatinine	diabetes	ejection_f	high_bloo	platelets	serum_cre	serum_so	sex	smoking	time	DEATH_EVENT
2	75	0	582	0	20	1	265000	1.9	130	1	0	4	1
3	55	0	7861	0	38	0	263358	1.1	136	1	0	6	1
4	65	0	146	0	20	0	162000	1.3	129	1	1	7	1
5	50	1	111	0	20	0	210000	1.9	137	1	0	7	1
6	65	1	160	1	20	0	327000	2.7	116	0	0	8	1
7	90	1	47	0	40	1	204000	2.1	132	1	1	8	1
8	75	1	246	0	15	0	127000	1.2	137	1	0	10	1
9	60	1	315	1	60	0	454000	1.1	131	1	1	10	1
10	65	0	157	0	65	0	263358	1.5	138	0	0	10	1
11	80	1	123	0	35	1	388000	9.4	133	1	1	10	1
12	75	1	81	0	38	1	368000	4	131	1	1	10	1
13	62	0	231	0	25	1	253000	0.9	140	1	1	10	1
14	45	1	981	0	30	0	136000	1.1	137	1	0	11	1
15	50	1	168	0	38	1	276000	1.1	137	1	0	11	1
16	49	1	80	0	30	1	427000	1	138	0	0	12	0
17	82	1	379	0	50	0	47000	1.3	136	1	0	13	1
18	87	1	149	0	38	0	262000	0.9	140	1	0	14	1
19	45	0	582	0	14	0	166000	0.8	127	1	0	14	1
20	70	1	125	0	25	1	237000	1	140	0	0	15	1
21	48	1	582	1	55	0	87000	1.9	121	0	0	15	1
22	65	1	52	0	25	1	276000	1.3	137	0	0	16	0
23	65	1	128	1	30	1	297000	1.6	136	0	0	20	1
24	68	1	220	0	35	1	289000	0.9	140	1	1	20	1
25	53	0	63	1	60	0	368000	0.8	135	1	0	22	0
26	75	0	582	1	30	1	263358	1.83	134	0	0	23	1
27	80	0	148	1	38	0	149000	1.9	144	1	1	23	1
28	95	1	112	0	40	1	196000	1	138	0	0	24	1
29	70	0	122	1	45	1	284000	1.3	136	1	1	26	1
30	58	1	60	0	38	0	153000	5.8	134	1	0	26	1
31	82	0	70	1	30	0	200000	1.2	132	1	1	26	1
32	94	0	582	1	38	1	263358	1.83	134	1	0	27	1
33	85	0	23	0	45	0	360000	3	132	1	0	28	1
34	50	1	249	1	35	1	319000	1	128	0	0	28	1
35	50	1	159	1	30	0	302000	1.2	138	0	0	29	0
36	65	0	94	1	50	1	188000	1	140	1	0	29	1
37	69	0	582	1	35	0	228000	3.5	134	1	0	30	1
38	90	1	60	1	50	0	226000	1	134	1	0	30	1
39	82	1	855	1	50	1	321000	1	145	0	0	30	1
40	60	0	2656	1	30	0	305000	2.3	137	1	0	30	0
41	60	0	235	1	38	0	329000	3	142	0	0	30	1
42	70	0	582	0	20	1	263358	1.83	134	1	1	31	1
43	50	0	124	1	30	1	153000	1.2	136	0	1	32	1
44	70	0	571	1	45	1	185000	1.2	139	1	1	33	1
45	72	0	127	1	50	1	218000	1	134	1	0	33	0
46	60	1	588	1	60	0	194000	1.1	142	0	0	33	1
47	50	0	582	1	38	0	310000	1.9	135	1	1	35	1
48	51	0	1380	0	25	1	271000	0.9	130	1	0	38	1
49	60	0	582	1	38	1	451000	0.6	138	1	1	40	1
50	80	1	553	0	20	1	140000	4.4	133	1	0	41	1
51	57	1	129	0	30	0	395000	1	140	0	0	42	1
52	68	1	577	0	25	1	166000	1	138	1	0	43	1
53	53	1	91	0	20	1	418000	1.4	139	0	0	43	1
54	60	0	3964	1	62	0	263358	6.8	146	0	0	43	1
55	70	1	69	1	50	1	351000	1	134	0	0	44	1
56	60	1	260	1	38	0	255000	2.2	132	0	1	45	1
57	95	1	371	0	30	0	461000	2	132	1	0	50	1

58	70	1	75	0	35	0	223000	2.7	138	1	1	54	0
59	60	1	607	0	40	0	216000	0.6	138	1	1	54	0
60	49	0	789	0	20	1	319000	1.1	136	1	1	55	1
61	72	0	364	1	20	1	254000	1.3	136	1	1	59	1
62	45	0	7702	1	25	1	390000	1	139	1	0	60	1
63	50	0	318	0	40	1	216000	2.3	131	0	0	60	1
64	55	0	109	0	35	0	254000	1.1	139	1	1	60	0
65	45	0	582	0	35	0	385000	1	145	1	0	61	1
66	45	0	582	0	80	0	263358	1.18	137	0	0	63	0
67	60	0	68	0	20	0	119000	2.9	127	1	1	64	1
68	42	1	250	1	15	0	213000	1.3	136	0	0	65	1
69	72	1	110	0	25	0	274000	1	140	1	1	65	1
70	70	0	161	0	25	0	244000	1.2	142	0	0	66	1
71	65	0	113	1	25	0	497000	1.83	135	1	0	67	1
72	41	0	148	0	40	0	374000	0.8	140	1	1	68	0
73	58	0	582	1	35	0	122000	0.9	139	1	1	71	0
74	85	0	5882	0	35	0	243000	1	132	1	1	72	1
75	65	0	224	1	50	0	149000	1.3	137	1	1	72	0
76	69	0	582	0	20	0	266000	1.2	134	1	1	73	1
77	60	1	47	0	20	0	204000	0.7	139	1	1	73	1
78	70	0	92	0	60	1	317000	0.8	140	0	1	74	0
79	42	0	102	1	40	0	237000	1.2	140	1	0	74	0
80	75	1	203	1	38	1	283000	0.6	131	1	1	74	0
81	55	0	336	0	45	1	324000	0.9	140	0	0	74	0
82	70	0	69	0	40	0	293000	1.7	136	0	0	75	0
83	67	0	582	0	50	0	263358	1.18	137	1	1	76	0
84	60	1	76	1	25	0	196000	2.5	132	0	0	77	1
85	79	1	55	0	50	1	172000	1.8	133	1	0	78	0
86	59	1	280	1	25	1	302000	1	141	0	0	78	1
87	51	0	78	0	50	0	406000	0.7	140	1	0	79	0
88	55	0	47	0	35	1	173000	1.1	137	1	0	79	0
89	65	1	68	1	60	1	304000	0.8	140	1	0	79	0
90	44	0	84	1	40	1	235000	0.7	139	1	0	79	0
91	57	1	115	0	25	1	181000	1.1	144	1	0	79	0
92	70	0	66	1	45	0	249000	0.8	136	1	1	80	0
93	60	0	897	1	45	0	297000	1	133	1	0	80	0
94	42	0	582	0	60	0	263358	1.18	137	0	0	82	0
95	60	1	154	0	25	0	210000	1.7	135	1	0	82	1
96	58	0	144	1	38	1	327000	0.7	142	0	0	83	0
97	58	1	133	0	60	1	219000	1	141	1	0	83	0
98	63	1	514	1	25	1	254000	1.3	134	1	0	83	0
99	70	1	59	0	60	0	255000	1.1	136	0	0	85	0
100	60	1	156	1	25	1	318000	1.2	137	0	0	85	0
101	63	1	61	1	40	0	221000	1.1	140	0	0	86	0
102	65	1	305	0	25	0	298000	1.1	141	1	0	87	0
103	75	0	582	0	45	1	263358	1.18	137	1	0	87	0
104	80	0	898	0	25	0	149000	1.1	144	1	1	87	0
105	42	0	5209	0	30	0	226000	1	140	1	1	87	0
106	60	0	53	0	50	1	286000	2.3	143	0	0	87	0
107	72	1	328	0	30	1	621000	1.7	138	0	1	88	1
108	55	0	748	0	45	0	263000	1.3	137	1	0	88	0
109	45	1	1876	1	35	0	226000	0.9	138	1	0	88	0
110	63	0	936	0	38	0	304000	1.1	133	1	1	88	0
111	45	0	292	1	35	0	850000	1.3	142	1	1	88	0
112	85	0	129	0	60	0	306000	1.2	132	1	1	90	1
113	55	0	60	0	35	0	228000	1.2	135	1	1	90	0
114	50	0	369	1	25	0	252000	1.6	136	1	0	90	0

115	70	1	143	0	60	0	351000	1.3	137	0	0	90	1
116	60	1	754	1	40	1	328000	1.2	126	1	0	91	0
117	58	1	400	0	40	0	164000	1	139	0	0	91	0
118	60	1	96	1	60	1	271000	0.7	136	0	0	94	0
119	85	1	102	0	60	0	507000	3.2	138	0	0	94	0
120	65	1	113	1	60	1	203000	0.9	140	0	0	94	0
121	86	0	582	0	38	0	263358	1.83	134	0	0	95	1
122	60	1	737	0	60	1	210000	1.5	135	1	1	95	0
123	66	1	68	1	38	1	162000	1	136	0	0	95	0
124	60	0	96	1	38	0	228000	0.75	140	0	0	95	0
125	60	1	582	0	30	1	127000	0.9	145	0	0	95	0
126	60	0	582	0	40	0	217000	3.7	134	1	0	96	1
127	43	1	358	0	50	0	237000	1.3	135	0	0	97	0
128	46	0	168	1	17	1	271000	2.1	124	0	0	100	1
129	58	1	200	1	60	0	300000	0.8	137	0	0	104	0
130	61	0	248	0	30	1	267000	0.7	136	1	1	104	0
131	53	1	270	1	35	0	227000	3.4	145	1	0	105	0
132	53	1	1808	0	60	1	249000	0.7	138	1	1	106	0
133	60	1	1082	1	45	0	250000	6.1	131	1	0	107	0
134	46	0	719	0	40	1	263358	1.18	137	0	0	107	0
135	63	0	193	0	60	1	295000	1.3	145	1	1	107	0
136	81	0	4540	0	35	0	231000	1.18	137	1	1	107	0
137	75	0	582	0	40	0	263358	1.18	137	1	0	107	0
138	65	1	59	1	60	0	172000	0.9	137	0	0	107	0
139	68	1	646	0	25	0	305000	2.1	130	1	0	108	0
140	62	0	281	1	35	0	221000	1	136	0	0	108	0
141	50	0	1548	0	30	1	211000	0.8	138	1	0	108	0
142	80	0	805	0	38	0	263358	1.1	134	1	0	109	1
143	46	1	291	0	35	0	348000	0.9	140	0	0	109	0
144	50	0	482	1	30	0	329000	0.9	132	0	0	109	0
145	61	1	84	0	40	1	229000	0.9	141	0	0	110	0
146	72	1	943	0	25	1	338000	1.7	139	1	1	111	1
147	50	0	185	0	30	0	266000	0.7	141	1	1	112	0
148	52	0	132	0	30	0	218000	0.7	136	1	1	112	0
149	64	0	1610	0	60	0	242000	1	137	1	0	113	0
150	75	1	582	0	30	0	225000	1.83	134	1	0	113	1
151	60	0	2261	0	35	1	228000	0.9	136	1	0	115	0
152	72	0	233	0	45	1	235000	2.5	135	0	0	115	1
153	62	0	30	1	60	1	244000	0.9	139	1	0	117	0
154	50	0	115	0	45	1	184000	0.9	134	1	1	118	0
155	50	0	1846	1	35	0	263358	1.18	137	1	1	119	0
156	65	1	335	0	35	1	235000	0.8	136	0	0	120	0
157	60	1	231	1	25	0	194000	1.7	140	1	0	120	0
158	52	1	58	0	35	0	277000	1.4	136	0	0	120	0
159	50	0	250	0	25	0	262000	1	136	1	1	120	0
160	85	1	910	0	50	0	235000	1.3	134	1	0	121	0
161	59	1	129	0	45	1	362000	1.1	139	1	1	121	0
162	66	1	72	0	40	1	242000	1.2	134	1	0	121	0
163	45	1	130	0	35	0	174000	0.8	139	1	1	121	0
164	63	1	582	0	40	0	448000	0.9	137	1	1	123	0
165	50	1	2334	1	35	0	75000	0.9	142	0	0	126	1
166	45	0	2442	1	30	0	334000	1.1	139	1	0	129	1
167	80	0	776	1	38	1	192000	1.3	135	0	0	130	1
168	53	0	196	0	60	0	220000	0.7	133	1	1	134	0
169	59	0	66	1	20	0	70000	2.4	134	1	0	135	1
170	65	0	582	1	40	0	270000	1	138	0	0	140	0
171	70	0	835	0	35	1	305000	0.8	133	0	0	145	0

172	51	1	582	1	35	0	263358	1.5	136	1	1	145	0
173	52	0	3966	0	40	0	325000	0.9	140	1	1	146	0
174	70	1	171	0	60	1	176000	1.1	145	1	1	146	0
175	50	1	115	0	20	0	189000	0.8	139	1	0	146	0
176	65	0	198	1	35	1	281000	0.9	137	1	1	146	0
177	60	1	95	0	60	0	337000	1	138	1	1	146	0
178	69	0	1419	0	40	0	105000	1	135	1	1	147	0
179	49	1	69	0	50	0	132000	1	140	0	0	147	0
180	63	1	122	1	60	0	267000	1.2	145	1	0	147	0
181	55	0	835	0	40	0	279000	0.7	140	1	1	147	0
182	40	0	478	1	30	0	303000	0.9	136	1	0	148	0
183	59	1	176	1	25	0	221000	1	136	1	1	150	1
184	65	0	395	1	25	0	265000	1.2	136	1	1	154	1
185	75	0	99	0	38	1	224000	2.5	134	1	0	162	1
186	58	1	145	0	25	0	219000	1.2	137	1	1	170	1
187	60.667	1	104	1	30	0	389000	1.5	136	1	0	171	1
188	50	0	582	0	50	0	153000	0.6	134	0	0	172	1
189	60	0	1896	1	25	0	365000	2.1	144	0	0	172	1
190	60.667	1	151	1	40	1	201000	1	136	0	0	172	0
191	40	0	244	0	45	1	275000	0.9	140	0	0	174	0
192	80	0	582	1	35	0	350000	2.1	134	1	0	174	0
193	64	1	62	0	60	0	309000	1.5	135	0	0	174	0
194	50	1	121	1	40	0	260000	0.7	130	1	0	175	0
195	73	1	231	1	30	0	160000	1.18	142	1	1	180	0
196	45	0	582	0	20	1	126000	1.6	135	1	0	180	1
197	77	1	418	0	45	0	223000	1.8	145	1	0	180	1
198	45	0	582	1	38	1	263358	1.18	137	0	0	185	0
199	65	0	167	0	30	0	259000	0.8	138	0	0	186	0
200	50	1	582	1	20	1	279000	1	134	0	0	186	0
201	60	0	1211	1	35	0	263358	1.8	113	1	1	186	0
202	63	1	1767	0	45	0	73000	0.7	137	1	0	186	0
203	45	0	308	1	60	1	377000	1	136	1	0	186	0
204	70	0	97	0	60	1	220000	0.9	138	1	0	186	0
205	60	0	59	0	25	1	212000	3.5	136	1	1	187	0
206	78	1	64	0	40	0	277000	0.7	137	1	1	187	0
207	50	1	167	1	45	0	362000	1	136	0	0	187	0
208	40	1	101	0	40	0	226000	0.8	141	0	0	187	0
209	85	0	212	0	38	0	186000	0.9	136	1	0	187	0
210	60	1	2281	1	40	0	283000	1	141	0	0	187	0
211	49	0	972	1	35	1	268000	0.8	130	0	0	187	0
212	70	0	212	1	17	1	389000	1	136	1	1	188	0
213	50	0	582	0	62	1	147000	0.8	140	1	1	192	0
214	78	0	224	0	50	0	481000	1.4	138	1	1	192	0
215	48	1	131	1	30	1	244000	1.6	130	0	0	193	1
216	65	1	135	0	35	1	290000	0.8	134	1	0	194	0
217	73	0	582	0	35	1	203000	1.3	134	1	0	195	0
218	70	0	1202	0	50	1	358000	0.9	141	0	0	196	0
219	54	1	427	0	70	1	151000	9	137	0	0	196	1
220	68	1	1021	1	35	0	271000	1.1	134	1	0	197	0
221	55	0	582	1	35	1	371000	0.7	140	0	0	197	0
222	73	0	582	0	20	0	263358	1.83	134	1	0	198	1
223	65	0	118	0	50	0	194000	1.1	145	1	1	200	0
224	42	1	86	0	35	0	365000	1.1	139	1	1	201	0
225	47	0	582	0	25	0	130000	0.8	134	1	0	201	0
226	58	0	582	1	25	0	504000	1	138	1	0	205	0
227	75	0	675	1	60	0	265000	1.4	125	0	0	205	0
228	58	1	57	0	25	0	189000	1.3	132	1	1	205	0

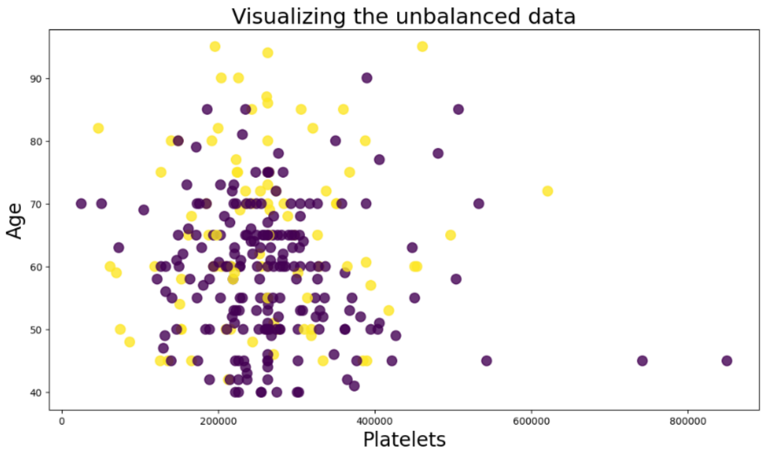
229	55	1	2794	0	35	1	141000	1	140	1	0	206	0
230	65	0	56	0	25	0	237000	5	130	0	0	207	0
231	72	0	211	0	25	0	274000	1.2	134	0	0	207	0
232	60	0	166	0	30	0	62000	1.7	127	0	0	207	1
233	70	0	93	0	35	0	185000	1.1	134	1	1	208	0
234	40	1	129	0	35	0	255000	0.9	137	1	0	209	0
235	53	1	707	0	38	0	330000	1.4	137	1	1	209	0
236	53	1	582	0	45	0	305000	1.1	137	1	1	209	0
237	77	1	109	0	50	1	406000	1.1	137	1	0	209	0
238	75	0	119	0	50	1	248000	1.1	148	1	0	209	0
239	70	0	232	0	30	0	173000	1.2	132	1	0	210	0
240	65	1	720	1	40	0	257000	1	136	0	0	210	0
241	55	1	180	0	45	0	263358	1.18	137	1	1	211	0
242	70	0	81	1	35	1	533000	1.3	139	0	0	212	0
243	65	0	582	1	30	0	249000	1.3	136	1	1	212	0
244	40	0	90	0	35	0	255000	1.1	136	1	1	212	0
245	73	1	1185	0	40	1	220000	0.9	141	0	0	213	0
246	54	0	582	1	38	0	264000	1.8	134	1	0	213	0
247	61	1	80	1	38	0	282000	1.4	137	1	0	213	0
248	55	0	2017	0	25	0	314000	1.1	138	1	0	214	1
249	64	0	143	0	25	0	246000	2.4	135	1	0	214	0
250	40	0	624	0	35	0	301000	1	142	1	1	214	0
251	53	0	207	1	40	0	223000	1.2	130	0	0	214	0
252	50	0	2522	0	30	1	404000	0.5	139	0	0	214	0
253	55	0	572	1	35	0	231000	0.8	143	0	0	215	0
254	50	0	245	0	45	1	274000	1	133	1	0	215	0
255	70	0	88	1	35	1	236000	1.2	132	0	0	215	0
256	53	1	446	0	60	1	263358	1	139	1	0	215	0
257	52	1	191	1	30	1	334000	1	142	1	1	216	0
258	65	0	326	0	38	0	294000	1.7	139	0	0	220	0
259	58	0	132	1	38	1	253000	1	139	1	0	230	0
260	45	1	66	1	25	0	233000	0.8	135	1	0	230	0
261	53	0	56	0	50	0	308000	0.7	135	1	1	231	0
262	55	0	66	0	40	0	203000	1	138	1	0	233	0
263	62	1	655	0	40	0	283000	0.7	133	0	0	233	0
264	65	1	258	1	25	0	198000	1.4	129	1	0	235	1
265	68	1	157	1	60	0	208000	1	140	0	0	237	0
266	61	0	582	1	38	0	147000	1.2	141	1	0	237	0
267	50	1	298	0	35	0	362000	0.9	140	1	1	240	0
268	55	0	1199	0	20	0	263358	1.83	134	1	1	241	1
269	56	1	135	1	38	0	133000	1.7	140	1	0	244	0
270	45	0	582	1	38	0	302000	0.9	140	0	0	244	0
271	40	0	582	1	35	0	222000	1	132	1	0	244	0
272	44	0	582	1	30	1	263358	1.6	130	1	1	244	0
273	51	0	582	1	40	0	221000	0.9	134	0	0	244	0
274	67	0	213	0	38	0	215000	1.2	133	0	0	245	0
275	42	0	64	0	40	0	189000	0.7	140	1	0	245	0
276	60	1	257	1	30	0	150000	1	137	1	1	245	0
277	45	0	582	0	38	1	422000	0.8	137	0	0	245	0
278	70	0	618	0	35	0	327000	1.1	142	0	0	245	0
279	70	0	582	1	38	0	25100	1.1	140	1	0	246	0
280	50	1	1051	1	30	0	232000	0.7	136	0	0	246	0
281	55	0	84	1	38	0	451000	1.3	136	0	0	246	0
282	70	0	2695	1	40	0	241000	1	137	1	0	247	0
283	70	0	582	0	40	0	51000	2.7	136	1	1	250	0
284	42	0	64	0	30	0	215000	3.8	128	1	1	250	0
285	65	0	1688	0	38	0	263358	1.1	138	1	1	250	0

286	50	1	54	0	40	0	279000	0.8	141	1	0	250	0
287	55	1	170	1	40	0	336000	1.2	135	1	0	250	0
288	60	0	253	0	35	0	279000	1.7	140	1	0	250	0
289	45	0	582	1	55	0	543000	1	132	0	0	250	0
290	65	0	892	1	35	0	263358	1.1	142	0	0	256	0
291	90	1	337	0	38	0	390000	0.9	144	0	0	256	0
292	45	0	615	1	55	0	222000	0.8	141	0	0	257	0
293	60	0	320	0	35	0	133000	1.4	139	1	0	258	0
294	52	0	190	1	38	0	382000	1	140	1	1	258	0
295	63	1	103	1	35	0	179000	0.9	136	1	1	270	0
296	62	0	61	1	38	1	155000	1.1	143	1	1	270	0
297	55	0	1820	0	38	0	270000	1.2	139	0	0	271	0
298	45	0	2060	1	60	0	742000	0.8	138	0	0	278	0
299	45	0	2413	0	38	0	140000	1.4	140	1	1	280	0
300	50	0	196	0	45	0	395000	1.6	136	1	1	285	0

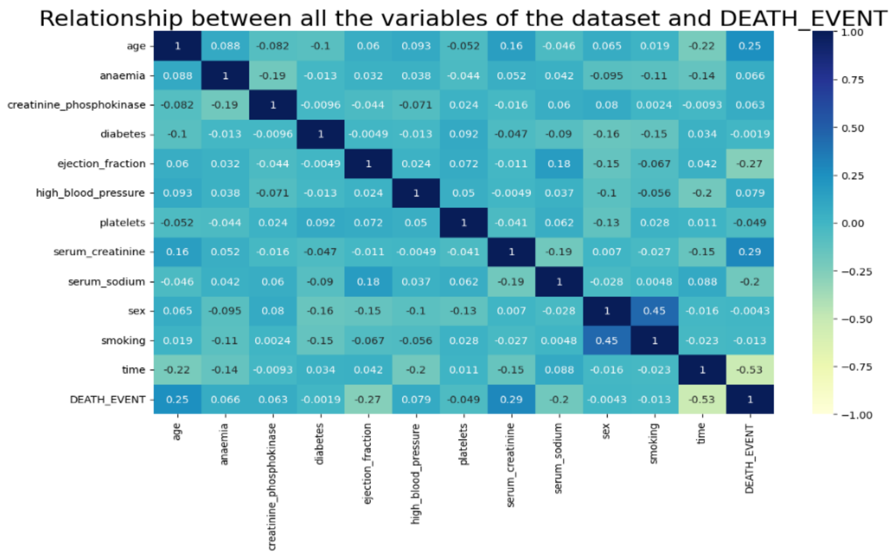
3. Exploratory Data Analysis

Getting Insights About the Dataset

```python
plt.figure(figsize=(13,7))
plt.scatter(platelets, age, c = data["DEATH_EVENT"], s=100, alpha=0.8)
plt.xlabel("Platelets", fontsize=20)
plt.ylabel("Age",fontsize=20)
plt.title("Visualizing the unbalanced data", fontsize=22)
plt.show()
```



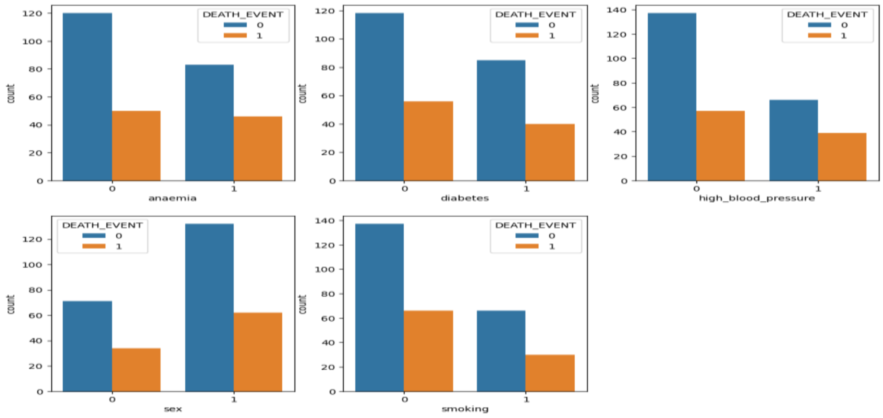

```python
plt.figure(figsize=(13,7))
sns.heatmap(data.corr(), vmin=-1, vmax=1, cmap="YlGnBu", annot=True)
plt.title("Relationship between all the variables of the dataset and DEATH_EVENT", fontsize = 22)
plt.show()
```



Data visualization

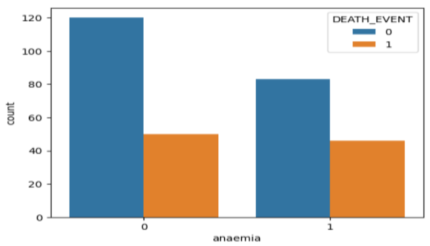
```
categorical_data = ["anaemia","diabetes","high_blood_pressure","sex","smoking"]
continuous_data = ["age","creatinine_phosphokinase","ejection_fraction","platelets","serum_creatinine","serum_sodium","time"]
```

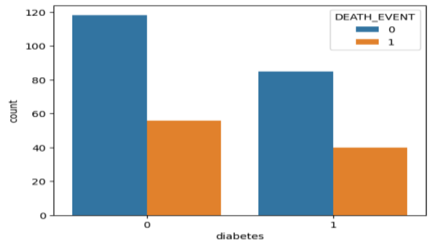
```
plt.figure(figsize=(13,10))
for i,cat in enumerate(categorical_data):
    plt.subplot(2,3,i+1)
    sns.countplot(data = data, x= cat, hue = "DEATH_EVENT")
plt.show()
```



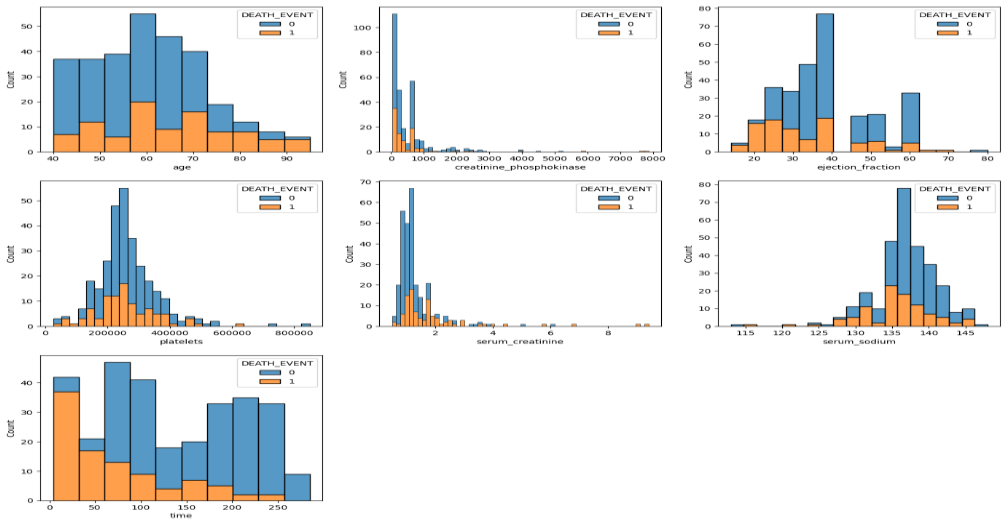

```
plt.figure(figsize=(13,10))
plt.subplot(2,2,1)
sns.countplot(data = data, x= 'anaemia', hue = "DEATH_EVENT")
plt.subplot(2,2,4)
sns.countplot(data = data, x= 'diabetes', hue = "DEATH_EVENT")
```

```
<Axes: xlabel='diabetes', ylabel='count'>
```



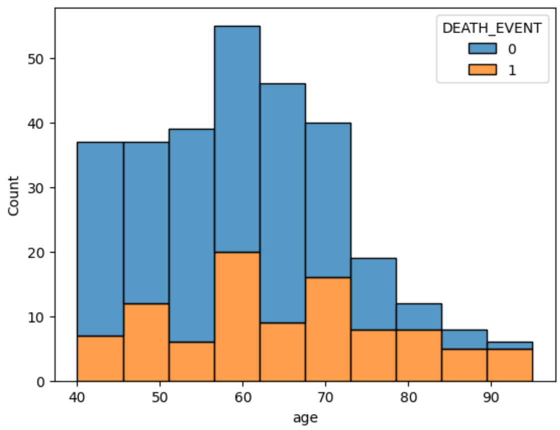



```
plt.figure(figsize=(17,15))
for j,con in enumerate(continuous_data):
    plt.subplot(3,3,j+1)
    sns.histplot(data = data, x= con, hue = "DEATH_EVENT", multiple="stack")
plt.show()
```

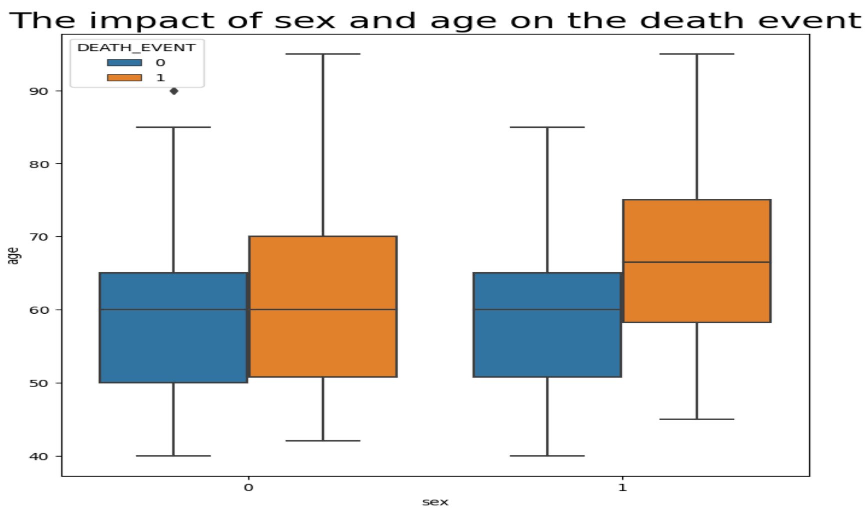



```
sns.histplot(data = data, x= 'age', hue = "DEATH_EVENT", multiple="stack")
```

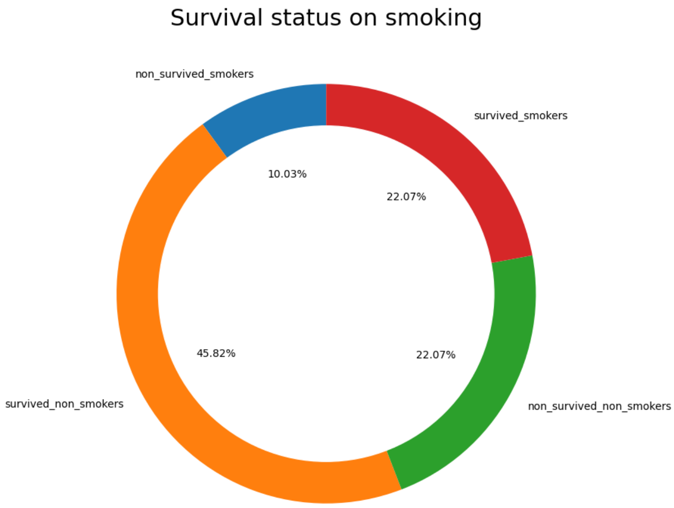
```
<Axes: xlabel='age', ylabel='Count'>
```




```
plt.figure(figsize=(8,8))
sns.boxplot(data=data, x="sex", y="age", hue="DEATH_EVENT")
plt.title("The impact of sex and age on the death event", fontsize=22)
plt.show()
```

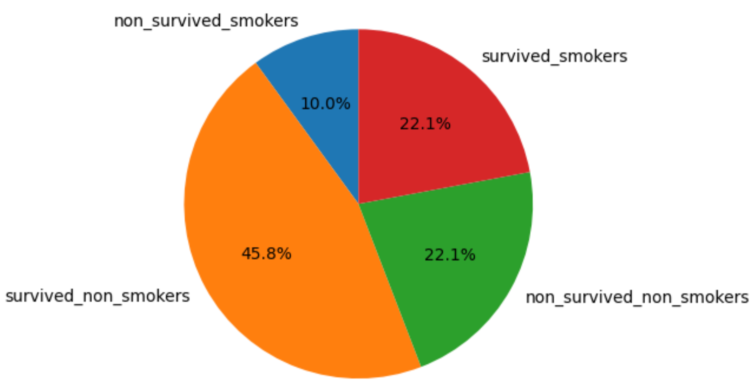


```python
smokers = data[data["smoking"]==1]
non_smokers = data[data["smoking"]==0]

non_survived_smokers = smokers[smokers["DEATH_EVENT"]==1]
survived_non_smokers = non_smokers[non_smokers["DEATH_EVENT"]==0]
non_survived_non_smokers = non_smokers[non_smokers["DEATH_EVENT"]==1]
survived_smokers = smokers[smokers["DEATH_EVENT"]==0]

smoking_data = [len(non_survived_smokers), len(survived_non_smokers),len(non_survived_non_smokers),len(survived_smokers)]
smoking_labels = ["non_survived_smokers", "survived_non_smokers", "non_survived_non_smokers", "survived_smokers"]

plt.figure(figsize=(9,9))
plt.pie(smoking_data, labels = smoking_labels, autopct='%.2f%%', startangle=90)
circle = plt.Circle((0,0), 0.8, color="white")
p = plt.gcf()
p.gca().add_artist(circle)
plt.title("Survival status on smoking", fontsize=22)
plt.show()
```

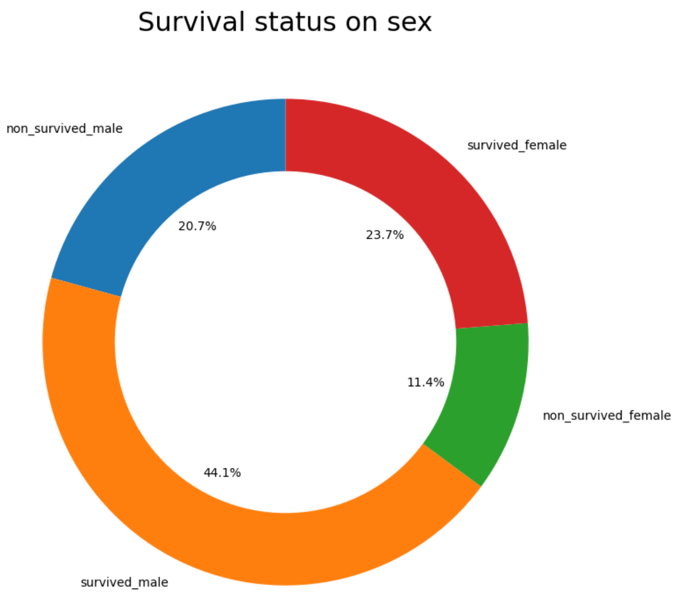



```python
plt.pie(smoking_data, labels = smoking_labels, autopct='%.1f%%', startangle=90)
```

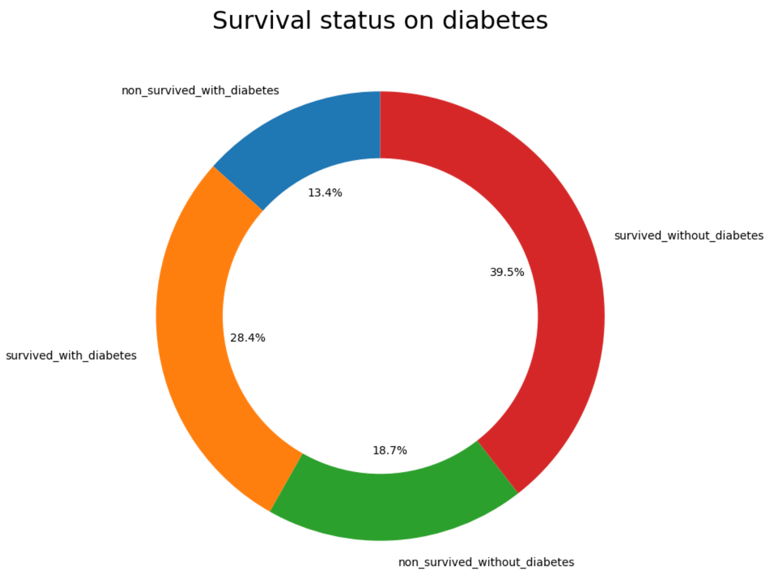
```
([<matplotlib.patches.Wedge at 0x7d7e9a710310>,
  <matplotlib.patches.Wedge at 0x7d7e9a710220>,
  <matplotlib.patches.Wedge at 0x7d7e9a711030>,
  <matplotlib.patches.Wedge at 0x7d7e9a7116c0>],
 [Text(-0.34101771647194173, 1.045804435376071, 'non_survived_smokers'),
  Text(-0.9658242801224177, -0.5264821553728236, 'survived_non_smokers'),
  Text(0.9602393822474176, -0.5366007163441904, 'non_survived_non_smokers'),
  Text(0.7031236200758868, 0.8459415907090634, 'survived_smokers')],
 [Text(-0.18600966353015, 0.5704387829324022, '10.0%'),
  Text(-0.5268132437031369, -0.28717208474881284, '45.8%'),
  Text(0.5237669357713186, -0.292691299824103 8, '22.1%'),
  Text(0.3835219745868473, 0.4614226858413072, '22.1%')])
```



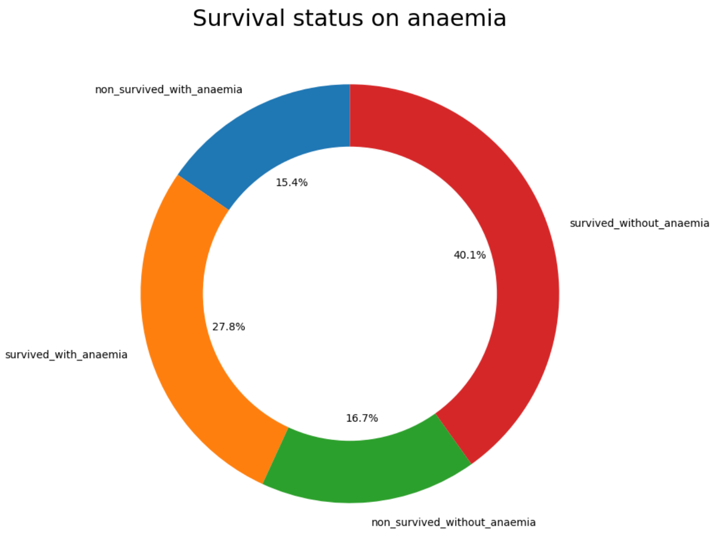
```
male = data[data["sex"]==1]
female = data[data["sex"]==0]

non_survived_male = male[male["DEATH_EVENT"]==1]
survived_male = male[male["DEATH_EVENT"]==0]
non_survived_female = female[female["DEATH_EVENT"]==1]
survived_female = female[female["DEATH_EVENT"]==0]

sex_data = [len(non_survived_male), len(survived_male), len(non_survived_female),len(survived_female)]
sex_labels = ["non_survived_male","survived_male","non_survived_female","survived_female"]

plt.figure(figsize=(9,9))
plt.pie(sex_data, labels = sex_labels, autopct='%.1f%%', startangle=90)
circle = plt.Circle((0,0), 0.7, color="white")
p = plt.gcf()
p.gca().add_artist(circle)
plt.title("Survival status on sex", fontsize=22)
plt.show()
```



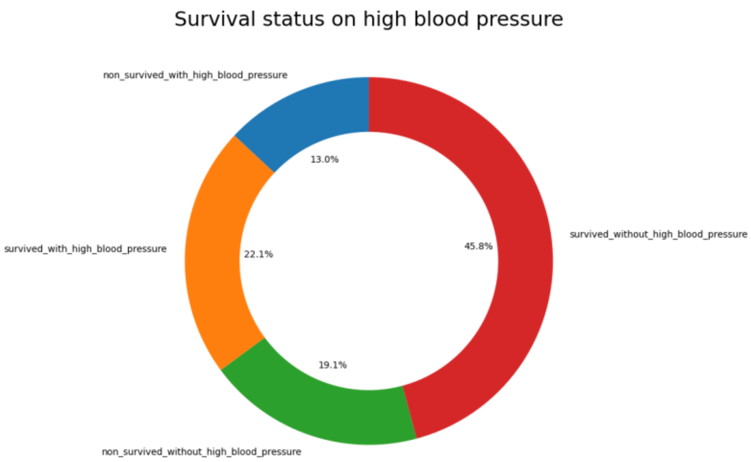

```
with_diabetes = data[data["diabetes"]==1]
without_diabetes = data[data["diabetes"]==0]

non_survived_with_diabetes = with_diabetes[with_diabetes["DEATH_EVENT"]==1]
survived_with_diabetes = with_diabetes[with_diabetes["DEATH_EVENT"]==0]
non_survived_without_diabetes = without_diabetes[without_diabetes["DEATH_EVENT"]==1]
survived_without_diabetes = without_diabetes[without_diabetes["DEATH_EVENT"]==0]

diabetes_data = [len(non_survived_with_diabetes), len(survived_with_diabetes), len(non_survived_without_diabetes), \
                 len(survived_without_diabetes)]
diabetes_labels = ["non_survived_with_diabetes","survived_with_diabetes","non_survived_without_diabetes",\
                   "survived_without_diabetes"]

plt.figure(figsize=(9,9))
plt.pie(diabetes_data, labels = diabetes_labels, autopct='%.1f%%', startangle=90)
circle = plt.Circle((0,0), 0.7, color="white")
p = plt.gcf()
p.gca().add_artist(circle)
plt.title("Survival status on diabetes", fontsize=22)
plt.show()
```




```
with_anaemia = data[data["anaemia"]==1]
without_anaemia = data[data["anaemia"]==0]

non_survived_with_anaemia = with_anaemia[with_anaemia["DEATH_EVENT"]==1]
survived_with_anaemia = with_anaemia[with_anaemia["DEATH_EVENT"]==0]
non_survived_without_anaemia = without_anaemia[without_anaemia["DEATH_EVENT"]==1]
survived_without_anaemia = without_anaemia[without_anaemia["DEATH_EVENT"]==0]

anaemia_data = [len(non_survived_with_anaemia), len(survived_with_anaemia), len(non_survived_without_anaemia), \
                len(survived_without_anaemia)]
anaemia_labels = ["non_survived_with_anaemia","survived_with_anaemia","non_survived_without_anaemia",\
                  "survived_without_anaemia"]

plt.figure(figsize=(9,9))
plt.pie(anaemia_data, labels = anaemia_labels, autopct='%.1f%%', startangle=90)
circle = plt.Circle((0,0), 0.7, color="white")
p = plt.gcf()
p.gca().add_artist(circle)
plt.title("Survival status on anaemia", fontsize=22)
plt.show()
```



```
with_high_blood_pressure = data[data["high_blood_pressure"]==1]
without_high_blood_pressure = data[data["high_blood_pressure"]==0]

non_survived_with_high_blood_pressure = with_high_blood_pressure[with_high_blood_pressure["DEATH_EVENT"]==1]
survived_with_high_blood_pressure = with_high_blood_pressure[with_high_blood_pressure["DEATH_EVENT"]==0]
non_survived_without_high_blood_pressure = without_high_blood_pressure[without_high_blood_pressure["DEATH_EVENT"]==1]
survived_without_high_blood_pressure = without_high_blood_pressure[without_high_blood_pressure["DEATH_EVENT"]==0]

high_blood_pressure_data = [len(non_survived_with_high_blood_pressure), len(survived_with_high_blood_pressure), \
                            len(non_survived_without_high_blood_pressure), len(survived_without_high_blood_pressure)]

high_blood_pressure_labels = ["non_survived_with_high_blood_pressure","survived_with_high_blood_pressure",\
                   "non_survived_without_high_blood_pressure","survived_without_high_blood_pressure"]

plt.figure(figsize=(9,9))
plt.pie(high_blood_pressure_data, labels = high_blood_pressure_labels, autopct='%.1f%%', startangle=90)
circle = plt.Circle((0,0), 0.7, color="white")
p = plt.gcf()
p.gca().add_artist(circle)
plt.title("Survival status on high blood pressure", fontsize=22)
plt.show()
```



4. Methods Used

4.1 Algorithm

1.Logistic Regression: A commonly used statistical technique for binary classification tasks like predicting heart strokes.

2.Decision Trees: Effective for capturing non-linear relationships and interactions between features.

3. Support Vector Machine: It is a supervised machine learning algorithm that can be used for classification or regression tasks. The primary goal of an SVM is to find a hyperplane in a high-dimensional space that best separates data points of one class from another.

4. K-Nearest Neighbors (KNN): It is a supervised machine learning algorithm used for both classification and regression tasks. It's a type of instance-based learning or lazy learning, meaning it doesn't explicitly build a model but memorizes the training dataset.

5. Random Forest: It is a powerful and versatile algorithm that is widely used in practice due to its good performance and ease of use. It is effective in handling high-dimensional data, capturing complex r relationships, and dealing with noisy datasets.

6. Naive Bayes: It is a family of probabilistic classification algorithms based on Bayes' theorem, with the "naive" assumption of independence between features. These classifiers are particularly popular for text classification tasks, such as spam filtering and sentiment analysis.

5. Results and Evaluation

5.1 Results

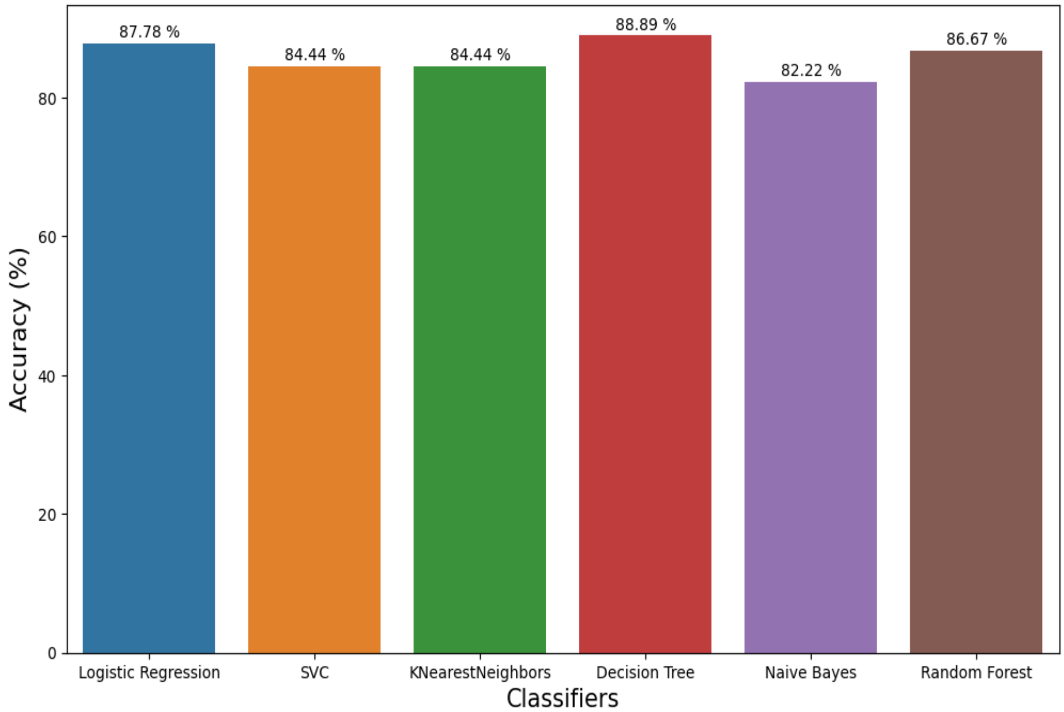


Fig 1. Comparison of Accuracy for Different Classifiers

6. Conclusion

1.It seemed like both BMI and Age were positively correlated, though the association was not strong.

2.Older patient was more likely to suffer a stroke than a younger patient.

3.Higher BMI does not increase the stroke risk.

4.Diabetes is one of the risk factors for stroke occurrence and prediabetes patients have an increased risk of stroke.

5.Higher proportion of patients who suffered from hypertension or heart disease experienced a stroke, all else being equal.

6.Regardless of patient’s gender, and where they stayed, they have the same likelihood to experience stroke.

7.Work type variable was highly associated with age. 8.Marital status variable was highly associated with age.

7. References

[1] E. S. Donkor, Stroke in the 21st century: a snapshot of the burden, epidemiology, and quality of life (2018), Stroke research and treatment.

[2] W. Johnson, O. Onuma, M. Owolabi and S. Sachdev, Stroke: a global response is needed (2016), Bulletin of the World Health Organization.

Annexure 1:

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

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score
```

1. Importing and Exploring the dataset:

+ Code + Text

```
[18]
data=pd.read_csv("/content/heart_failure_clinical_records_dataset (1).csv")
```

```
[19] type(data)
```

```
[ ] data.shape
```

Note: We have 299 lines and 13 columns

```
[ ] data.columns
```

```
[ ] data.head(10)
```

```
[ ] categorical_variables = data[["anaemia","diabetes","high_blood_pressure","sex","smoking"]]
    continuous_variables = data[["age","creatinine_phosphokinase","ejection_fraction","platelets","serum_creatinine",\
                                 "serum_sodium","time"]]
```

```
[ ] type(categorical_variables)
```

Notes for categorical data: Anaemia : 0 means that the person does not have anaemia, if 1 it does.

Diabetes : 0 means that the person does not have diabetes, if 1 it does.

High_blood_pressure : 0 means that the person does not have high_blood_pressure, if 1 it does.

Smoking : 0 means that the person does not smoke, if 1 it smokes.

Sex : 0 for female, 1 for male.

DEATH_EVENT : 0 means heart failure is not the cause of the death, if 1 it is.

```
[ ] pd.set_option('display.max_rows', 300)
    data.isna().sum()
```

```
[ ] data.isnull().sum()
```

Note: We can deduce that the dataset does not contain null values

```
[ ] """
    describe() function helps us with the descriptive statistics.
    For example we have the minimum age is 40 and the maximum is 95 with a mean of 60.834,
    for the same variable, we have the median is 60, standard deviation is 11.895 ...
    """

    continuous_variables.describe()
```

```
[ ] data.groupby("DEATH_EVENT").count()
```

```python
age = data[["age"]]
platelets = data[["platelets"]]
```

```python
type(data[['age']])
```

```python
plt.figure(figsize=(13,7))
plt.scatter(platelets, age, c = data["DEATH_EVENT"], s=100, alpha=0.8)
plt.xlabel("Platelets", fontsize=20)
plt.ylabel("Age",fontsize=20)
plt.title("Visualizing the unbalanced data", fontsize=22)
plt.show()
```

```python
plt.figure(figsize=(13,7))
sns.heatmap(data.corr(), vmin=-1, vmax=1, cmap="YlGnBu", annot=True)
plt.title("Relationship between all the variables of the dataset and DEATH_EVENT", fontsize = 22)
plt.show()
```

2. Data visualization:

```python
categorical_data = ["anaemia","diabetes","high_blood_pressure","sex","smoking"]
continuous_data = ["age","creatinine_phosphokinase","ejection_fraction","platelets","serum_creatinine","serum_sodium","time"]
```

```python
plt.figure(figsize=(13,10))
for i,cat in enumerate(categorical_data):
    plt.subplot(2,3,i+1)
    sns.countplot(data = data, x= cat, hue = "DEATH_EVENT")
plt.show()
```

```python
plt.figure(figsize=(13,10))
plt.subplot(2,2,1)
sns.countplot(data = data, x= 'anaemia', hue = "DEATH_EVENT")
plt.subplot(2,2,4)
sns.countplot(data = data, x= 'diabetes', hue = "DEATH_EVENT")
```

```python
for i,cat in enumerate(categorical_data):
    print(i, cat)
```

```python
plt.figure(figsize=(17,15))
for j,con in enumerate(continuous_data):
    plt.subplot(3,3,j+1)
    sns.histplot(data = data, x= con, hue = "DEATH_EVENT", multiple="stack")
plt.show()
```

```python
for i,cat in enumerate(continuous_data):
    print(i, cat)
```

```python
sns.histplot(data = data, x= 'age', hue = "DEATH_EVENT", multiple="stack")
```

```python
plt.figure(figsize=(8,8))
sns.boxplot(data=data, x="sex", y="age", hue="DEATH_EVENT")
plt.title("The impact of sex and age on the death event", fontsize=22)
plt.show()
```

```python
smokers = data[data["smoking"]==1]
non_smokers = data[data["smoking"]==0]

non_survived_smokers = smokers[smokers["DEATH_EVENT"]==1]
survived_non_smokers = non_smokers[non_smokers["DEATH_EVENT"]==0]
non_survived_non_smokers = non_smokers[non_smokers["DEATH_EVENT"]==1]
```

```
survived_smokers = smokers[smokers["DEATH_EVENT"]==0]

smoking_data = [len(non_survived_smokers), len(survived_non_smokers),len(non_survived_non_smokers),len(survived_smokers)]
smoking_labels = ["non_survived_smokers", "survived_non_smokers", "non_survived_non_smokers", "survived_smokers"]

plt.figure(figsize=(9,9))
plt.pie(smoking_data, labels = smoking_labels, autopct='%.2f%%', startangle=90)
circle = plt.Circle((0,0), 0.8, color="white")
p = plt.gcf()
p.gca().add_artist(circle)
plt.title("Survival status on smoking", fontsize=22)
plt.show()
```

```
plt.pie(smoking_data, labels = smoking_labels, autopct='%.1f%%', startangle=90)
```

```
type(non_smokers)
```

```
smokers[smokers["DEATH_EVENT"]==1]
```

```
(len(non_survived_smokers)/299)*100
```

```
len(smokers[smokers["DEATH_EVENT"]==1])
```

```
smoking_data
```

```
smoking_labels
```

```
male = data[data["sex"]==1]
female = data[data["sex"]==0]

non_survived_male = male[male["DEATH_EVENT"]==1]
survived_male = male[male["DEATH_EVENT"]==0]
non_survived_female = female[female["DEATH_EVENT"]==1]
survived_female = female[female["DEATH_EVENT"]==0]

sex_data = [len(non_survived_male), len(survived_male), len(non_survived_female),len(survived_female)]
sex_labels = ["non_survived_male","survived_male","non_survived_female","survived_female"]

plt.figure(figsize=(9,9))
plt.pie(sex_data, labels = sex_labels, autopct='%.1f%%', startangle=90)
circle = plt.Circle((0,0), 0.7, color="white")
p = plt.gcf()
p.gca().add_artist(circle)
plt.title("Survival status on sex", fontsize=22)
```

```
plt.show()
```

```
with_diabetes = data[data["diabetes"]==1]
without_diabetes = data[data["diabetes"]==0]

non_survived_with_diabetes = with_diabetes[with_diabetes["DEATH_EVENT"]==1]
survived_with_diabetes = with_diabetes[with_diabetes["DEATH_EVENT"]==0]
non_survived_without_diabetes = without_diabetes[without_diabetes["DEATH_EVENT"]==1]
survived_without_diabetes = without_diabetes[without_diabetes["DEATH_EVENT"]==0]

diabetes_data = [len(non_survived_with_diabetes), len(survived_with_diabetes), len(non_survived_without_diabetes), \
                len(survived_without_diabetes)]
diabetes_labels = ["non_survived_with_diabetes","survived_with_diabetes","non_survived_without_diabetes",\
                  "survived_without_diabetes"]

plt.figure(figsize=(9,9))
plt.pie(diabetes_data, labels = diabetes_labels, autopct='%.1f%%', startangle=90)
circle = plt.Circle((0,0), 0.7, color="white")
p = plt.gcf()
p.gca().add_artist(circle)
plt.title("Survival status on diabetes", fontsize=22)
```

```
plt.show()
```

```
with_anaemia = data[data["anaemia"]==1]
without_anaemia = data[data["anaemia"]==0]

non_survived_with_anaemia = with_anaemia[with_anaemia["DEATH_EVENT"]==1]
survived_with_anaemia = with_anaemia[with_anaemia["DEATH_EVENT"]==0]
non_survived_without_anaemia = without_anaemia[without_anaemia["DEATH_EVENT"]==1]
survived_without_anaemia = without_anaemia[without_anaemia["DEATH_EVENT"]==0]

anaemia_data = [len(non_survived_with_anaemia), len(survived_with_anaemia), len(non_survived_without_anaemia), \
                len(survived_without_anaemia)]
anaemia_labels = ["non_survived_with_anaemia","survived_with_anaemia","non_survived_without_anaemia",\
                  "survived_without_anaemia"]

plt.figure(figsize=(9,9))
plt.pie(anaemia_data, labels = anaemia_labels, autopct='%.1f%%', startangle=90)
circle = plt.Circle((0,0), 0.7, color="white")
p = plt.gcf()
p.gca().add_artist(circle)
plt.title("Survival status on anaemia", fontsize=22)
plt.show()
```

```
with_high_blood_pressure = data[data["high_blood_pressure"]==1]
without_high_blood_pressure = data[data["high_blood_pressure"]==0]

non_survived_with_high_blood_pressure = with_high_blood_pressure[with_high_blood_pressure["DEATH_EVENT"]==1]
survived_with_high_blood_pressure = with_high_blood_pressure[with_high_blood_pressure["DEATH_EVENT"]==0]
non_survived_without_high_blood_pressure = without_high_blood_pressure[without_high_blood_pressure["DEATH_EVENT"]==1]
survived_without_high_blood_pressure = without_high_blood_pressure[without_high_blood_pressure["DEATH_EVENT"]==0]

high_blood_pressure_data = [len(non_survived_with_high_blood_pressure), len(survived_with_high_blood_pressure), \
                            len(non_survived_without_high_blood_pressure), len(survived_without_high_blood_pressure)]

high_blood_pressure_labels = ["non_survived_with_high_blood_pressure","survived_with_high_blood_pressure",\
                   "non_survived_without_high_blood_pressure","survived_without_high_blood_pressure"]

plt.figure(figsize=(9,9))
plt.pie(high_blood_pressure_data, labels = high_blood_pressure_labels, autopct='%.1f%%', startangle=90)
circle = plt.Circle((0,0), 0.7, color="white")
p = plt.gcf()
p.gca().add_artist(circle)
plt.title("Survival status on high blood pressure", fontsize=22)
plt.show()
```

3. Data modeling & prediction using continuous data:

```
[ ] x = data[["age","creatinine_phosphokinase","ejection_fraction","serum_creatinine","serum_sodium","time"]]
    y = data["DEATH_EVENT"]
```

```
[ ] x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=2)
```

```
[ ] scaler = StandardScaler()
    x_train_scaled = scaler.fit_transform(x_train)
    x_test_scaled = scaler.transform(x_test)
```

```
accuracy_list = [] # A list to save all the values from different models accuracy for comparaison
```

3.1 Logistic Regression

```
lr_model = LogisticRegression()
lr_model.fit(x_train_scaled, y_train)
lr_prediction = lr_model.predict(x_test_scaled)
lr_accuracy = (round(accuracy_score(lr_prediction, y_test), 4) * 100) #percentage
```

```
[ ] accuracy_list.append(lr_accuracy)
```

3.2 Support Vector Machine

```
[ ] svc_model = SVC()
    svc_model.fit(x_train_scaled, y_train)
    svc_prediction = svc_model.predict(x_test_scaled)
    svc_accuracy = (round(accuracy_score(svc_prediction, y_test), 4) * 100) #percentage
    accuracy_list.append(svc_accuracy)
```

3.3 KNearestNeighbor:

```
knn_list = []
for k in range(1,50):
    knn_model = KNeighborsClassifier(n_neighbors=k)
    knn_model.fit(x_train_scaled, y_train)
    knn_prediction = knn_model.predict(x_test_scaled)
    knn_accuracy = (round(accuracy_score(knn_prediction, y_test), 4) * 100)
    knn_list.append(knn_accuracy)
k = np.arange(1,50)
plt.plot(k, knn_list)
```

```
knn_model = KNeighborsClassifier(n_neighbors=6)
knn_model.fit(x_train_scaled, y_train)
knn_prediction = knn_model.predict(x_test_scaled)
knn_accuracy = (round(accuracy_score(knn_prediction, y_test), 4) * 100) #percentage
accuracy_list.append(knn_accuracy)
```

Decison Tree Classifier

```
dt_model = DecisionTreeClassifier(criterion="entropy", max_depth=2)
dt_model.fit(x_train_scaled, y_train)
dt_prediction = dt_model.predict(x_test_scaled)
dt_accuracy = (round(accuracy_score(dt_prediction, y_test), 4) * 100) #percentage
accuracy_list.append(dt_accuracy)
```

Naive Bayes

```
nb_model = GaussianNB()
nb_model.fit(x_train_scaled, y_train)
nb_prediction = nb_model.predict(x_test_scaled)
nb_accuracy = (round(accuracy_score(nb_prediction, y_test), 4) * 100) #percentage
accuracy_list.append(nb_accuracy)
```

3.6 Random Forest Classifier

```
[ ] rf_model = RandomForestClassifier()
    rf_model.fit(x_train_scaled, y_train)
    rf_prediction = rf_model.predict(x_test_scaled)
    rf_accuracy = (round(accuracy_score(rf_prediction, y_test), 4) * 100) #percentage
    accuracy_list.append(rf_accuracy)
```

```
[ ] accuracy_list
```

```
[ ] models = ["Logistic Regression","SVC","KNearestNeighbors","Decision Tree","Naive Bayes","Random Forest"]
```

```
plt.figure(figsize=(12,7))
ax = sns.barplot(x = models, y = accuracy_list)
plt.xlabel("Classifiers", fontsize=15)
plt.ylabel("Accuracy (%)", fontsize=15)
for p in ax.patches:
    width = p.get_width()
    height = p.get_height()
    x = p.get_x()
    y = p.get_y()
    ax.annotate(f"{height} %", (x + width/2, y+ height*1.01), ha="center")
```

```
plt.show()
```