APPLIED DATASCIENCE PHASE - 5

CREDIT CARD FRAUD DETECTION USING DATASCIENCE

PROBLEM STATEMENT:

The problem is to develop a machine learning-based system for real-time credit card fraud detection. The goal is to create a solution that can accurately identify fraudulent transactions while minimizing false positives. This project involves

- data preprocessing
- feature engineering
- model selection
- training
- evaluation to create a robust fraud detection system.

DESIGN THINKING STEPS:

1. Data Source
2. Data Preprocessing
3. Feature Engineering
4. Model Selection
5. Model Training
6. Evaluation

DATASET EXPLANATION:

The dataset that is used with this proposed approach is a real-world dataset obtained from Kaggle . It contains transactions made by credit cards

Dataset Link: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

The raw dataset taken for the study was sorted and pre-processed for the sole intention of improving the performance of the classifiers and reducing their training and operating time

Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19	V20	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0	-1.3598	-0.0728	2.5363	1.3781	-0.3383	0.4624	0.2396	0.0987	0.3638	0.0908	-0.5516	-0.6178	-0.9914	-0.3112	1.4682	-0.4704	0.2079	0.0258	0.4039	0.2514	-0.0183	0.2778	-0.1105	0.0669	0.1285	-0.1891	0.1336	-0.0211	149.62	0
0	1.1919	0.2661	0.1665	0.4481	0.0601	-0.0824	-0.0788	0.0851	-0.2554	-0.167	1.6127	1.0652	0.489	-0.1438	0.6356	0.4639	-0.1148	-0.1784	-0.1458	-0.0691	-0.2258	-0.6387	0.1013	-0.3398	0.1672	0.1259	-0.009	0.0147	2.69	0
1	-1.3584	-1.3402	1.7732	0.3797	-0.5032	1.8005	0.7914	0.2476	-1.5147	0.2076	0.6245	0.0661	0.7172	-0.1659	2.3458	-2.8901	1.1097	-0.1214	-2.2619	0.5249	0.248	0.7717	0.9094	-0.6893	-0.3276	-0.1391	-0.0554	-0.0598	378.66	0
1	-0.9663	-0.1852	1.7929	-0.8633	-0.0103	1.2472	0.2376	0.3774	-1.387	-0.055	-0.2265	0.1782	0.5077	-0.2879	-0.6314	-1.0596	-0.6841	1.9658	-1.2326	-0.208	-0.1083	0.0053	-0.1903	-1.1756	0.6474	-0.2219	0.0627	0.0615	123.5	0
2	-1.1582	0.8777	1.5487	0.4030	-0.4072	0.0959	0.5929	-0.2705	0.8177	0.7530	-0.8228	0.5382	1.3459	-1.1197	0.1751	-0.4514	-0.237	-0.0382	0.8034	0.4095	-0.0094	0.7962	-0.1375	0.1412	-0.206	0.5022	0.2194	0.2151	69.99	0
2	-0.426	0.9606	1.1411	-0.1683	0.4209	-0.0297	0.4762	0.2603	-0.5687	-0.3714	1.3416	0.3599	-0.3581	-0.1371	0.5176	0.4073	-0.0581	0.0688	-0.0332	0.0849	-0.2083	-0.5598	-0.0264	-0.3714	-0.2328	0.1059	0.2538	0.0881	3.67	0
4	1.2296	0.141	0.0453	1.2026	0.1918	0.2727	-0.0052	0.0812	0.4649	-0.0993	-1.4169	-0.1538	-0.7511	0.1673	0.0501	-0.4436	0.0028	-0.612	-0.0456	-0.2196	-0.1677	-0.2707	-0.1541	-0.7801	0.7501	-0.2572	0.0345	0.0051	4.99	0
7	-0.6443	1.4179	1.0743	-0.4322	0.9483	0.4281	1.1206	-3.8079	0.6153	1.2493	-0.6195	0.2914	1.7573	-1.3239	0.6861	-0.0761	-1.2221	-0.3582	0.3245	-0.1567	1.9434	-1.0155	0.0575	-0.6497	-0.4153	-0.0516	-1.2069	-1.0853	40.8	0
7	-0.8943	0.2861	-0.113	-0.2715	2.6696	3.7218	0.3701	0.8511	-0.392	-0.4104	-0.7051	-0.1105	-0.2863	0.0743	-0.3288	-0.2101	-0.4998	0.1187	0.5703	0.0527	-0.0734	-0.2681	-0.2042	1.0115	0.3732	-0.3842	0.0117	0.1424	93.2	0
9	-0.3383	1.1195	1.0443	-0.2222	0.4993	-0.2468	0.6518	0.0695	-0.7367	-0.3688	1.0176	0.8363	1.0068	-0.4435	0.1502	0.7394	-0.541	0.4768	0.4517	0.2037	-0.2469	-0.6338	-0.1208	-0.385	-0.0697	0.0942	0.2462	0.0830	3.68	0
10	1.4493	-1.1763	0.9139	-1.3757	-1.9714	-0.6292	-1.4232	0.0484	-1.7204	1.6266	1.1996	-0.6714	-0.5139	-0.095	0.2309	0.0319	0.2534	0.8543	-0.2214	-0.3872	-0.0093	0.3139	0.0277	0.5005	0.2503	-0.1295	0.0429	0.0162	7.8	0
10	0.3849	0.6161	-0.0743	-0.094	2.3247	3.3705	0.4704	0.5382	-0.5589	0.3976	-0.2591	-0.1261	-0.09	0.3629	0.3289	-0.1295	-0.81	0.3599	0.7076	0.1299	0.0499	0.2384	0.0693	0.9671	-0.7673	-0.4922	0.0424	-0.0543	9.99	0
10	1.25	-1.2216	0.3839	-1.2349	-1.4854	-0.7532	-0.6894	-0.2275	-2.094	1.3237	0.2276	-0.2427	1.2054	-0.3176	0.7256	-0.8156	0.8739	-0.8478	-0.6832	-0.1028	-0.2318	-0.4833	0.0846	0.3928	0.1613	-0.355	0.0264	0.0424	121.5	0
11	1.0693	0.2877	0.8286	2.7125	-0.1784	0.3375	-0.0967	0.1159	-0.2211	0.4602	-0.7737	0.3233	-0.0111	-0.1785	-0.6556	-0.1999	0.1240	-0.9805	-0.9829	-0.1532	-0.0369	0.0744	-0.0714	0.1047	0.5482	0.1040	0.0214	0.0212	27.5	0
12	-2.7919	-0.3278	1.6417	1.7674	-0.1366	0.8076	-0.4229	-1.9071	0.7557	1.1519	0.8445	0.7929	0.3704	-0.735	0.4068	-0.3031	-0.1559	0.7782	2.2218	-1.5821	1.1516	0.2221	1.0205	0.0283	-0.2327	-0.2356	-0.1648	-0.0302	58.8	0
12	-0.7524	0.3454	2.0573	-1.4686	-1.1584	-0.0778	-0.6086	0.0036	-0.4362	0.7477	-0.794	-0.7704	1.0476	-1.0668	1.1069	1.6601	-0.2793	-0.42	0.4325	0.2634	0.4996	1.3536	-0.2568	-0.0651	-0.0391	-0.0871	-0.191	0.1293	15.99	0
12	1.1003	-0.0403	1.2673	1.2690	-0.736	0.2880	-0.5861	0.1838	0.7823	-0.268	-0.4503	0.9367	0.7083	-0.4686	0.3545	-0.2466	-0.0092	-0.5959	-0.5757	-0.1139	-0.0246	0.196	0.0138	0.1037	0.3643	-0.3823	0.0928	0.0370	12.99	0
13	-0.4389	0.9189	0.9245	-0.7272	0.9156	-0.1279	0.7076	0.0876	-0.6853	-0.738	0.3241	0.2771	0.2582	-0.2191	-0.1845	1.1437	-0.3287	0.6804	0.0254	-0.047	-0.1948	-0.6726	-0.1569	-0.8884	-0.3424	-0.049	0.0796	0.1310	0.89	0
14	-5.4013	-5.4501	1.1863	1.7362	3.0491	-1.7634	-1.5597	0.1608	1.2339	0.3451	0.9172	0.9701	-0.2666	-0.4791	-0.5266	0.472	-0.7255	0.0758	-0.4069	-2.1968	-0.5036	0.9844	2.4585	0.0421	-0.4816	-0.6213	0.3920	0.9495	46.8	0
15	1.4929	-1.0293	0.4547	-1.438	-1.5554	-0.721	-1.0807	-0.0531	-1.9787	1.6388	1.0775	-0.632	-0.417	0.0521	-0.043	-0.1664	0.3042	0.5544	0.0542	-0.3879	-0.1776	-0.1751	0.04	0.2958	0.3329	-0.2204	0.0223	0.0076	5	0
16	0.6948	-1.3618	1.0292	0.8341	-1.1912	1.3091	-0.8786	0.4452	-0.4462	0.5685	1.0191	1.2983	0.4204	-0.3727	-0.808	-2.0446	0.5156	0.6258	-1.3004	-0.1383	-0.2956	-0.572	-0.0509	-0.3042	0.072	-0.4222	0.0865	0.0635	231.71	0
17	0.9625	0.3284	-0.1715	2.1092	1.1295	1.6960	0.1077	0.5215	-1.1913	0.7244	1.6903	0.4067	-0.9364	0.9837	0.7191	-0.6022	0.4024	-1.7372	-2.0276	-0.2693	0.144	0.4024	-0.0485	-1.3719	0.3908	0.1999	0.0163	-0.0146	34.09	0
18	1.1666	0.5021	-0.0673	2.2657	0.4288	0.0894	0.2411	0.1338	-0.9692	0.9227	0.7447	-0.5314	-2.1053	1.1266	0.0903	0.4244	-0.4545	-0.0989	-0.1656	-0.3072	0.0187	-0.062	-0.1039	-0.3704	0.6032	0.1065	-0.0405	-0.0114	2.28	0
18	0.2474	0.2776	1.1854	-0.0826	-1.3144	-0.1501	-0.9464	-1.6179	1.5440	-0.8239	-0.5832	0.5249	-0.4534	0.0839	1.5552	-1.3969	0.7833	0.4366	2.1788	-0.231	1.6501	0.2004	-0.1854	0.4230	0.8205	-0.2276	0.3366	0.2504	22.75	0
22	-1.9465	-0.0449	-0.4056	-1.0131	2.9419	2.9550	-0.0631	0.9555	0.0499	0.5737	-0.0813	-0.2157	0.0441	0.0339	1.1907	0.5788	-0.9757	0.0440	0.4886	-0.2167	-0.5795	-0.7992	0.8703	0.9834	0.3212	0.1496	0.7075	0.0146	0.89	0
22	-2.0743	-0.1215	1.3220	0.4100	0.2952	-0.9595	0.5439	-0.1046	0.4756	0.1494	-0.8566	-0.1805	-0.6552	-0.2798	-0.2117	-0.3333	0.0107	-0.4885	0.5057	-0.3867	-0.4036	-0.2274	0.7424	0.3985	0.2492	0.2744	0.3599	0.2432	26.43	0
23	1.1732	0.3535	0.2839	1.1335	-0.1726	-0.9161	0.3690	-0.3273	-0.2467	-0.0461	-0.1434	0.9793	1.4922	0.1014	0.7614	-0.0146	-0.5116	-0.3251	-0.3909	0.0278	0.067	0.2278	-0.1505	0.4350	0.7248	-0.3371	0.0163	0.0300	41.88	0
23	1.3227	-0.174	0.4345	0.5760	-0.8368	-0.8311	-0.2649	-0.221	-1.0714	0.8695	-0.6445	-0.1113	0.3649	0.1719	0.7821	-1.3559	-0.2169	1.2717	-1.2406	-0.523	-0.2944	-0.3234	-0.0377	0.3471	0.5596	-0.2802	0.0423	0.0288	16	0
23	-0.4143	0.9054	1.7227	1.4734	0.0077	-0.2003	0.7402	-0.0292	-0.5934	-0.3462	-0.0121	0.7868	0.6359	-0.0863	0.0768	-1.4059	0.7755	-0.9429	0.5439	0.0973	0.0772	0.4573	-0.0385	0.6425	-0.1839	-0.2775	0.1826	0.1526	33	0
23	1.0593	-0.1753	1.2661	1.1861	-0.786	0.5784	-0.7671	0.4010	0.6995	-0.0647	1.0482	1.0056	-0.542	-0.0399	-0.2187	0.0044	-0.1936	0.0423	-0.2778	-0.178	0.0136	0.2137	0.0144	0.0029	0.2946	-0.3951	0.0814	0.0242	12.99	0
24	1.2374	0.0610	0.3805	0.7615	-0.3598	-0.4941	0.0064	-0.1339	0.4388	-0.2074	-0.9292	0.5271	0.3486	-0.1525	-0.2184	-0.1916	-0.1166	-0.6338	0.3484	-0.0664	-0.2457	-0.5309	-0.0443	0.0791	0.5091	0.2888	-0.0227	0.0118	17.28	0
25	1.1140	0.0855	0.4937	1.3357	-0.3002	-0.0108	-0.1188	0.1886	0.2056	0.0822	1.1335	0.6267	-1.4928	0.5207	-0.6746	-0.5291	0.1582	-0.3988	-0.1457	-0.2738	-0.0532	-0.0048	-0.0315	0.1980	0.5650	-0.3377	0.0290	0.0044	4.45	0
26	-0.5299	0.8738	1.3472	0.1454	0.4142	0.1002	0.7112	0.1760	-0.2867	-0.4847	0.8724	0.8516	-0.5717	0.1009	-1.5198	-0.2844	-0.3105	-0.4042	-0.8234	-0.2903	0.0469	0.2081	-0.1855	0.0010	0.0988	-0.5529	-0.0733	0.0233	6.14	0
26	-0.5299	0.8738	1.3472	0.1454	0.4142	0.1002	0.7112	0.1760	-0.2867	-0.4847	0.8724	0.8516	-0.5717	0.1009	-1.5198	-0.2844	-0.3105	-0.4042	-0.8234	-0.2903	0.0469	0.2081	-0.1855	0.0010	0.0988	-0.5529	-0.0733	0.0233	6.14	0
26	-0.5354	0.8652	1.3510	0.1475	0.4336	0.0869	0.6930	0.1797	-0.2856	-0.4825	0.8718	0.8534	-0.5718	0.1022	-1.52	-0.2859	-0.3096	-0.4039	-0.8237	-0.2833	0.0495	0.2065	-0.1871	0.0007	0.0981	-0.5535	-0.0783	0.0254	1.77	0
26	-0.5354	0.8652	1.3510	0.1475	0.4336	0.0869	0.6930	0.1797	-0.2856	-0.4825	0.8718	0.8534	-0.5718	0.1022	-1.52	-0.2859	-0.3096	-0.4039	-0.8237	-0.2833	0.0495	0.2065	-0.1871	0.0007	0.0981	-0.5535	-0.0783	0.0254	1.77	0
27	-0.246	0.4732	1.6957	0.2624	-0.0109	-0.6108	0.7934	-0.2473	0.1388	-0.401	-0.8121	-0.1835	-0.6301	-0.2862	-0.337	-0.4281	-0.0306	-0.5046	0.3551	0.0457	-0.1946	-0.3351	-0.0782	0.3927	-0.0312	0.1980	-0.1753	-0.2036	30.49	0
27	-1.4522	1.7651	0.6116	1.1768	-0.446	0.2468	-0.2576	1.0924	-0.6075	0.0471	0.7837	1.0963	-0.2681	0.7686	-0.5244	-0.8088	0.7103	-0.1184	0.9792	0.0087	0.0822	0.3257	-0.0691	0.0209	-0.0447	-0.2434	0.1491	0.1205	1.8	0
29	0.9963	-0.1226	0.5466	0.7065	0.1345	1.157	-0.2946	0.4074	0.3378	-0.4082	0.6826	1.2537	0.3919	-0.0375	0.7953	-1.6682	1.3229	-2.83	-1.4634	-0.2035	-0.0763	0.1098	0.1622	-0.5756	0.1097	0.3738	0.0505	0.0051	20.53	0
29	1.1008	0.1687	0.5171	1.3254	-0.1916	0.0195	-0.0318	0.1176	0.0176	0.0449	1.3450	1.2863	-0.2523	0.2744	-0.8104	-0.587	0.8745	-0.5505	-0.1547	-0.1901	-0.0377	0.0957	-0.0482	0.2321	0.6062	-0.3421	0.0367	0.0074	6.54	0
32	1.2490	-0.6247	-0.7106	-0.9916	1.4239	3.6829	-1.0902	0.9672	0.8505	-0.3071	-0.4562	0.2239	-0.0169	-0.2208	0.3624	0.3152	-0.5123	0.1889	0.5747	0.0979	-0.0063	0.0092	-0.1295	1.1237	0.5003	1.1955	-0.0482	0.0050	29.89	0
32	-2.0089	2.1985	0.1442	1.5943	-0.8152	0.1822	-0.6171	1.5308	-0.5868	0.1298	0.6584	1.2650	-0.2065	1.0136	-0.5494	-0.708	0.8696	-0.0957	0.3444	0.0287	0.0949	0.2949	0.0108	0.0152	0.0342	-0.2361	0.1282	0.1179	2.35	0
33	-0.6079	1.0313	1.7404	1.2321	0.4185	0.1917	0.8508	-0.1763	-0.2435	0.1846	-0.387	0.3983	0.4819	-0.3654	0.2355	-1.3478	0.5046	-0.7984	0.7597	0.2543	-0.0873	0.2583	-0.2648	0.1182	0.1735	-0.217	0.0943	-0.033	14.8	0
33	-0.9357	0.1704	2.7462	-1.078	-0.3056	0.0115	-0.2962	0.4027	-0.0405	-0.852	1.5521	0.9535	0.4154	-0.2339	0.4523	0.8443	-0.8227	0.3388	-1.2142	-0.0054	0.4012	1.0648	-0.1583	0.2955	-0.2594	0.7541	0.0466	0.0939	9.1	0
34	-0.7623	0.1277	2.6500	-1.8757	-0.9934	-0.2687	-0.4199	0.1462	-0.8696	-0.0387	-0.6861	-0.5029	1.3808	-1.0391	0.6528	1.8967	-0.4584	-0.5598	-0.2951	0.3593	0.4333	1.2604	-0.4047	0.0949	0.4809	-0.0907	0.3192	0.1356	15.99	0
34	1.1393	0.0569	0.6494	0.6730	-0.4685	-0.4192	-0.0139	-0.0724	0.0067	-0.27	-0.0026	1.1243	0.7442	-0.1884	-0.0757	-0.7092	0.3812	-1.3721	-0.2737	-0.0794	-0.1642	-0.2474	0.0594	0.4562	0.361	0.2744	-0.0025	0.0171	21.24	0
34	-0.2915	0.4455	1.2497	-1.7357	0.0857	-0.1219	0.4077	0.0953	0.8159	-1.4912	-0.8462	0.0585	-0.059	0.1519	1.9825	-0.4433	-0.3183	0.0647	0.6135	-0.0335	-0.0649	-0.1204	-0.1565	-0.8002	-0.0006	-0.8352	0.131	0.0629	19.95	0
34	0.2016	0.4974	1.3739	0.5714	-0.6307	-0.5397	-0.0763	-0.3171	0.2700	-0.4803	-0.5125	0.6809	0.0899	-0.1574	-0.5888	-0.4779	0.2253	-0.6687	-0.0236	-0.2223	0.7189	-0.1719	-0.1662	0.7764	0.8184	0.4425	0.1427	0.2192	9.99	0
35	1.3864	-0.7942	0.7782	-0.8647	-1.0641	0.3513	-1.1915	0.0526	-0.3044	0.5765	-1.6311	0.0425	6	2.0479	-0.7393	1.4562	-2.272	-0.932	1.9265	3	-0.6599	-0.273	-0.2287	-0.1235	-0.131	-0.9297	0.1813	1.1943	0.0005	3	0.0199	30.9	0
35	-1.0632	1.4189	1.0866	1.2414	0.0023	1	0.0459	0.5142	0.2412	-0.1545	0.2806	-0.5319	0.5899	0.5559	-0.1164	0.1658	9	-1.2372	0.6384	1	-0.7761	0.6928	0.2350	1	-0.0572	0.3141	5	-0.1299	0.1142	8	-0.0278	-0.2599	0.1187	5	-0.009	20.22	0
35	1.1993	0.1299	0.8635	1.0026	3	-0.7838	-0.8947	-0.0407	-0.2091	0.3924	8	-0.2482	-0.0954	1.3143	1.0916	8	-0.4093	-0.578	-0.5813	0.2123	4	-0.8932	-0.0207	-0.0726	-0.0425	0.1984	7	-0.033	1.0132	9	0.5591	0.4018	2	-0.0059	0.0179	4	0.99	0
36	-1.0049	-0.986	-0.038	3.7106	-6.632	5.1221	4.3716	3	-2.0069	-0.2787	-0.2309	0.1458	-0.0632	-0.7996	-0.342	-0.9305	0.5105	1	0.0924	3	0.8239	8	1.1904	-0.002	1.3934	1	-0.3817	0.9637	2	0.0194	4	0.5709	2	0.3332	8	0.8573	7	-0.0755	1402.95	0

Importing necessary libraries

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from pylab  import  rcParams
import warnings
warnings.filterwarnings('ignore')
```

reading dataset:

```
data=pd.read_csv('creditcard.csv')
```

PREPROCESSING STEPS:

Data Cleaning:

Identify and handle missing values, outliers and inconsistences in the

dataset to ensure the data integrity and accuracy.

Handling missing data

```
data.isnull().sum()
```

Spliting the data

```
fraud_cases=len(data[data['Class']==1])
print(' Number of Fraud Cases:',fraud_cases)
non_fraud_cases=len(data[data['Class']==0])
```

output

```
Number of Fraud Cases: 150
```

```
Number of Non Fraud Cases: 5144
```

Test set and Training set

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
scalar = StandardScaler()

X = data.drop('Class', axis=1)
y = data.Class

X = scalar.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X, y,
```

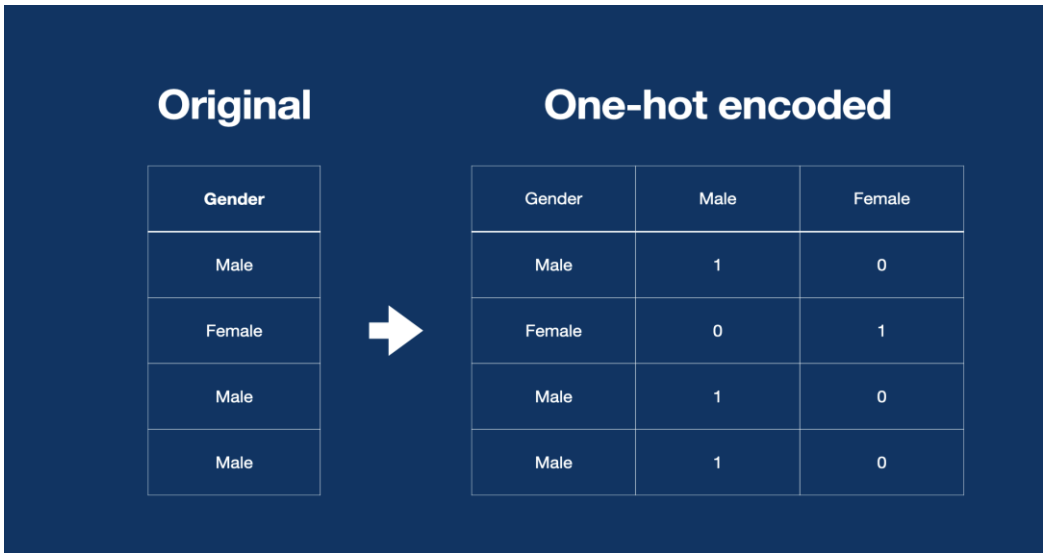
Determine which feature are relevant to project objectives and remove unnecessary variables.

ONE HOT ENCODING:

sklearn comes with a one-hot encoding tool built-in one hand encoder class. The onehot encoder class takes an array of data and can be used to one-hot encode the data.

```
from sklearn.preprocessing import OneHotEncoder
#creating instance of one-hot-encoder

encoder = OneHotEncoder(handle_unknown='ignore')
```



Data transformation:

Normalize or scale the data as needed to bring to consistent and comparable format. It includes

- Transforming units
- Aggregration
- Spatial scales

Information about valid transactions

```
print('Amount details of valid transaction')
valid_info= data[(data['Class']==0)]
```

output

```
Amount details of valid transaction
count  51440.000000
mean      94.000267
std      253.580381
min        0.000000
25%        7.680000
```

information about fraud transactions

```
print('Amount details of fraud transaction')
fraud_info = data[data['Class'] ==1]
```

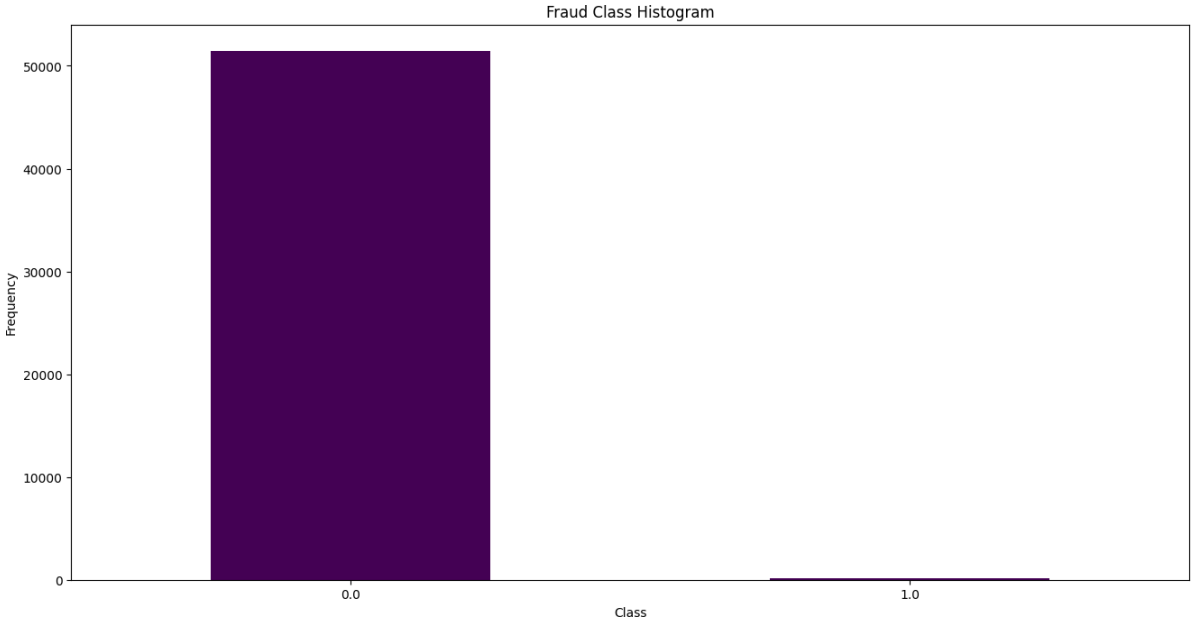
output

```
Amount details of fraud transaction
count    150.000000
mean      98.848400
std      232.056904
min        0.000000
25%        1.000000
```

Representation in Barplot

```
count_classes = pd.value_counts(data['Class'], sort = True
).sort_index()
count_classes.plot(kind = 'bar' ,rot = 0 ,colormap ='viridis')
```

```
plt.title ( "Fraud Class Histogram" )
plt.xlabel( "Class" )
```



FEATURE ENGINEERING:

This often involves a combination of

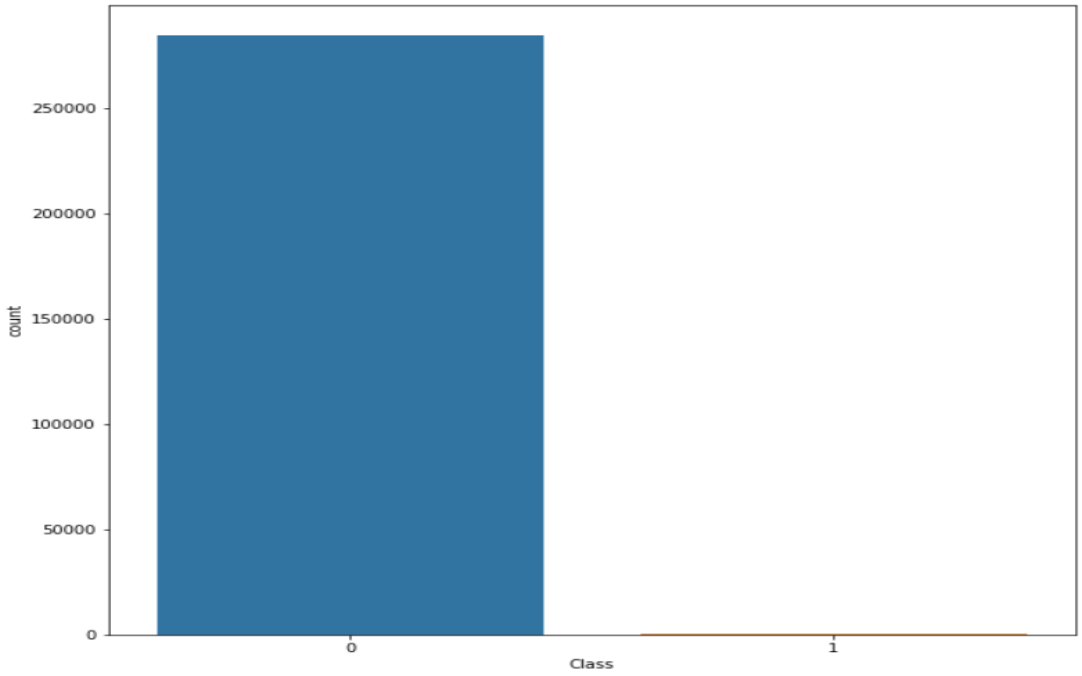
- domain expertise
- data exploration
- experimentation

to identify the most relevant and informative feature for input. Once these features have been identified, they may need to be transformed .
The process of feature engineering is crucial for achieving good performance in fraud detection and other machine learning tasks.

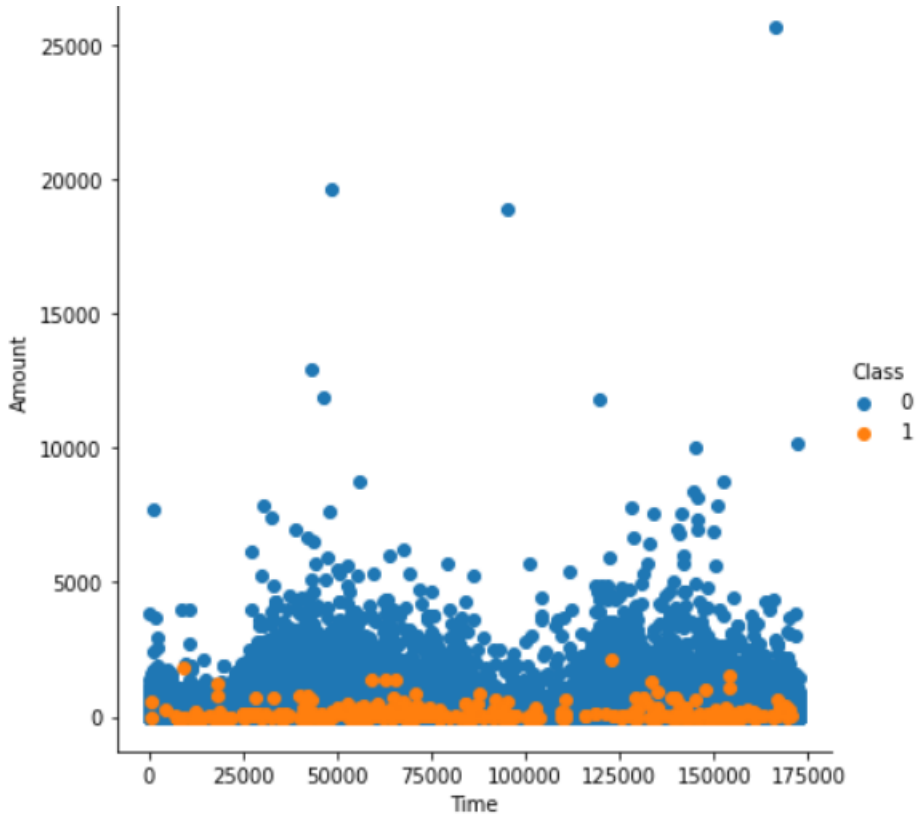
```
    def pairplot data grid(data,feature1,feature2,target);
'''
       Method to construct pairplot of the given feature wrt data
       Parameters:
           data(pd.DataFrame): Input Dataframe
           feature1(str): First Feature for Pair Plot
           feature2(str): Second Feature for Pair Plot
           target: Target or Label (y)
    '''
```

```
    sns.FacetGrid(data, hue=target, size=6).map(plt.scatter, feature1,
feature2).add_legend()
    plt.show()
```

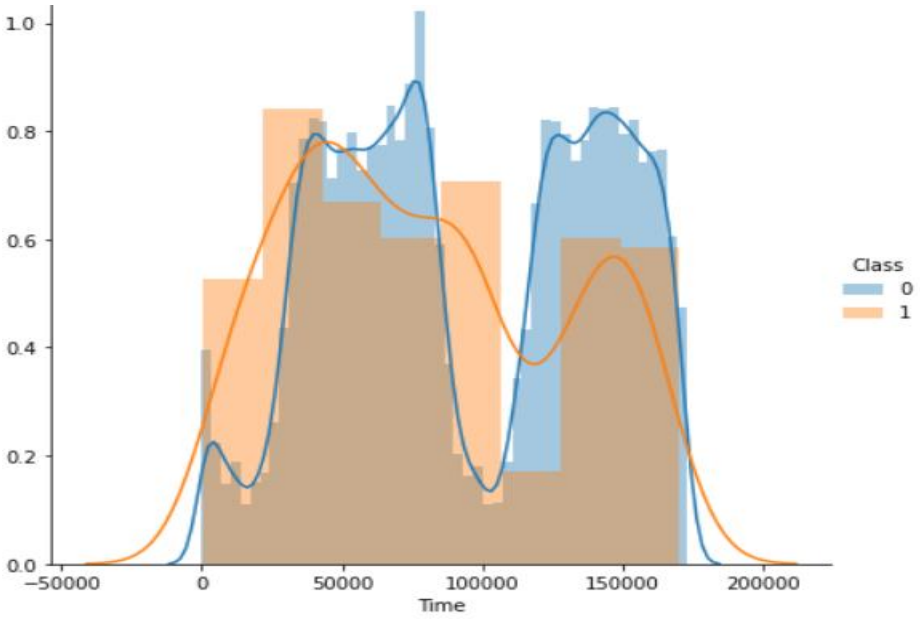
```
countplot_data(df, df.Class)
```




```
pairplot_data_grid(df, "Time", "Amount", "Class")
```




```
sns.FacetGrid(df_refine, hue="Class",
size=6).map(sns.distplot,"Time").add_legend()
plt.show()
```



MODELLING:

- Study the Feature Correlations of the given data
- Plot a Heatmap
- Run GridSearch on the data
- The plan is to train the models on the training data set which we have analyzed above and then use the testing dataset to evaluate the model performance.

- As data models need numeric input, we need to convert some of our categorical observations into numeric ones

```
plt.figure(figsize=(20,20))
df_corr = df.corr()
  sns.heatmap(df_corr)
```

```
 Create Train and Test Data in ratio 70:30
X = df.drop(labels='Class', axis=1) # Features
y = df.loc[:,'Class']               # Target Variable


X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=1, stratify=y)
```

EVALUATION:

Evaluations of the models by presenting their efficiency, the accuracies of the models will be presented in addition to any comment observed, to find the best and most suited model for detecting the fraud transactions made by credit card.

```
def grid_eval(grid_clf):
    """
        Method to Compute the best score and parameters computed by
grid search
        Parameter:
            grid_clf: The Grid Search Classifier
    """
    print("Best Score", grid_clf.best_score_)
    print("Best Parameter", grid_clf.best_params_)

def evaluation(y_test, grid_clf, X_test):
    """
        Method to compute the following:
            1. Classification Report
```

```
            2. F1-score
            3. AUC-ROC score
            4. Accuracy
        Parameters:
            y_test: The target variable test set
            grid_clf: Grid classifier selected
            X_test: Input Feature Test Set
    """
    y_pred = grid_clf.predict(X_test)
    print('CLASSIFICATION REPORT')
    print(classification_report(y_test, y_pred))

    print('AUC-ROC')
    print(roc_auc_score(y_test, y_pred))

    print('F1-Score')
    print(f1_score(y_test, y_pred))

    print('Accuracy')
    print(accuracy_score(y_test, y_pred))
```