Importing Libraries

~ Author : Saurabh Kumar ~ Date : 06-Jan-23

In [2]:

```
import numpy as np
import pandas as pd
import matplotlib as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('darkgrid')
import plotly.express as px
from wordcloud import WordCloud
from scipy import signal
import scipy
#to supress warning
import warnings
warnings.filterwarnings('ignore')

#to make shell more intractive
from IPython.display import display

# setting up the chart size and background
plt.rcParams['figure.figsize'] = (16, 8)
plt.style.use('fivethirtyeight')
```

In [3]:

```
pwd
```

Out[3]:

```
'E:\\DataScience\\MachineLearning\\Sonar vs Mine Prediction logistic regression'
```

In [4]:

```
import os
os.listdir()
```

Out[4]:

```
['.ipynb_checkpoints', 'Sonar vs Mine Prediction.ipynb', 'SonarData.csv']
```

In [5]:

```
path = 'E:\\DataScience\\MachineLearning\\Sonar vs Mine Prediction logistic regression'+'\\SonarData.csv'
```

Data Collection & Data processing

In [6]:

```
#Loading the dataset to a pandas Dataframe
sonar_data = pd.read_csv(path,header=None)
```

In [7]:

```
sonar_data.head()
```

Out[7]:

	0	1	2	3	4	5	6	7	8	9	...	51	52	53	54	55	56	57
0	0.0200	0.0371	0.0428	0.0207	0.0954	0.0986	0.1539	0.1601	0.3109	0.2111	...	0.0027	0.0065	0.0159	0.0072	0.0167	0.0180	0.0084
1	0.0453	0.0523	0.0843	0.0689	0.1183	0.2583	0.2156	0.3481	0.3337	0.2872	...	0.0084	0.0089	0.0048	0.0094	0.0191	0.0140	0.0049
2	0.0262	0.0582	0.1099	0.1083	0.0974	0.2280	0.2431	0.3771	0.5598	0.6194	...	0.0232	0.0166	0.0095	0.0180	0.0244	0.0316	0.0164
3	0.0100	0.0171	0.0623	0.0205	0.0205	0.0368	0.1098	0.1276	0.0598	0.1264	...	0.0121	0.0036	0.0150	0.0085	0.0073	0.0050	0.0044
4	0.0762	0.0666	0.0481	0.0394	0.0590	0.0649	0.1209	0.2467	0.3564	0.4459	...	0.0031	0.0054	0.0105	0.0110	0.0015	0.0072	0.0048

5 rows × 61 columns

In [8]:

```
#Number of rows and columns
sonar_data.shape
```

Out[8]: (208, 61)

In [9]:

```
#decription --> statistical measures of the data
sonar_data.describe()
```

Out[9]:

	0	1	2	3	4	5	6	7	8	9	...	50
count	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	...	208.000000
mean	0.029164	0.038437	0.043832	0.053892	0.075202	0.104570	0.121747	0.134799	0.178003	0.208259	...	0.016069
std	0.022991	0.032960	0.038428	0.046528	0.055552	0.059105	0.061788	0.085152	0.118387	0.134416	...	0.012008
min	0.001500	0.000600	0.001500	0.005800	0.006700	0.010200	0.003300	0.005500	0.007500	0.011300	...	0.000000
25%	0.013350	0.016450	0.018950	0.024375	0.038050	0.067025	0.080900	0.080425	0.097025	0.111275	...	0.008425
50%	0.022800	0.030800	0.034300	0.044050	0.062500	0.092150	0.106950	0.112100	0.152250	0.182400	...	0.013900
75%	0.035550	0.047950	0.057950	0.064500	0.100275	0.134125	0.154000	0.169600	0.233425	0.268700	...	0.020825
max	0.137100	0.233900	0.305900	0.426400	0.401000	0.382300	0.372900	0.459000	0.682800	0.710600	...	0.100400

8 rows × 60 columns

In [10]:

```
sonar_data.columns
```

Out[10]:

```
Int64Index([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
            17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
            34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
            51, 52, 53, 54, 55, 56, 57, 58, 59, 60],
           dtype='int64')
```

In [11]:

```
sonar_data[60].value_counts()
```

Out[11]:

```
M    111
R     97
Name: 60, dtype: int64
```

Correlation heatmap

In [12]:

```
sonar_data.corr().style.background_gradient(cmap = 'rocket_r')
```

Out[12]:

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	1.000000	0.735896	0.571537	0.491438	0.344797	0.238921	0.260815	0.355523	0.353420	0.318276	0.344058	0.210861	0.210722
1	0.735896	1.000000	0.779916	0.606684	0.419669	0.332329	0.279040	0.334615	0.316733	0.270782	0.297065	0.194102	0.249596
2	0.571537	0.779916	1.000000	0.781786	0.546141	0.346275	0.190434	0.237884	0.252691	0.219637	0.274610	0.214807	0.258767
3	0.491438	0.606684	0.781786	1.000000	0.726943	0.352805	0.246440	0.246742	0.247078	0.237769	0.271881	0.175381	0.215754
4	0.344797	0.419669	0.546141	0.726943	1.000000	0.597053	0.335422	0.204006	0.177906	0.183219	0.231684	0.211657	0.299086
5	0.238921	0.332329	0.346275	0.352805	0.597053	1.000000	0.702889	0.471683	0.327578	0.288621	0.333570	0.344451	0.411107
6	0.260815	0.279040	0.190434	0.246440	0.335422	0.702889	1.000000	0.675774	0.470580	0.425448	0.396588	0.274432	0.365391
7	0.355523	0.334615	0.237884	0.246742	0.204006	0.471683	0.675774	1.000000	0.778577	0.652525	0.584583	0.328329	0.322951
8	0.353420	0.316733	0.252691	0.247078	0.177906	0.327578	0.470580	0.778577	1.000000	0.877131	0.728063	0.363404	0.316899
9	0.318276	0.270782	0.219637	0.237769	0.183219	0.288621	0.425448	0.652525	0.877131	1.000000	0.853140	0.485392	0.405370
10	0.344058	0.297065	0.274610	0.271881	0.231684	0.333570	0.396588	0.584583	0.728063	0.853140	1.000000	0.781164	0.646978
11	0.210861	0.194102	0.214807	0.175381	0.211657	0.344451	0.274432	0.328329	0.363404	0.485392	0.781164	1.000000	0.827106
12	0.210722	0.249596	0.258767	0.215754	0.299086	0.411107	0.365391	0.322951	0.316899	0.405370	0.646978	0.827106	1.000000
13	0.256278	0.273170	0.291724	0.286708	0.359062	0.396233	0.409576	0.387114	0.329659	0.345684	0.460504	0.467793	0.756758
14	0.304878	0.307599	0.285663	0.278529	0.318059	0.367908	0.411692	0.391514	0.299575	0.294699	0.356516	0.281758	0.530892
15	0.239079	0.261844	0.237017	0.248245	0.328725	0.353783	0.363086	0.322237	0.241819	0.242869	0.304069	0.245383	0.465977
16	0.137845	0.152170	0.201093	0.223203	0.326477	0.293190	0.250024	0.140912	0.100146	0.121264	0.237679	0.250611	0.438185
17	0.041817	0.042870	0.120587	0.194992	0.299266	0.235778	0.208057	0.061333	0.027380	0.063745	0.191567	0.241910	0.394510
18	0.055227	0.040911	0.099303	0.189405	0.340543	0.232605	0.215495	0.061825	0.067237	0.099632	0.221669	0.258685	0.398479
19	0.156760	0.102428	0.103117	0.188317	0.285737	0.206841	0.196496	0.204950	0.266455	0.246924	0.315975	0.267826	0.366260
20	0.117663	0.075255	0.063990	0.142271	0.205088	0.174768	0.165827	0.208785	0.264109	0.284062	0.287338	0.197051	0.261398
21	-0.056973	-0.074157	-0.026815	0.036010	0.152897	0.123770	0.063773	0.023786	0.019512	0.070381	0.150078	0.108243	0.168600
22	-0.163426	-0.179365	-0.073400	-0.029749	0.073934	0.064081	0.009359	-0.092087	-0.154752	-0.094887	-0.007886	0.002739	0.035288
23	-0.218093	-0.196469	-0.085380	-0.102975	-0.000624	0.027026	0.011982	-0.124427	-0.189343	-0.178304	-0.066705	-0.051297	-0.058068
24	-0.295683	-0.295302	-0.214256	-0.206673	-0.067296	-0.043280	-0.057147	-0.196354	-0.198658	-0.179890	-0.071493	-0.048840	-0.078022
25	-0.342865	-0.365749	-0.291974	-0.291357	-0.125675	-0.100309	-0.126074	-0.203178	-0.137459	-0.091960	-0.066961	-0.082879	-0.101443
26	-0.341703	-0.337046	-0.263111	-0.294749	-0.169618	-0.129094	-0.179526	-0.233332	-0.119143	-0.095820	-0.072859	-0.046687	-0.090046
27	-0.224340	-0.234386	-0.256674	-0.256074	-0.214692	-0.118645	-0.116848	-0.120343	-0.028002	-0.052303	-0.069156	-0.058846	-0.129290
28	-0.199099	-0.228940	-0.290728	-0.300476	-0.283863	-0.156081	-0.129694	-0.139750	-0.093443	-0.137413	-0.147221	-0.105324	-0.183920
29	-0.077430	-0.115301	-0.197493	-0.236602	-0.273350	-0.151186	-0.068142	-0.017654	0.053398	-0.043998	-0.134092	-0.177031	-0.254774
30	-0.048370	-0.055862	-0.106198	-0.190086	-0.214336	-0.054136	-0.096945	-0.081072	-0.041649	-0.091193	-0.154080	-0.102913	-0.183574
31	-0.030444	-0.049583	-0.109895	-0.169987	-0.173485	-0.051934	-0.115871	-0.108115	-0.028629	-0.058493	-0.121932	-0.050570	-0.140976
32	-0.031939	-0.108272	-0.170671	-0.164651	-0.200586	-0.144391	-0.127052	-0.087246	-0.017885	-0.027245	-0.125731	-0.124675	-0.217275
33	0.031319	-0.004247	-0.099409	-0.083965	-0.140559	-0.070337	-0.077662	-0.014578	0.013594	-0.021291	-0.120645	-0.148579	-0.255059
34	0.098118	0.115824	0.017053	0.015200	-0.086529	-0.028815	-0.015531	0.035733	0.015065	-0.035765	-0.162117	-0.248430	-0.316324
35	0.080722	0.132611	0.053070	0.039282	-0.073481	-0.023621	0.002979	0.087187	0.036120	-0.004460	-0.135543	-0.232932	-0.284109
36	0.119565	0.169169	0.107530	0.063486	-0.064617	-0.064798	-0.001376	0.110739	0.111769	0.085072	-0.057248	-0.149604	-0.235742
37	0.209873	0.217494	0.130276	0.089887	-0.008620	-0.048745	0.065900	0.186609	0.223983	0.175717	0.009732	-0.071704	-0.151781
38	0.208371	0.186828	0.110499	0.089346	0.063408	0.030599	0.080942	0.211897	0.233833	0.116225	0.070055	-0.051297	-0.093362
39	0.099993	0.098350	0.074137	0.045141	0.061616	0.081119	0.112673	0.184411	0.122735	0.117757	0.066532	0.055238	-0.027754
40	0.127313	0.188226	0.189047	0.145241	0.098832	0.075797	0.041071	0.097517	0.019589	-0.002523	-0.080723	-0.030371	-0.068478
41	0.213592	0.261345	0.233442	0.144693	0.125181	0.048763	-0.028720	0.076054	-0.005785	-0.018880	-0.011556	0.072670	0.040461
42	0.206057	0.186368	0.113920	0.050629	0.063706	0.034380	-0.025727	0.114721	0.052409	0.076138	0.101743	0.160656	0.082965
43	0.157949	0.133018	0.071946	-0.008407	0.031575	0.048870	0.061404	0.135426	0.215710	0.216742	0.185591	0.154133	0.084226
44	0.279968	0.285716	0.180734	0.087824	0.089202	0.085468	0.110813	0.240176	0.320573	0.287459	0.220705	0.123762	0.091938
45	0.319354	0.304247	0.173649	0.080012	0.081964	0.029524	0.076537	0.169099	0.195447	0.138447	0.077971	0.023391	0.016202
46	0.230343	0.255797	0.179528	0.046109	0.041419	0.016640	0.098925	0.109744	0.084191	0.090662	0.038045	0.022328	0.015964
47	0.203234	0.265279	0.234896	0.121065	0.084435	0.067196	0.155221	0.222783	0.225667	0.268123	0.185728	0.084092	0.076681
48	0.247560	0.313995	0.223074	0.133294	0.088128	0.080729	0.194720	0.271422	0.222135	0.264885	0.197313	0.103959	0.101629
49	0.269287	0.245868	0.081096	0.077925	0.066751	0.017300	0.166112	0.191615	0.150527	0.162010	0.047535	0.008992	0.058744
50	0.254450	0.320538	0.238110	0.174676	0.115936	0.171767	0.184152	0.260692	0.174873	0.167096	0.157615	0.113418	0.203347
51	0.355299	0.434548	0.394076	0.374651	0.266617	0.252288	0.144051	0.219038	0.207996	0.165537	0.165748	0.117699	0.147479
52	0.311729	0.346076	0.332914	0.364772	0.314985	0.162404	0.046403	0.102447	0.105352	0.097544	0.084801	0.042263	0.058599
53	0.322299	0.383960	0.367186	0.334211	0.205306	0.164073	0.163074	0.234008	0.202615	0.146725	0.142572	0.078457	0.160916
54	0.312067	0.380165	0.289731	0.284955	0.196472	0.133464	0.195541	0.239551	0.179342	0.175254	0.228991	0.164590	0.272492
55	0.220642	0.262263	0.287661	0.280938	0.199323	0.166758	0.174143	0.276819	0.232764	0.151889	0.122332	0.115658	0.183743
56	0.313725	0.280341	0.380819	0.340254	0.219395	0.161333	0.186324	0.267212	0.193963	0.140327	0.103405	0.030732	0.057870
57	0.368132	0.353042	0.334108	0.344865	0.238793	0.203986	0.242646	0.287603	0.231745	0.212277	0.193358	0.065273	0.171140
58	0.357116	0.352200	0.425047	0.420266	0.290982	0.220573	0.183578	0.194400	0.097293	0.058273	0.067726	0.044614	0.151804
59	0.347078	0.358761	0.373948	0.400626	0.253710	0.178158	0.222493	0.146216	0.095243	0.097358	0.089695	0.071364	0.061411

In [13]:

```
sonar_data.describe().T.style.background_gradient(cmap = 'rocket_r')
```

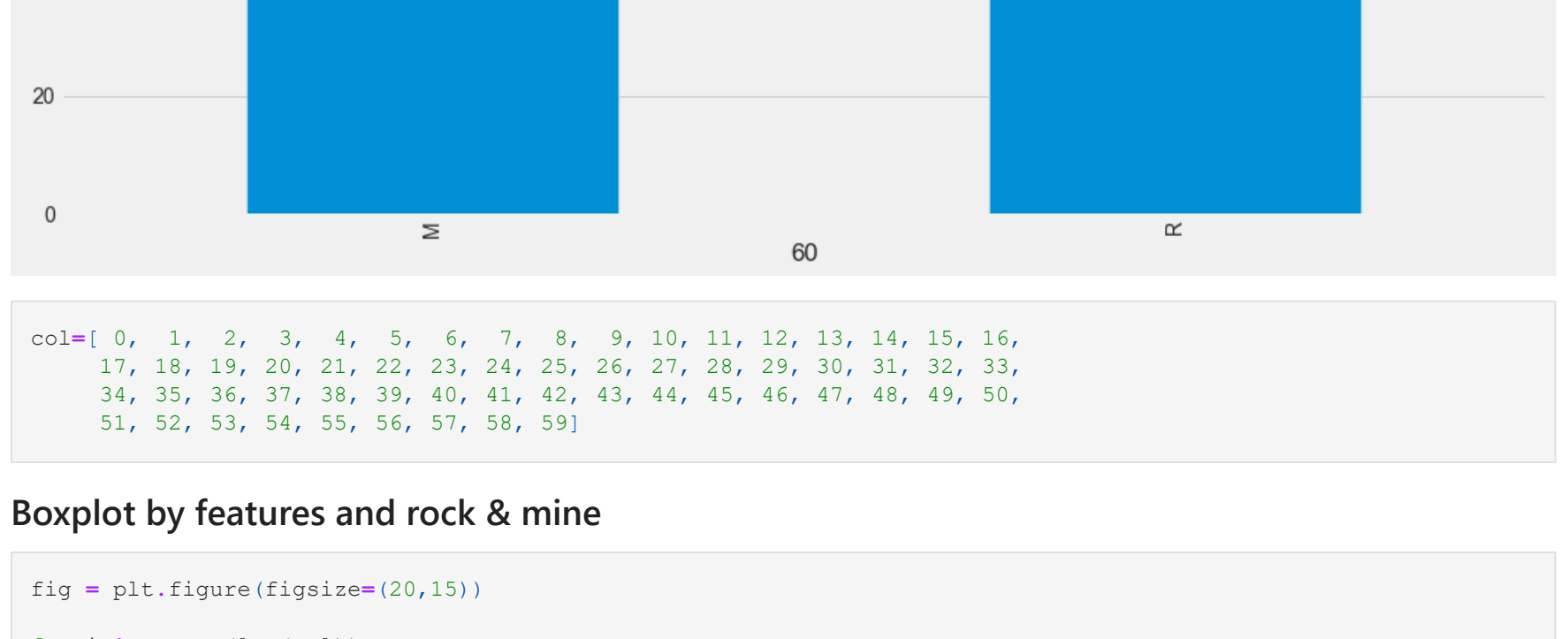
Out[13]:

	count	mean	std	min	25%	50%	75%	max
0	208.000000	0.029164	0.022991	0.001500	0.013350	0.022800	0.035550	0.137100
1	208.000000	0.038437	0.032960	0.000600	0.016450	0.030800	0.047950	0.233900
2	208.000000	0.043832	0.038428	0.001500	0.018950	0.034300	0.057950	0.305900
3	208.000000	0.053892	0.046528	0.005800	0.024375	0.044050	0.064500	0.426400
4	208.000000	0.075202	0.055552	0.006700	0.038050	0.062500	0.100275	0.401000
5	208.000000	0.104570	0.059105	0.010200	0.067025	0.092150	0.134125	0.382300
6	208.000000	0.121747	0.061788	0.003300	0.080900	0.106950	0.154000	0.372900
7	208.000000	0.134799	0.085152	0.005500	0.080425	0.112100	0.169600	0.459000
8	208.000000	0.178003	0.118387	0.007500	0.097025	0.152250	0.233425	0.682800
9	208.000000	0.208259	0.134416	0.011300	0.111275	0.182400	0.268700	0.710600
10	208.000000	0.236013	0.132705	0.028900	0.129250	0.224800	0.301650	0.734200
11	208.000000	0.250221	0.140072	0.023600	0.133475	0.249050	0.331250	0.706000
12	208.000000	0.273305	0.140962	0.018400	0.166125	0.263950	0.351250	0.713100
13	208.000000	0.296568	0.164474	0.027300	0.175175	0.281100	0.386175	0.997000
14	208.000000	0.320201	0.205427	0.003100	0.164625	0.281700	0.452925	1.000000
15	208.000000	0.378487	0.232650	0.016200	0.196300	0.304700	0.535725	0.998800
16	208.000000	0.415983	0.263677	0.034900	0.205850	0.308400	0.659425	1.000000
17	208.000000	0.452318	0.261529	0.037500	0.242075	0.368300	0.679050	1.000000
18	208.000000	0.504812	0.257988	0.049400	0.299075	0.434950	0.731400	1.000000
19	208.000000	0.563047	0.262653	0.065600	0.350625	0.542500	0.809325	1.000000
20	208.000000	0.609060	0.257818	0.051200	0.399725	0.617700	0.816975	1.000000
21	208.000000	0.624275	0.255883	0.021900	0.406925	0.664900	0.831975	1.000000
22	208.000000	0.646975	0.250175	0.056300	0.450225	0.699700	0.848575	1.000000
23	208.000000	0.672654	0.239116	0.023900	0.540725	0.698500	0.872175	1.000000
24	208.000000	0.675424	0.244926	0.024000	0.525800	0.721100	0.873725	1.000000
25	208.000000	0.699866	0.237228	0.092100	0.544175	0.754500	0.893800	1.000000
26	208.000000	0.702155	0.245657	0.048100	0.531900	0.745600	0.917100	1.000000
27	208.000000	0.694024	0.237189	0.028400	0.534775	0.731900	0.900275	1.000000
28	208.000000	0.642074	0.240250	0.014400	0.463700	0.680800	0.852125	1.000000
29	208.000000	0.580928	0.220749	0.061300	0.411400	0.607150	0.735175	1.000000
30	208.000000	0.504475	0.213992	0.048200	0.345550	0.490350	0.641950	0.965700
31	208.000000	0.439040	0.213237	0.040400	0.281400	0.429600	0.580300	0.930600
32	208.000000	0.417220	0.206513	0.047700	0.257875	0.391200	0.556125	1.000000
33	208.000000	0.403233	0.231242	0.021200	0.217575	0.351050	0.596125	0.964700
34	208.000000	0.392571	0.259132	0.022300	0.179375	0.312750	0.593350	1.000000
35	208.000000	0.384848	0.264121	0.008000	0.154350	0.321150	0.556525	1.000000
36	208.000000	0.363807	0.239912	0.035100	0.160100	0.306300	0.518900	0.949700
37	208.000000	0.339657	0.212973	0.038300	0.174275	0.312700	0.440550	1.000000
38	208.000000	0.325800	0.199075	0.037100	0.173975	0.283500	0.434900	0.985700
39	208.000000	0.311207	0.178662	0.011700	0.186450	0.278050	0.424350	0.929700
40	208.000000	0.289252	0.171111	0.036000	0.163100	0.259500	0.387525	0.899500
41	208.000000	0.278293	0.168728	0.005600	0.158900	0.245100	0.384250	0.824600
42	208.000000	0.246542	0.138993	0.000000	0.155200	0.222550	0.324525	0.773300
43	208.000000	0.214075	0.133291	0.000000	0.126875	0.177700	0.271750	0.776200
44	208.000000	0.197232	0.151628	0.000000	0.094475	0.148000	0.231550	0.703400
45	208.000000	0.160631	0.133938	0.000000	0.068550	0.121350	0.200375	0.729200
46	208.000000	0.122453	0.086953	0.000000	0.064250	0.101650	0.154425	0.552200
47	208.000000	0.091424	0.062417	0.000000	0.045125	0.078100	0.120100	0.333900
48	208.000000	0.051929	0.035954	0.000000	0.026350	0.044700	0.068525	0.198100
49	208.000000	0.020424	0.013665	0.000000	0.011550	0.017900	0.025275	0.082500
50	208.000000	0.016069	0.012008	0.000000	0.008425	0.013900	0.020825	0.100400
51	208.000000	0.013420	0.009634	0.000800	0.007275	0.011400	0.016725	0.070900
52	208.000000	0.010709	0.007060	0.000500	0.005075	0.009550	0.014900	0.039000
53	208.000000	0.010941	0.007301	0.001000	0.005375	0.009300	0.014500	0.035200
54	208.000000	0.009290	0.007088	0.000600	0.004150	0.007500	0.012100	0.044700
55	208.000000	0.008222	0.005736	0.000400	0.004400	0.006850	0.010575	0.039400
56	208.000000	0.007820	0.005785	0.000300	0.003700	0.005950	0.010425	0.035500
57	208.000000	0.007949	0.006470	0.000300	0.003600	0.005800	0.010350	0.044000
58	208.000000	0.007941	0.006181	0.000100	0.003675	0.006400	0.010325	0.036400
59	208.000000	0.006507	0.005031	0.000600	0.003100	0.005300	0.008525	0.043900

Barchart by rock and mine

In [14]:

```
sonar_data.groupby(60)[60].count().plot.bar();
```



In [15]:

```
col=[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
     17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
     34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
     51, 52, 53, 54, 55, 56, 57, 58, 59]
```

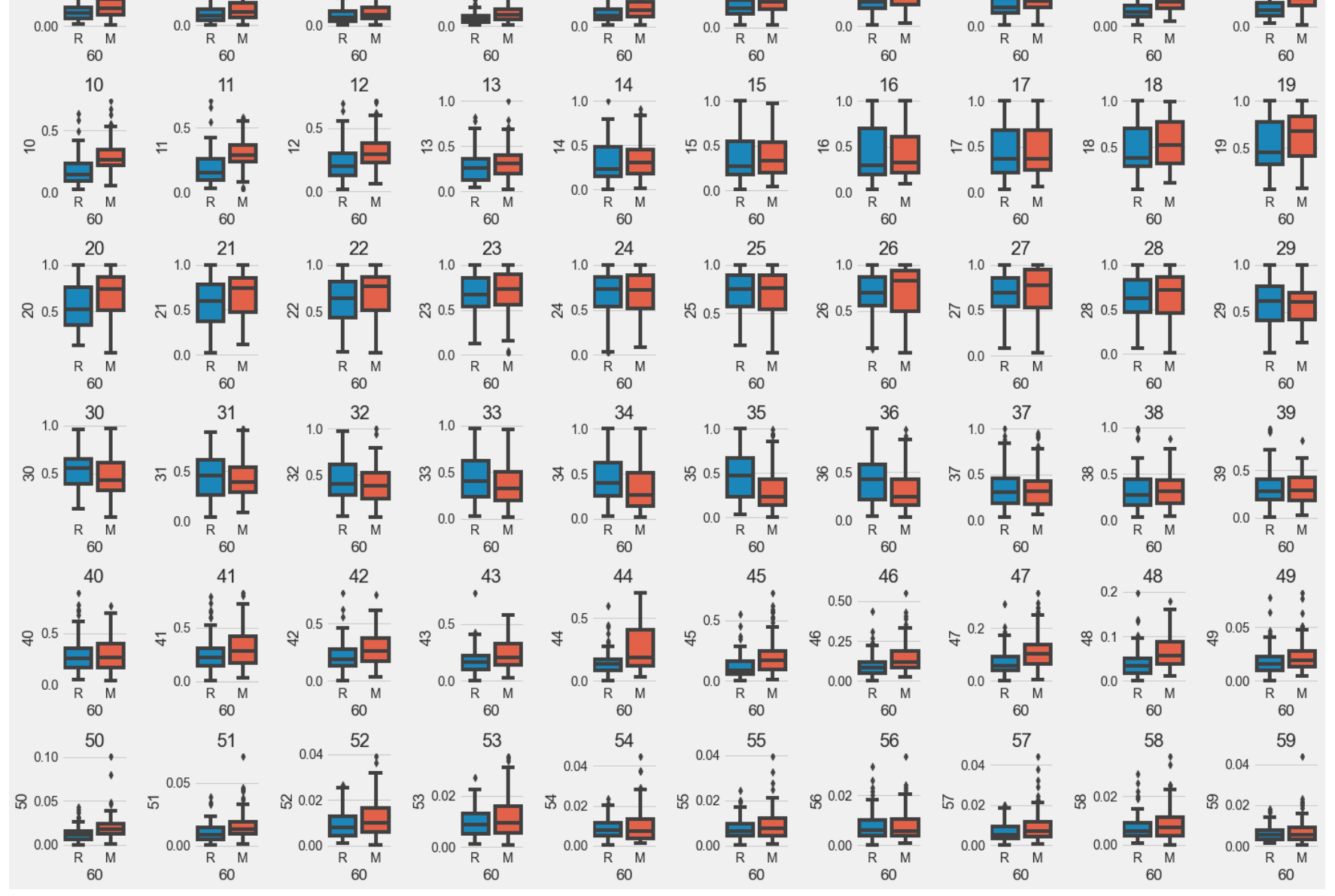
Boxplot by features and rock & mine

In [17]:

```
fig = plt.figure(figsize=(20,15))

for i in range(len(col)):
    plt.subplot(6,10,i+1)
    plt.title(col[i])
    sns.boxplot(data=sonar_data,y=sonar_data[col[i]],x=sonar_data[60])

plt.tight_layout()
plt.show()
```



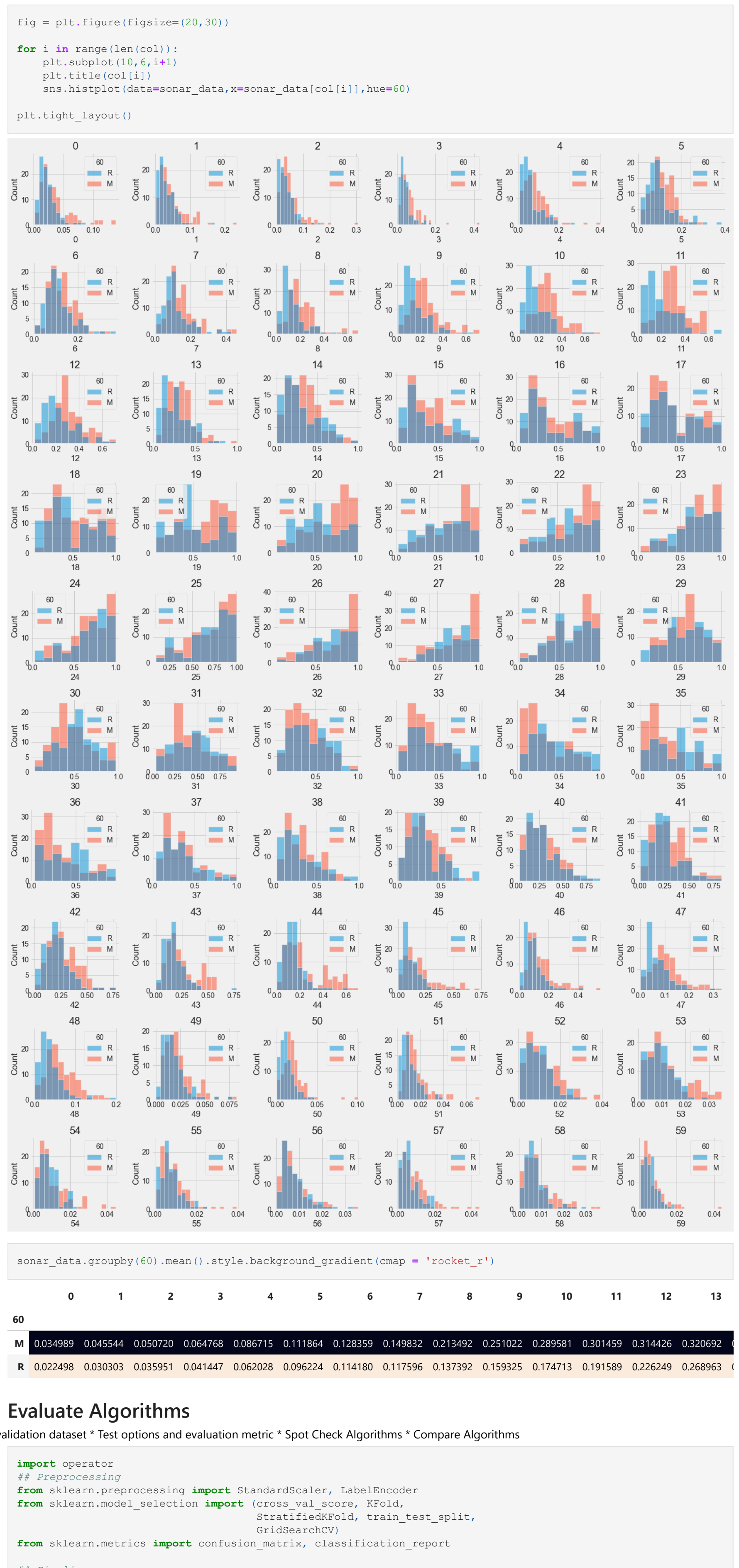
Histplot by features and rock & mine

In [18]:

```
fig = plt.figure(figsize=(20,30))

for i in range(len(col)):
    plt.subplot(10,6,i+1)
    plt.title(col[i])
    sns.histplot(data=sonar_data,x=sonar_data[col[i]],hue=60)

plt.tight_layout()
```



In [20]:

```
sonar_data.groupby(60).mean().style.background_gradient(cmap = 'rocket_r')
```

Out[20]:

60	0	1	2	3	4	5	6	7	8	9	10	11	12	13
M	0.034989	0.045544	0.050720	0.064768	0.086715	0.111864	0.128359	0.149832	0.213492	0.251022	0.289581	0.301459	0.314426	0.320692
R	0.022498	0.030303	0.035951	0.041447	0.062028	0.096224	0.114180	0.117596	0.137392	0.159325	0.174713	0.191589	0.226249	0.268963

Evaluate Algorithms

* Split-out validation dataset * Test options and evaluation metric * Spot Check Algorithms * Compare Algorithms

In [33]:

```
import operator
## Preprocessing
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import (cross_val_score, KFold,
                                     StratifiedKFold, train_test_split,
                                     GridSearchCV)
from sklearn.metrics import confusion_matrix, classification_report

## Piprline
from sklearn.pipeline import make_pipeline

### Linear Estimators
from sklearn.linear_model import LogisticRegression, SGDClassifier
### non Linear Estimators
from sklearn.svm import SVC, LinearSVC
from sklearn.neighbors import KNeighborsClassifier
### Ensemble Estimators

from sklearn.ensemble import (RandomForestClassifier, AdaBoostClassifier,GradientBoostingClassifier)
### Metrics
from sklearn.metrics import (ConfusionMatrixDisplay, confusion_matrix,
                             classification_report,RocCurveDisplay)
```

In [34]:

```
X = sonar_data.drop(columns=60, axis=1)
y = sonar_data[60]
'''
I split our data into Training and testing sets,
I specified my test size to 15% of the dataset
and last, I chose a random state of 101
'''
X_train, X_test, y_train, y_test =train_test_split(
                                   X,y,test_size=0.15, random_state = 101)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [35]:

```
np.random.seed(101)
models = {
    'LogisticRegression': LogisticRegression(),
    'SGDClassifier': SGDClassifier(),
    'LinearSVC': LinearSVC(),
    'SVC': SVC(),
    'KNeighborsClassifier': KNeighborsClassifier(),
    'RandomForestClassifier': RandomForestClassifier(),
    'AdaBoostClassifier':AdaBoostClassifier(),
    'GradientBoostingClassifier': GradientBoostingClassifier(),
}

Skfold = StratifiedKFold(n_splits = 13)
metrics = ['accuracy']
```

In [36]:

```
np.random.seed(101)
score ={}
def mod(model):
    for k,v in model.items():
        v.fit(X_train, y_train)
        score[k] = np.round(cross_val_score(estimator= v,
                                 X= X_test,y= y_test, cv= Skfold, scoring = 'accuracy').mean(), 4)
    best = max(score.items(), key=operator.itemgetter(1))[0]
    print(f'Best Estimator : {best} with score = {100*score[best]:.2f}')

    return score
mod(models)
```

```
Best Estimator : GradientBoostingClassifier with score = 89.74
```

Out[36]:

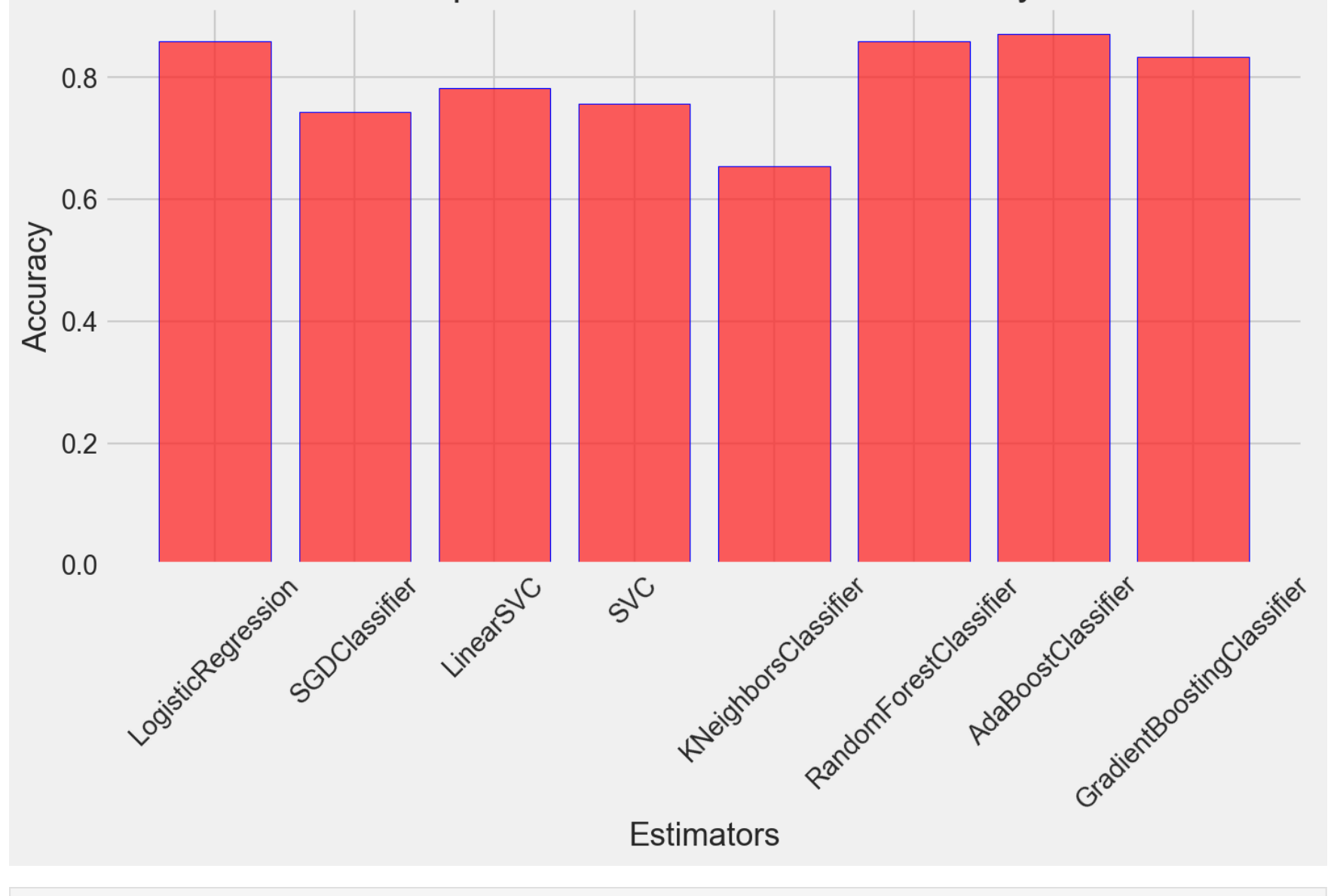
```
{'LogisticRegression': 0.859,
 'SGDClassifier': 0.859,
 'LinearSVC': 0.7821,
 'SVC': 0.7564,
 'KNeighborsClassifier': 0.6538,
 'RandomForestClassifier': 0.8846,
 'AdaBoostClassifier': 0.8718,
 'GradientBoostingClassifier': 0.8974}
```

In [37]:

```
# compare algorithms
results=mod(models)
estimators = list(results.keys())
score = list(results.values())


fig =  plt.figure(figsize= (10,5), dpi = 150)
plt.bar(range(len(results)), score, tick_label = estimators,
        color=(1, 0.1, 0.1, 0.7),  edgecolor='blue')
plt.xticks(rotation=45)
plt.xlabel('Estimators')
plt.ylabel('Accuracy')
plt.title('Comparison of Estimators on Accuracy');
```

```
Best Estimator : AdaBoostClassifier with score = 87.18
```



In [ ]: