Machine Learning: Final Project

Noah Buchanan
Zachary Mackay
Jacob Burton
Computer Science Department
University of Arkansas - Fort Smith

May 9, 2021

1 Introduction

This was an experimentation process by us to determine the best hyper-parameters for different machine learning algorithms and the interaction of those hyper-parameters potentially if any. The Machine Learning algorithms chosen were Multi-layer Perceptron(MLP), Support Vector Machine(SVM), and AdaBoost.

2 Datsets

Spam Dataset

Our first dataset was a dataset of spam or otherwise denoting so with a binary classification. The features include 48 word frequency features out of the total amount of words. 6 character frequency features out of the total amount of characters. Lastly capital_run_length_average, capital_run_length_longest, and capital_run_length_total. Capital run in this case being a sting of capital letters bounded by a non alphanumeric character, and those denote the average longest and total lengths. Below are some examples of what the data looks like, the data is split so the first 1800 records are all spam and past that all not spam records but we randomly split the data so this is not important and the data has a ratio of about 40% spam to 60% not spam. Lastly all data was normalized to be on the same scale with z-scores.

q_make	word_frec	word_frec	word_frec	word_frec	word_frec	word_frec	word_frec	word_frec	word_frec	word_frec	word_frec	word_frec	word_frec	word_frec	word_frec	word_frec	word_frec	word_frec	word_frec	word_frec	word_frec	word_frec	word_frec	word_frec	word_frec	w
0	0.64	0.64	0	0.32	0	0	0	0	0	0	0.64	0	0	0	0.32	0	1.29	1.93	0	0.96	0	0	0	0	0	0	0	0
0.21	0.28	0.5	0	0.14	0.28	0.21	0.07	0	0.94	0.21	0.79	0.65	0.21	0.14	0.14	0.07	0.28	3.47	0	1.59	0	0.43	0.43	0	0	0	0	0
0.06	0	0.71	0	1.23	0.19	0.19	0.12	0.64	0.25	0.38	0.45	0.12	0	1.75	0.06	0.06	1.03	1.36	0.32	0.51	0	1.16	0.06	0	0	0	0	0
0	0	0	0	0.63	0	0.31	0.63	0.31	0.63	0.31	0.31	0.31	0	0	0.31	0	0	3.18	0	0.31	0	0	0	0	0	0	0	0
0	0	0	0	0.63	0	0.31	0.63	0.31	0.63	0.31	0.31	0.31	0	0	0.31	0	0	3.18	0	0.31	0	0	0	0	0	0	0	0
0	0	0	0	1.85	0	0	1.85	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1.92	0	0	0	0	0.64	0.96	1.28	0	0	0	0.96	0	0.32	3.85	0	0.64	0	0	0	0	0	0	0	0
0	0	0	0	1.88	0	0	1.88	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.15	0	0.46	0	0.61	0	0.3	0	0.92	0.76	0.76	0.92	0	0	0	0	0	0.15	1.23	3.53	2	0	0	0.15	0	0	0	0	0
0.06	0.12	0.77	0	0.19	0.32	0.38	0	0.06	0	0	0.64	0.25	0	0.12	0	0	0.12	1.67	0.06	0.71	0	0.19	0	0	0	0	0	0
0	0	0	0	0	0	0.96	0	0	1.92	0.96	0	0	0	0	0	0	0.96	3.84	0	0.96	0	0	0	0	0	0	0	0
0	0	0.25	0	0.38	0.25	0.25	0	0	0	0.12	0.12	0.12	0	0	0	0	0	1.16	0	0.77	0	0	0	0	0	0	0	0
0	0.69	0.34	0	0.34	0	0	0	0	0	0	0.69	0	0	0	0.34	0	1.39	2.09	0	1.04	0	0	0	0	0	0	0	0
0	0	0	0	0.9	0	0.9	0	0	0.9	0.9	0	0.9	0	0	0	0	0	2.72	0	0.9	0	0	0	0	0	0	0	0
0	0	1.42	0	0.71	0.35	0	0.35	0	0.71	0	0.35	0	0	0	5.35	0	0	3.21	0	2.85	0	0.35	0	0	0	0	0	0
0	0.42	0.42	0	1.27	0	0.42	0	0	1.27	0	0	0	0	0	1.27	0	0	1.7	0.42	1.27	0	0	0.42	0	0	0	0	0
0	0	0	0	0.94	0	0	0	0	0	0	0	0	0	0	0	0	0	1.88	0	2.83	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2.11	0	0.7	0	0	0	0	0	0
0	0	0.55	0	1.11	0	0.18	0	0	0	0	0	0.92	0	0.18	0	0.37	0.37	3.15	0	0.92	0	0	0	0	0	0	0	0
0	0.63	0	0	1.59	0.31	0	0	0.31	0	0	0.63	0	0	1.27	0.63	0.31	3.18	2.22	0	1.91	0	0.31	0.63	0	0	0	0.31	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.05	0.07	0.1	0	0.76	0.05	0.15	0.02	0.55	0	0.1	0.47	0.02	0	0	0	0.02	0.13	2.09	0.1	1.57	0	0.05	0	0	0	0	0	0
0	0	0	0	2.94	0	0	0	0	0	0	0	0	0	0	2.94	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1.16	0	0	0	0	0	0	0.58	0	0	0	1.16	0	1.16	1.16	0	1.75	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.05	0.07	0.1	0	0.76	0.05	0.15	0.02	0.55	0	0.1	0.47	0.02	0	0	0	0.02	0.13	2.09	0.1	1.57	0	0.05	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1.66	0	0	0	0	0	0	0	0	0	0	0	3.33	0	0	0	0	3.33	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3.33	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0.65	0	0.65	0	0	0	0.65	0.65	0	0	0	0.65	1.3	0	1.3	5.22	1.3	0	0	0.65	0	0	0	0	0
1.17	0	0	0	0	0	0	0	0	0	0	1.17	0	0	0	0	1.17	0	3.52	0	0	0	0	0	0	0	0	0	0
0	0	3.03	0	0	0	0	0	0	0	0	0	0	0	0	3.03	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1.89	0.27	0	0	0	0	0	0.81	0	0	0	0.27	0	0	3.51	0	2.7	0	0	0.27	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4.83	4.83	0	0	0	0	0	0	0	0	0	0
0	0.68	0	0	0	0	0	0	0	0.68	1.36	0	0	0	0	0	0	0	2.04	0	0.68	0	0	0	0.68	0	0.68	0	0
0	0	2.56	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5.12	0	0	0	0	0	0	0	0	0	0
0	0	0	0	2.94	0	0	0	0	0	0	0	0	0	0	2.94	0	0	0	0	0	0	0	0	0	0	0	0	0

word_frec	word_frec	word_frec	word_frec	word_frec	word_frec	word_frec	word_frec	word_frec	word_frec	word_frec	word_frec	word_frec	word_frec	word_frec	word_frec	word_frec	word_frec	char_freq	char_freq	char_freq	char_freq	char_freq	capital_ru	capital_ru	capital_ru	class
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.778	0	0	3.756	61	278	1
0	0	0	0	0	0	0	0.07	0	0	0	0	0	0	0	0	0	0	0	0	0.132	0	0.372	0.18	0.048	5.114	101	1028	1
0	0	0	0	0	0	0	0	0	0	0.06	0	0	0.12	0	0.06	0.06	0	0	0.01	0.143	0	0.276	0.184	0.01	9.821	485	2259	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.137	0	0.137	0	0	3.537	40	191	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.135	0	0.135	0	0	3.537	40	191	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.223	0	0	0	0	3	15	54	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.054	0	0.164	0.054	0	1.671	4	112	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.206	0	0	0	0	2.45	11	49	1
0	0	0	0.15	0	0	0	0	0	0	0	0	0	0.3	0	0	0	0	0	0	0.271	0	0.181	0.203	0.022	9.744	445	1257	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.06	0	0	0	0	0.04	0.03	0	0.244	0.081	0	1.729	43	749	1
0	0	0	0	0	0	0	0	0	0	0.96	0	0	0	0	0	0	0	0	0	0	0	0.462	0	0	1.312	6	21	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.022	0.044	0	0.663	0	0	1.243	11	184	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.056	0	0.786	0	0	3.728	61	261	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2.083	7	25	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.102	0	0.357	0	0	1.971	24	205	1
0	0	0	0	0	0	0	1.27	0	0	0.42	0	0	0	0	0	0	0	0	0	0.063	0	0.572	0.063	0	5.659	55	249	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.428	0	0	4.652	31	107	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.975	0.37	0	35.461	95	461	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.182	0	0.455	0	0	1.32	4	70	1
0	0	0	0	0	0	0	0	0	0	1.59	0	0	0	0	0	0	0	0	0	0.275	0	0.055	0.496	0	3.509	91	186	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.729	0	0.729	0	0	3.833	9	23	1
0	0	0	0	0	0.02	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0.042	0.101	0.016	0.25	0.046	0.059	2.569	66	2259	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.404	0.404	0	0.809	0	0	4.857	12	34	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.133	0	0.667	0	0	1.131	5	69	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.44	0	0	0	0	0.196	0	0.392	0.196	0	5.466	22	82	1
0	0	0	0	0	0.02	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0.042	0.101	0.016	0.25	0.046	0.059	2.565	66	2258	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.44	0	0	0	0	0.196	0	0.392	0.196	0	5.466	22	82	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.368	0	0	2.611	12	47	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.352	0	0.352	0	0	4	11	36	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.459	0	0.091	0	0	2.687	66	129	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.17	0	0	0	0	0	0	0.886	0	0	1.966	10	59	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.352	0	2.112	0	0	3.909	11	43	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.045	0	0	0.091	0	1.39	11	89	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.302	0	1.7	5	17	1
0	1.36	0	0	0	0.68	0	1.36	0	0	0	0	0	0	0	0	0	0	0	0	0	0.185	0	0	0	3.826	30	264	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.308	0	1.543	0	0	2.777	6	25	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.436	0.436	0	0.873	0	0	4.142	12	29	1

League of Legends Dataset

Our second dataset was various game statistics from the 10 minute mark into a match for a game called League of Legends. The features include things such as team total kills, team total deaths, objectives taken by either team, the gold(in-game money) differential between the teams. Our y column prediction was a binary class denoting 1 for a blue win and 0 for a red win. This data was once again normalized by z-score. I will provide examples of what the data looks like again below. The game ID was unimportant and dropped, it is used by riot game's api to uniquely identify matches and not important.

gameId	blueWins	blueWard	blueWard	blueFirstB	blueKills	blueDeath	blueAssists	blueEliteM	blueDragon	blueHerald	blueTowe	blueTotalG	blueAvgLe	blueTotalE	blueTotalM	blueTotalJ	blueGoldD	blueExperi	blueCSPer	blueGoldPe	redWards	redWards	redFirstBl
4.52E+09	0	28	2	1	9	6	11	0	0	0	0	17210	6.6	17039	195	36	643	-8	19.5	1721	15	6	0
4.52E+09	0	12	1	0	5	5	5	0	0	0	0	14712	6.6	16265	174	43	-2908	-1173	17.4	1471.2	12	1	1
4.52E+09	0	15	0	0	7	11	4	1	1	0	0	16113	6.4	16221	186	46	-1172	-1033	18.6	1611.3	15	3	1
4.52E+09	0	43	1	0	4	5	5	1	0	1	0	15157	7	17954	201	55	-1321	-7	20.1	1515.7	15	2	1
4.44E+09	0	75	4	0	6	6	6	0	0	0	0	16400	7	18543	210	57	-1004	230	21	1640	17	2	1
4.48E+09	1	18	0	0	5	3	6	1	1	0	0	15899	7	18161	225	42	698	101	22.5	1589.9	36	5	1
4.49E+09	1	18	3	1	7	6	7	1	1	0	0	16874	6.8	16967	225	53	2411	1563	22.5	1687.4	57	1	0
4.5E+09	0	16	2	0	5	13	3	0	0	0	0	15305	6.4	16138	209	48	-2615	-800	20.9	1530.5	15	0	1
4.44E+09	0	16	3	0	7	7	8	0	0	0	0	16401	7.2	18527	189	61	-1979	-771	18.9	1640.1	15	2	1
4.51E+09	1	13	1	1	4	5	5	1	1	0	0	15057	6.8	16805	220	39	-1548	-1574	22	1505.7	16	2	0
4.45E+09	0	20	3	1	4	4	6	0	0	0	0	15474	6.6	16611	231	28	331	-1585	23.1	1547.4	15	2	0
4.45E+09	0	33	2	1	11	11	7	1	0	1	0	16695	7	18507	157	40	-1505	-635	15.7	1669.5	17	1	0
4.52E+09	1	18	1	1	7	1	11	1	1	0	0	17865	7.4	19102	238	53	3274	1659	23.8	1786.5	12	1	0
4.52E+09	0	14	3	0	4	9	1	1	0	1	0	14979	6.6	17213	210	52	-3414	-1141	21	1497.9	20	3	1
4.52E+09	1	15	3	1	4	4	4	0	0	0	0	15722	6.8	17896	224	51	-470	-187	22.4	1572.2	102	1	0
4.48E+09	0	17	1	0	3	7	3	0	0	0	0	15015	6.8	16974	209	53	-1996	-1804	20.9	1501.5	18	3	1
4.52E+09	1	14	1	1	10	2	8	0	0	0	0	19733	7.6	20862	263	56	5228	3378	26.3	1973.3	13	2	0
4.5E+09	0	43	3	0	3	7	3	1	0	1	0	14852	6.8	16888	203	54	-1975	-1345	20.3	1485.2	17	14	1
4.49E+09	1	21	4	1	5	4	11	0	0	0	0	16282	6.8	17378	213	49	882	512	21.3	1628.2	19	3	0
4.46E+09	0	11	3	0	5	9	5	0	0	0	0	14994	7	17924	188	48	-3155	-2773	18.8	1499.4	15	1	1
4.47E+09	1	14	3	1	11	6	15	1	1	0	0	18606	7.2	19254	214	45	2526	1625	21.4	1860.6	12	2	0
4.45E+09	0	13	1	0	4	13	5	0	0	0	0	15878	6.8	16945	203	43	-2835	-2821	20.3	1587.8	15	3	1
4.48E+09	0	17	2	0	4	6	3	0	0	0	0	15773	7	17562	238	56	-1254	-567	23.8	1577.3	16	3	1
4.52E+09	0	78	4	0	4	3	4	2	1	1	0	15906	6.8	19219	238	67	406	1283	23.8	1590.6	15	3	1
4.52E+09	1	33	3	0	6	9	10	0	0	0	0	16515	6.8	17223	208	45	-1020	493	20.8	1651.5	18	3	1
4.52E+09	0	17	1	0	4	8	4	0	0	0	0	15415	6.2	16047	204	44	-2472	-1067	20.4	1541.5	19	2	1
4.51E+09	1	18	1	0	8	6	9	1	1	0	0	17178	7	17854	209	44	1260	-55	20.9	1717.8	10	3	1
4.5E+09	0	19	3	1	7	4	9	0	0	0	0	17226	7	18982	228	61	1857	2015	22.8	1722.6	14	4	0
4.52E+09	0	105	5	1	8	6	13	1	1	0	0	17474	6.8	17436	217	53	1320	-413	21.7	1747.4	65	3	0
4.48E+09	0	18	5	0	3	8	2	1	1	0	0	14905	6.8	17974	233	52	-3282	-1647	23.3	1490.5	18	3	1
4.47E+09	1	22	7	1	9	4	10	1	1	0	0	18132	7	18076	228	40	2883	2111	22.8	1813.2	21	4	0
4.52E+09	1	17	4	1	5	2	6	2	1	1	0	16053	6.8	17406	218	51	1703	1696	21.8	1605.3	15	2	0
4.5E+09	0	12	3	0	6	4	7	0	0	0	0	16438	6.8	16989	203	32	608	-85	20.3	1643.8	19	2	1
4.5E+09	0	15	2	1	5	4	4	0	0	0	0	15634	6.8	16355	201	45	71	-359	20.1	1563.4	14	1	0
4.51E+09	1	40	2	0	6	4	4	0	0	0	0	16085	7.4	19076	204	56	275	872	20.4	1608.5	15	1	1
4.52E+09	0	24	3	0	3	4	6	0	0	0	0	14981	6.8	17082	235	52	-809	638	23.5	1498.1	17	2	1
4.52E+09	0	12	4	0	2	8	2	1	1	0	0	15421	7.2	19276	217	84	-2771	-1125	21.7	1542.1	14	2	1

redFirstBl	redKills	redDeaths	redAssists	redEliteM	redDragon	redHerald	redTower	redTotalG	redAvgLev	redTotalE	redTotalM	redTotalJu	redGoldDi	redExperi	redCSPer	redGoldPe
0	6	9	8	0	0	0	0	16567	6.8	17047	197	55	-643	8	19.7	1656.7
1	5	5	2	2	1	1	1	17620	6.8	17438	240	52	2908	1173	24	1762
1	11	7	14	0	0	0	0	17285	6.8	17254	203	28	1172	1033	20.3	1728.5
1	5	4	10	0	0	0	0	16478	7	17961	235	47	1321	7	23.5	1647.8
1	6	6	7	1	1	0	0	17404	7	18313	225	67	1004	-230	22.5	1740.4
1	3	5	2	0	0	0	0	15201	7	18060	221	59	-698	-101	22.1	1520.1
0	6	7	9	0	0	0	0	14463	6.4	15404	164	35	-2411	-1563	16.4	1446.3
1	13	5	11	1	1	0	0	17920	6.6	16938	157	54	2615	800	15.7	1792
1	7	7	5	2	1	1	0	18380	7.2	19298	240	53	1979	771	24	1838
0	5	4	4	0	0	0	0	16605	6.8	18379	247	43	1548	1574	24.7	1660.5
0	4	4	5	1	1	0	0	15143	7.2	18196	216	51	-331	1585	21.6	1514.3
0	11	11	9	0	0	0	0	18200	7	19142	188	52	1505	635	18.8	1820
0	1	7	1	0	0	0	0	14591	6.8	17443	240	50	-3274	-1659	24	1459.1
1	9	4	11	0	0	0	0	18393	7.2	18354	229	51	3414	1141	22.9	1839.3
0	4	4	3	0	0	0	0	16192	7	18083	242	48	470	187	24.2	1619.2
1	7	3	13	0	0	0	0	17011	7.2	18778	237	51	1996	1804	23.7	1701.1
0	2	10	2	1	1	0	0	14505	6.8	17484	210	64	-5228	-3378	21	1450.5
1	7	3	6	0	0	0	0	16827	6.8	18233	218	53	1975	1345	21.8	1682.7
0	4	5	3	2	1	1	0	15400	6.6	16866	228	52	-882	-512	22.8	1540
1	9	5	9	0	0	0	0	18149	7.4	20697	219	56	3155	2773	21.9	1814.9
0	6	11	4	0	0	0	0	16080	7	17629	225	48	-2526	-1625	22.5	1608
1	13	4	14	1	1	0	0	18713	7.2	19766	209	45	2835	2821	20.9	1871.3
1	6	4	10	1	1	0	0	17027	7	18129	231	60	1254	567	23.1	1702.7
1	3	4	3	0	0	0	0	15500	7	17936	220	64	-406	-1283	22	1550
1	9	6	13	0	0	0	0	17535	6.6	16730	197	36	1020	-493	19.7	1753.5
1	8	4	13	1	1	0	0	17887	7	17114	221	36	2472	1067	22.1	1788.7
1	6	8	3	0	0	0	0	15918	6.8	17909	233	36	-1260	55	23.3	1591.8
0	4	7	2	1	1	0	0	15369	6.6	16967	202	49	-1857	-2015	20.2	1536.9
0	6	8	11	0	0	0	0	16154	6.8	17849	223	40	-1320	413	22.3	1615.4
1	8	3	12	0	0	0	0	18187	7.4	19621	225	52	3282	1647	22.5	1818.7
0	4	9	2	0	0	0	0	15249	6.2	15965	227	36	-2883	-2111	22.7	1524.9
0	2	5	2	0	0	0	0	14350	6.4	15710	206	48	-1703	-1696	20.6	1435
1	4	6	4	1	1	0	0	15830	6.8	17074	206	44	-608	85	20.6	1583
0	4	5	4	1	1	0	0	15563	6.8	16714	222	38	-71	359	22.2	1556.3
1	4	6	5	1	1	0	0	15810	7.2	18204	223	62	-275	-872	22.3	1581
1	4	3	7	2	1	1	0	15790	6.8	16444	218	50	809	-638	21.8	1579
1	8	2	5	0	0	0	0	18192	7.4	20401	234	64	2771	1125	23.4	1819.2

3 Specifications

AdaBoost

Our first classifier was AdaBoost, AdaBoost is essentially a Random Forest classifier with some very key differences and methodologies. AdaBoost utilizes a forest of trees with only two leaves(stumps) and each stump has an amount of say determined by how correct its classification was. This then influences the next stumps amount of say by adjusting the weights assigned to the records and AdaBoost utilizes a weighted gini index to determine which feature will be split on to create the stump relative to the weights assigned. At first the weights are assigned as 1/n and change through the course of the algorithms execution.

Multi-Layer Perceptron

We also used a Mulit-Layer Perceptron as one of our classifiers. A "MLP" is a type of neural network that utilizes multiple hidden layers with nodes in those layers and weights connected the nodes between layers. How this algorithm learns is that after completing a feed forward process the algorithm will then gather the error information and change the weights accordingly by means of back propagating through all layers and nodes again. Based on the solver type the weight changes will change weight after each phase of back propagation or another method. Doing this for the multiple layers and nodes allows the discovery of patterns in the data for more in depth predictions.

Support Vector Machine

The final algorithm that we chose to use in our experimentation was a Support Vector Machine (SVM). An SVM is similar to a linear regression model, in that its objective is to find an optimal line of best fit (or optimal hyperplane in the case of 3-dimensional data or higher). A set of weights is used to calculate the weighted sum using the model input, which would produce the hyperplane. Afterwards, the error is calculated and the weights are adjusted accordingly using a method such as gradient descent. This will produce a hyperplane that separates the data, however, the optimal hyperplane (or the ultimate goal) is defined as the hyperplane that maximizes the amount of margin between the hyperplane and the first closest data point. Classification then becomes a simple task, input is fed into the model, and the predicted class becomes the corresponding side of the hyperplane where the output falls.

4 Implementation

AdaBoost Implementation

The AdaBoost hyper-parameters that were chosen to adjust were the algorithm SAMME or SAMME.R, Learning Rate, and Number of Estimators. The in-

teraction of these hyper-parameters was also measured and experimented with. However the interaction between all 3 was too costly to get conclusive results, each run took an enormous amount of time to run. Overfitting and underfitting were monitored through accuracy and loss, if the accuracy stayed unmoving and the loss began to increase it was a sure sign of overfitting and vice versa for underfitting when going to low with the hyper-parameters and it wasnt fitting the data well at all.

Multi-Layer Perceptron Implementation

For the MLP we established the default value as the a three hidden layers of 5 nodes, using an adam solver, a relu activation function and learning rate of 0.0001. Upon testing we altered the values for each of these described hyper-paramters. Our goal of this was to find if changing these values from their defaults would increase the accuracy. The answer to this question, right off the bat, is that it's a bit hard to tell. Our split method randomly splits the data so you never receive the same data every time. Also, with our data being so large and diversified a lot of variation could occur on how the algorithm takes in the inputs and learns from it. Both these reason causes each training session of the algorithm to produce wildly different results and this cause us to get an accuracy of around 74%, with slight variation, almost every time. Further testing would be needed, perhaps without allowing as much variation to occur, to allow accurate analysis.

Support Vector Machine Implementation

The SVM hyperparameters that we selected were: C-regularization parameter, kernel function, the degree of the polynomial kernel, and maximum iterations. The aim was to determine which kernel function produced the best results and why. We found that during our experimentation, the linear or rbf kernel functions consistently produced the best accuracy values given our datasets. On the otherhand, the polynomial and sigmoid kernel functions performed the worst. This was an intersting observation to us, as we did not expect the linear and rbf kernels to outperform the polynomial function, since our data had a high number of features. As for the other hyperparameters, only slight changes in performance were observed whenever they were altered. Our conclusion was that the kernel function was the most important hyperparameter to change, which would result in the greatest change in model performance. Aside from a few other interesting observations in the patterns created by the other hyper-parameters as we tweaked them, the SVM produced very consistent accuracy values for both datasets over dozens of runs.

5 Conclusions

The Classifiers and their best accuracy for the best hyperparameters per dataset:

League Dataset Accuraries

Classifier	Accuracy
AdaBoost+SAMME+300est	**73.83%**
MLP+40nodes	73.5%
SVM+1.9C+linear	72.7%

Spam Dataset Accuracies

Classifier	Accuracy
AdaBoost+SAMME+500est	**94.64%**
MLP+5nodes	77.28%
SVM+4C	94.2%