SONAR ROCK VS MINE DETECTION

PROJECT Report

Submitted by

Dhawal Gupta-RA2011028010081
Anuj Shukla-RA2011028010074

Under the guidance of

Dr. Vaishnavi Moorthy

(Associate Professor, Department of Networking and Communication)

In partial satisfaction of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

with specialization in CLOUD COMPUTING

SCHOOL OF COMPUTING

COLLEGE OF ENGINEERING AND TECHNOLOGY SRM INSTITUTE OF SCIENCE AND TECHNOLOGY KATTANKULATHUR - 603203

MAY 2023

ABSTRACT

During a war between two countries the naval forces use submarines to fight wars. Submarines are at a great risk of getting blown by ocean mines planted by the enemies. The submarines are fully equipped with SONARs. These SONARs detect objects near the submarines. But not all the objects being detected are mines. Now how the submarine differentiates between the mine and a rock is a perfect use case of machine learning.

Sonar (sound navigation and ranging) is a technique based on the principle of reflection of ultrasonic sound waves. These waves propagate through water and reflect on hitting the ocean bed or any object obstructing its path.

Sonar has been widely used in submarine navigation, communication with or detection of objects on or under the water surface (like other vessels), hazard identification, etc.

There are two types of sonar technology used — passive (listening to the sound emitted by vessels in the ocean) and active (emitting pulses and listening for their echoes).

It is important to note that research shows the use of active sonar can cause mass strandings of marine animals.

TABLE OF CONTENT

CHAPTER NO	TITLE	PAGE NO
1.	ABSTRACT	2
2.	LITERATURE SURVEY	4
3.	WORKFLOW	5
4.	REQUIREMENT ANALYSIS	7
5.	DATASET	8
6.	DEPENDANCIES	9
7.	DATA COLLECTION AND DATA PROCESSING	10
8.	TRAINING AND TESTING DATA	13
9.	MODEL TRAINING	14
10.	MODEL EVALUATION	15
11.	MAKING UP THE PREDICTIVE SYSTEM	16
12.	FUTURE PROSPECTS	17
13.	CONCLUSION	18
14.	REFERENCES	18

LITERATURE SURVEY

The discovery of rocks and minerals would have been very difficult past the development of the SONAR technique, which relays on certain parameters to be able to detect the obstacle or the surface is a rock or a mine. Machine learning has drawn the attention of maximum part of the technology related and based industries, by showing advancements in the predictive analytics. The main aim is to emanate a capable prediction representative, united by the machine learning algorithmic characteristics, which can figure out if the target of the sound wave is either a rock or a mine or any other organism or any kind of other body. This attempt is a clear-cut case study which comes up with a machine learning plan for the grading of rocks and minerals, executed on a huge, highly spatial and complex SONAR dataset. The attempts are done on highly spatial SONAR dataset and achieved an accuracy of 83.17% and AUC came out to be 0.92. With random forest algorithm, the results are further optimized by feature selection to get the accuracy of 90%. Assuring results are found, when the fulfilment of the designed groundwork is set side by side with the standard classifiers like SVM, random forest, etc. using different evaluation metrics like accuracy, sensitivity, etc. Machine learning is performing a major role in improving the quality of detection of underwater natural resources, and will tend be better in the near future.

WORKFLOW

- Collection of sonar data – The SONAR data of the can be collected in form of a CSV file. CSV file can be viewed through MS excel, Notepad, or Power BI.
- Data pre-processing and analysis – Removal of unwanted data and outliers are an important part because it can drastically effect the accuracy score and other statistical measures.
- Feeding the sonar data to machine learning model – This is done to train the model before testing can be done. The higher amount of data we feed higher is the level of training and higher will be the accuracy score. Splitting into training and test data- Segregation of the given dataset is very important. Training data is the data from which the machine will be learning and training the model on the other hand test data is the data which is completely unknown to our machine.
- Logistic Regression – We will be using logistic regression model to predict between rock and mine which we are going to discuss in further slides
- Supervised Learning algorithm - A supervised learning algorithm takes a known set of input data (the learning set) and known responses to the data (the output), and forms a model to generate reasonable predictions for the response to the new input data.

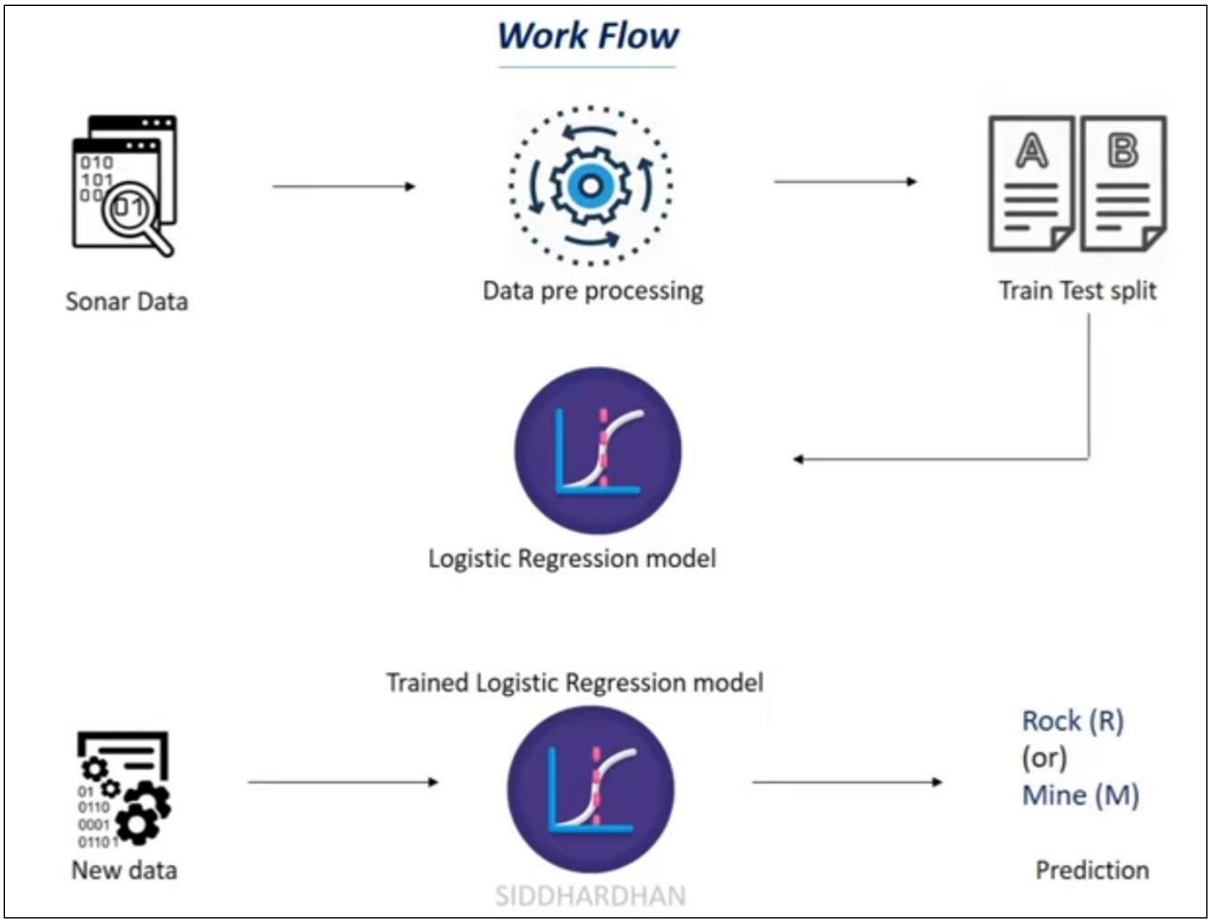


Figure 1 Workflow

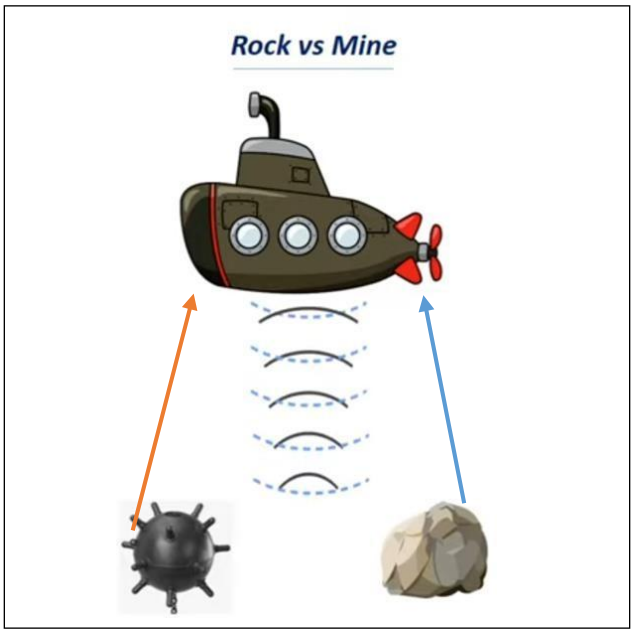


Figure 2 Working of SONAR

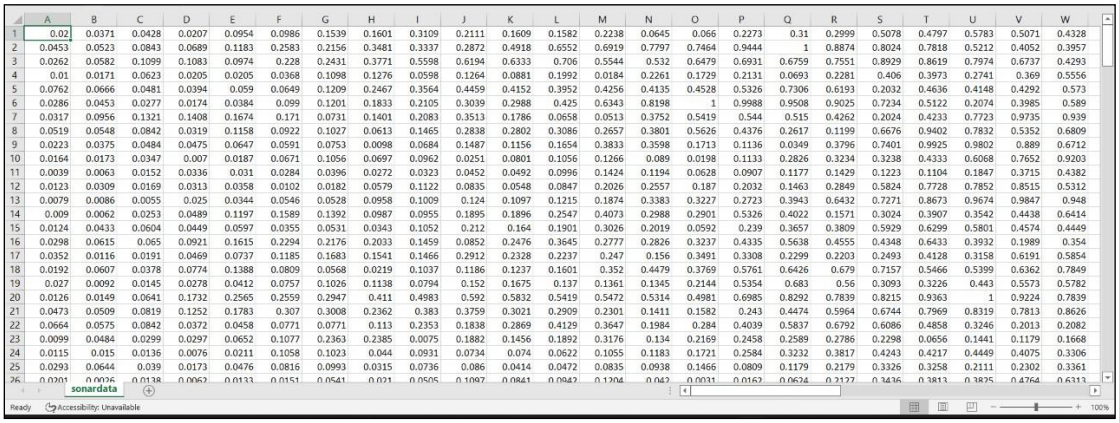
REQUIREMENT ANALYSIS

- MS EXCEL - A CSV is a comma-separated values file, which allows data to be saved in a tabular format. CSVs look like a garden-variety spreadsheet but with a . csv extension. CSV files can be used with most any spreadsheet program, such as Microsoft Excel.
- NOTEPAD -To extract data during the testing of our model
- JUPYTER NOTEBOOK - Jupyter Notebook is an open-source web application that allows a user, scientific researcher, scholar or analyst to create and share the document called the Notebook, containing live codes, documentation, graphs, plots, and visualizations.
- MS PowerPoint -To prepare the presentation of the project.
- AWS Sagemaker- Amazon SageMaker is a cloud-based machine learning platform offered by Amazon Web Services (AWS) that provides tools and infrastructure to build, train, and deploy machine learning models at scale. SageMaker is designed to help developers and data scientists quickly and easily build, train, and deploy machine learning models without requiring them to manage the underlying infrastructure.

OS and Device:

- 4GB Ram
- Windows 7 or above
- Chrome Browser (Google, Brave) updated.

DATASET



1	0.02	0.0371	0.0428	0.0207	0.0954	0.0986	0.1539	0.1601	0.3109	0.2111	0.1609	0.1582	0.2238	0.0645	0.066	0.2273	0.31	0.2999	0.5078	0.4797	0.5783	0.5071	0.4328
2	0.0453	0.0523	0.0843	0.0689	0.1183	0.2583	0.2156	0.3481	0.3337	0.2872	0.4918	0.6552	0.6919	0.7797	0.7464	0.9444	1	0.8874	0.8024	0.7818	0.5212	0.4052	0.3957
3	0.0262	0.0582	0.1099	0.1083	0.0974	0.228	0.2431	0.3771	0.5598	0.6194	0.6333	0.706	0.5544	0.532	0.6479	0.6931	0.6759	0.7551	0.8929	0.8619	0.7974	0.6737	0.4293
4	0.01	0.0171	0.0623	0.0205	0.0205	0.0368	0.1098	0.1276	0.0598	0.1264	0.0881	0.1992	0.0184	0.2261	0.1729	0.2131	0.0693	0.2281	0.406	0.3973	0.2741	0.369	0.5556
5	0.0762	0.0666	0.0481	0.0394	0.059	0.0649	0.1209	0.2467	0.3564	0.4459	0.4152	0.3952	0.4256	0.4135	0.4528	0.5326	0.7306	0.6193	0.2032	0.4636	0.4148	0.4292	0.573
6	0.0286	0.0453	0.0277	0.0174	0.0384	0.099	0.1201	0.1833	0.2105	0.3039	0.2988	0.425	0.6343	0.8198	1	0.9988	0.9508	0.9025	0.7234	0.5122	0.2074	0.3985	0.589
7	0.0317	0.0956	0.1321	0.1408	0.1674	0.171	0.0731	0.1401	0.2083	0.3513	0.1786	0.0658	0.0513	0.3752	0.5419	0.544	0.515	0.4262	0.2024	0.4233	0.7723	0.9735	0.939
8	0.0519	0.0548	0.0842	0.0319	0.1158	0.0922	0.1027	0.0613	0.1465	0.2838	0.2802	0.3086	0.2657	0.3801	0.5626	0.4376	0.2617	0.1199	0.6676	0.9402	0.7832	0.5352	0.6809
9	0.0223	0.0375	0.0484	0.0475	0.0647	0.0591	0.0753	0.0098	0.0684	0.1487	0.1156	0.1654	0.3833	0.3598	0.1713	0.1136	0.0349	0.3796	0.7401	0.9925	0.9802	0.889	0.6712
10	0.0164	0.0173	0.0347	0.007	0.0187	0.0671	0.1056	0.0697	0.0962	0.0251	0.0801	0.1056	0.1266	0.089	0.0198	0.1133	0.2826	0.3234	0.3238	0.4333	0.6068	0.7652	0.9203
11	0.0039	0.0063	0.0152	0.0336	0.031	0.0284	0.0396	0.0272	0.0323	0.0452	0.0492	0.0996	0.1424	0.1194	0.0628	0.0907	0.1177	0.1429	0.1223	0.1104	0.1847	0.3715	0.4382
12	0.0123	0.0309	0.0169	0.0313	0.0358	0.0102	0.0182	0.0579	0.1122	0.0835	0.0548	0.0847	0.2026	0.2557	0.187	0.2032	0.1463	0.2849	0.5824	0.7728	0.7852	0.8515	0.5312
13	0.0079	0.0086	0.0055	0.025	0.0344	0.0546	0.0528	0.0958	0.1009	0.124	0.1097	0.1215	0.1874	0.3383	0.3227	0.2723	0.3943	0.6432	0.7271	0.8673	0.9674	0.9847	0.948
14	0.009	0.0062	0.0253	0.0489	0.1197	0.1589	0.1392	0.0987	0.0955	0.1895	0.1896	0.2547	0.4073	0.2988	0.2901	0.5326	0.4022	0.1571	0.3024	0.3907	0.3542	0.4438	0.6414
15	0.0124	0.0433	0.0604	0.0449	0.0597	0.0355	0.0531	0.0343	0.1052	0.212	0.164	0.1901	0.3026	0.2019	0.0592	0.239	0.3657	0.3809	0.5929	0.6299	0.5801	0.4574	0.4449
16	0.0298	0.0615	0.065	0.0921	0.1615	0.2294	0.2176	0.2033	0.1459	0.0852	0.2476	0.3645	0.2777	0.2826	0.3237	0.4335	0.5638	0.4555	0.4348	0.6433	0.3932	0.1989	0.354
17	0.0352	0.0116	0.0191	0.0469	0.0737	0.1185	0.1683	0.1541	0.1466	0.2912	0.2328	0.2237	0.247	0.156	0.3491	0.3308	0.2299	0.2203	0.2493	0.4128	0.3158	0.6191	0.5854
18	0.0192	0.0607	0.0378	0.0774	0.1388	0.0809	0.0568	0.0219	0.1037	0.1186	0.1237	0.1601	0.352	0.4479	0.3769	0.5761	0.6426	0.679	0.7157	0.5466	0.5399	0.6362	0.7849
19	0.027	0.0092	0.0145	0.0278	0.0412	0.0757	0.1026	0.1138	0.0794	0.152	0.1675	0.137	0.1361	0.1345	0.2144	0.5354	0.683	0.56	0.3093	0.3226	0.443	0.5573	0.5782
20	0.0126	0.0149	0.0641	0.1732	0.2565	0.2559	0.2947	0.411	0.4983	0.592	0.5832	0.5419	0.5472	0.5314	0.4981	0.6985	0.8292	0.7839	0.8215	0.9363	1	0.9224	0.7839
21	0.0473	0.0509	0.0819	0.1252	0.1783	0.307	0.3008	0.2362	0.383	0.3759	0.3021	0.2909	0.2301	0.1411	0.1582	0.243	0.4474	0.5964	0.6744	0.7969	0.8319	0.7813	0.8626
22	0.0664	0.0575	0.0842	0.0372	0.0458	0.0771	0.0771	0.113	0.2353	0.1838	0.2869	0.4129	0.3647	0.1984	0.284	0.4039	0.5837	0.6792	0.6086	0.4858	0.3246	0.2013	0.2082
23	0.0099	0.0484	0.0299	0.0297	0.0652	0.1077	0.2363	0.2385	0.0075	0.1882	0.1456	0.1892	0.3176	0.134	0.2169	0.2458	0.2589	0.2786	0.2298	0.0656	0.1441	0.1179	0.1668
24	0.0115	0.015	0.0136	0.0076	0.0211	0.1058	0.1023	0.044	0.0931	0.0734	0.074	0.0622	0.1055	0.1183	0.1721	0.2584	0.3232	0.3817	0.4243	0.4217	0.4449	0.4075	0.3306
25	0.0293	0.0644	0.039	0.0173	0.0476	0.0816	0.0993	0.0315	0.0736	0.086	0.0414	0.0472	0.0835	0.0938	0.1466	0.0809	0.1179	0.2179	0.3326	0.3258	0.2111	0.2302	0.3361
26	0.0201	0.0026	0.0138	0.0092	0.0133	0.0151	0.0541	0.021	0.0505	0.1097	0.0841	0.0943	0.1204	0.042	0.0031	0.0162	0.0624	0.2177	0.3436	0.3813	0.3825	0.4764	0.6333

sonardata

Ready | Accessibility: Unavailable | 100%

27	0.0065	0.0159	0.0072	0.0167	0.018	0.0084	0.009	0.00 2 R
84	0.0089	0.0048	0.0094	0.0191	0.014	0.0049	0.0052	0.00 4 R
32	0.0166	0.0095	0.018	0.0244	0.0316	0.0164	0.0095	0.00 3 R
21	0.0036	0.015	0.0085	0.0073	0.005	0.0044	0.004	0.01 7 R
31	0.0054	0.0105	0.011	0.0015	0.0072	0.0048	0.0107	0.00 4 R
45	0.0014	0.0038	0.0013	0.0089	0.0057	0.0027	0.0051	0.00 2 R
01	0.0248	0.0131	0.007	0.0138	0.0092	0.0143	0.0036	0.01 3 R
81	0.012	0.0045	0.0121	0.0097	0.0085	0.0047	0.0048	0.00 3 R
45	0.0128	0.0145	0.0058	0.0049	0.0065	0.0093	0.0059	0.00 2 R
09	0.0223	0.0179	0.0084	0.0068	0.0032	0.0035	0.0056	0.0 4 R
62	0.012	0.0052	0.0056	0.0093	0.0042	0.0003	0.0053	0.00 5 R
33	0.0265	0.0224	0.0074	0.0118	0.0026	0.0092	0.0009	0.00 4 R
76	0.0127	0.0088	0.0098	0.0019	0.0059	0.0058	0.0059	0.00 2 R
59	0.0095	0.0194	0.008	0.0152	0.0158	0.0053	0.0189	0.01 2 R
83	0.0057	0.0174	0.0188	0.0054	0.0114	0.0196	0.0147	0.00 2 R
31	0.0153	0.0071	0.0212	0.0076	0.0152	0.0049	0.02	0.00 3 R
46	0.0158	0.0154	0.0109	0.0048	0.0095	0.0015	0.0073	0.00 7 R
31	0.0131	0.012	0.0108	0.0024	0.0045	0.0037	0.0112	0.00 5 R
84	0.001	0.0018	0.0068	0.0039	0.012	0.0132	0.007	0.00 3 R
92	0.0035	0.0098	0.0121	0.0006	0.0181	0.0094	0.0116	0.00 3 R
93	0.0118	0.0064	0.0042	0.0054	0.0049	0.0082	0.0028	0.00 7 R
41	0.019	0.0043	0.0036	0.0026	0.0024	0.0162	0.0109	0.00 9 R
73	0.0149	0.0115	0.0202	0.0139	0.0029	0.016	0.0106	0.01 4 R
91	0.0016	0.0084	0.0064	0.0026	0.0029	0.0037	0.007	0.00 1 R
35	0.0052	0.0083	0.0078	0.0075	0.0105	0.016	0.0095	0.00 1 R
08	0.007	0.0063	0.003	0.0011	0.0007	0.0024	0.0057	0.00

Figure 3 Dataset

- Data is collected and is in CSV format.
- There are all total of 208 data points.
- If we see the last column of every data point its mentioned R or M.
- R denotes Rock and M denotes Mine.
- Since we already know the results from the beginning, we can say it's a Supervised Learning Algorithm.
- We feed this data set for our machines for model training

DEPENDENCIES

A Python library is a collection of related modules. It contains bundles of code that can be used repeatedly in different programs. It makes Python Programming simpler and convenient for the programmer. As we don't need to write the same code again and again for different programs.

NumPy – NumPy can be used to perform a wide variety of mathematical operations on arrays.

Pandas - Pandas is mainly used for data analysis and associated manipulation of tabular data in data frames.

Sklearn - Scikit-learn is probably the most useful library for machine learning in Python. The sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.

Sklearn.modelselection – train_test_split - train_test_split is a function in Sklearn model selection for splitting data arrays into two subsets: for training data and for testing data.

Sklearn.linear_model import logisticRegression - linear_model is a class of the sklearn module if contain different functions for performing machine learning with linear models.

Sklearn.Metrics import accuracy_score - Accuracy classification score. In multilabel classification, this function computes subset accuracy.

DATA COLLECTION AND DATA PROCESSING

Data collection is the process of gathering and measuring information from countless different sources.

Data Processing is the task of converting data from a given form to a much more usable and desired form i.e. making it more meaningful and informative.

```
In [8]: import numpy as np
        import pandas as pd
        from sklearn.model_selection import train_test_split
        from sklearn.linear_model import LogisticRegression
        from sklearn.metrics import accuracy_score

In [9]: #Loading the dataset to a pandas Dataframe
        sonar_data = pd.read_csv('.\sonardata.csv', header=None)

In [10]: sonar_data.head()

Out[10]:
```

	0	1	2	3	4	5	6	7	8	9	...	51	52	53	54	55	56	57	58	59	60
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	0.0090	0.0032	R
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	0.0052	0.0044	R
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	0.0095	0.0078	R
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	0.0040	0.0117	R
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	0.0107	0.0094	R

5 rows × 61 columns

Figure 4 Reading the Dataset

According to Fig. 4:

1. Loading the dataset and making it an pandas data frame.
2. Read.csv () to read the csv file containing the sonar data.
3. Since there is no header file, we need to mention that as ‘none’.

```
In [11]: # number of rows and columns
         sonar_data.shape

Out[11]: (208, 61)

In [12]: sonar_data.describe()  #describe --> statistical measures of the data

Out[12]:
```

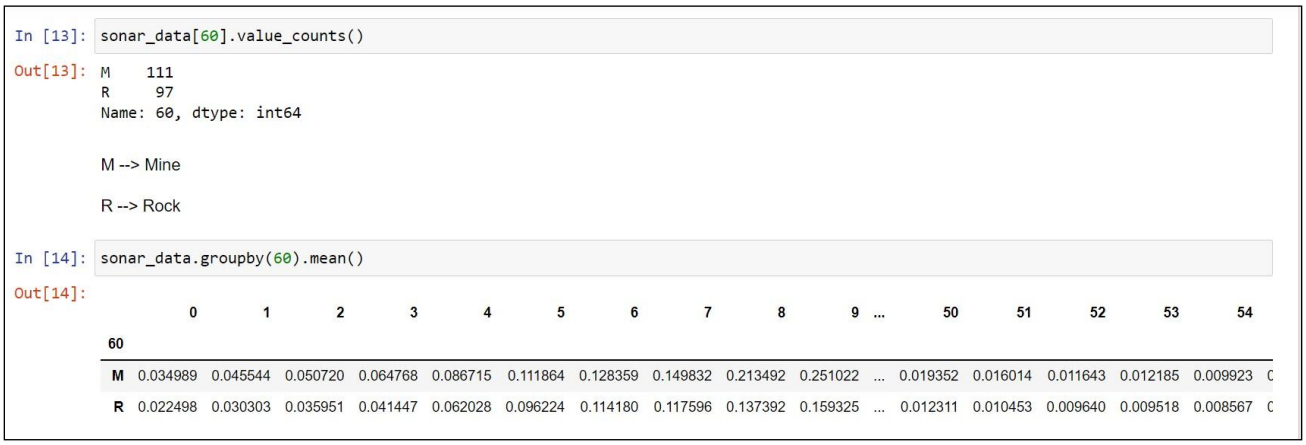
	0	1	2	3	4	5	6	7	8	9	...	50	51	20
count	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	208.000000	...	208.000000	208.000000	
mean	0.029164	0.038437	0.043832	0.053892	0.075202	0.104570	0.121747	0.134799	0.178003	0.208259	...	0.016069	0.013420	
std	0.022991	0.032960	0.038428	0.046528	0.055552	0.059105	0.061788	0.085152	0.118387	0.134416	...	0.012008	0.009634	
min	0.001500	0.000600	0.001500	0.005800	0.006700	0.010200	0.003300	0.005500	0.007500	0.011300	...	0.000000	0.000800	
25%	0.013350	0.016450	0.018950	0.024375	0.038050	0.067025	0.080900	0.080425	0.097025	0.111275	...	0.008425	0.007275	
50%	0.022800	0.030800	0.034300	0.044050	0.062500	0.092150	0.106950	0.112100	0.152250	0.182400	...	0.013900	0.011400	
75%	0.035550	0.047950	0.057950	0.064500	0.100275	0.134125	0.154000	0.169600	0.233425	0.268700	...	0.020825	0.016725	
max	0.137100	0.233900	0.305900	0.426400	0.401000	0.382300	0.372900	0.459000	0.682800	0.710600	...	0.100400	0.070900	

8 rows × 60 columns

Figure 5 Determining the statistical data.

According to Fig. 5:

4. .shape() - We check the number of rows and columns present in the data set.
5. We get the result as 208 rows, 61 columns.
6. .describe() – to determine mean, SD and other statistical measures for our data.
7. Determine how many data of rocks are there and how many data for mines.



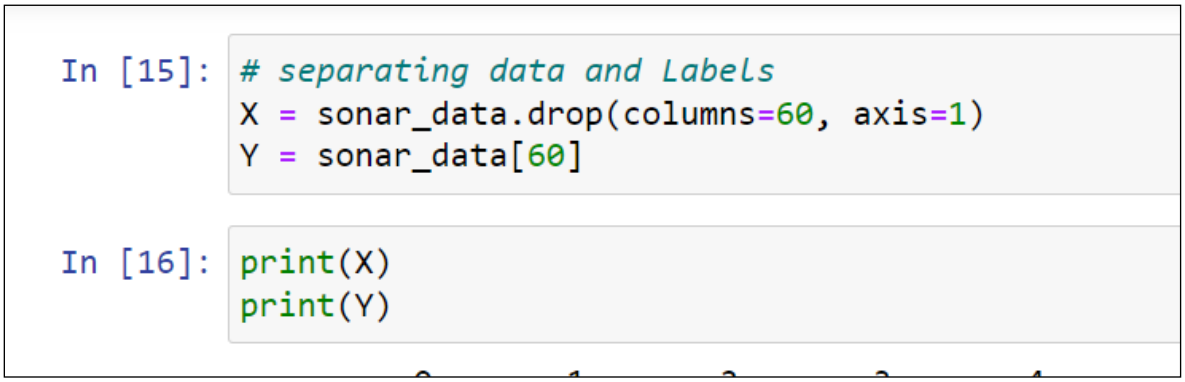
```
In [13]: sonar_data[60].value_counts()

Out[13]: M    111
         R     97
         Name: 60, dtype: int64

         M --> Mine

         R --> Rock

In [14]: sonar_data.groupby(60).mean()

Out[14]:
```

60	0	1	2	3	4	5	6	7	8	9	...	50	51	52	53	54
M	0.034989	0.045544	0.050720	0.064768	0.086715	0.111864	0.128359	0.149832	0.213492	0.251022	...	0.019352	0.016014	0.011643	0.012185	0.009923
R	0.022498	0.030303	0.035951	0.041447	0.062028	0.096224	0.114180	0.117596	0.137392	0.159325	...	0.012311	0.010453	0.009640	0.009518	0.008567

Figure 6 Grouping the data

According to Fig. 6:

8. groupby(60) because the rock and mine are specified in the 60th column
9. 111 – mines 97 – rocks
10. More the data more accurate will be the model.
11. groupby() - We need to group the data on the basis of rock and mine.



```
In [15]: # separating data and Labels
         X = sonar_data.drop(columns=60, axis=1)
         Y = sonar_data[60]

In [16]: print(X)
         print(Y)
```

Figure 7 Separating the data and lebels.

```
..      ...     ...     ...
203  0.0115  0.0193  0.0157
204  0.0032  0.0062  0.0067
205  0.0138  0.0077  0.0031
206  0.0079  0.0036  0.0048
207  0.0036  0.0061  0.0115

[208 rows x 60 columns]
0      R
1      R
2      R
3      R
4      R
      ..
203    M
204    M
205    M
206    M
207    M
Name: 60, Length: 208, dtype: object
```

Figure 8 The last column is separately stored

12.Separate data and labels – we are dropping the 60th column and storing 60th column in another variable.

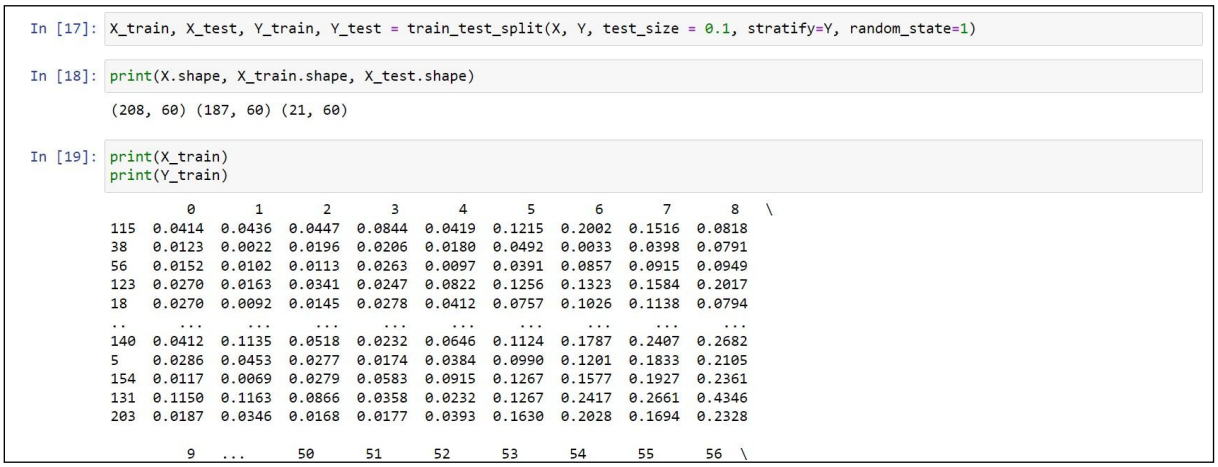
TRAINING AND TEST DATA

- We need to split the data into training and test data.

Training data is the data you use to train an algorithm or machine learning model to predict the outcome you design your model to predict.

Test data is used to measure the performance, such as accuracy or efficiency, of the algorithm you are using to train the machine.

We need to split a dataset into train and test sets to evaluate how well our machine learning model performs. Typically, when you separate a data set into a training set and testing set, most of the data is used for training, and a smaller portion of the data is used for testing.



```
In [17]: X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.1, stratify=Y, random_state=1)

In [18]: print(X.shape, X_train.shape, X_test.shape)

         (208, 60) (187, 60) (21, 60)

In [19]: print(X_train)
         print(Y_train)

                 0       1       2       3       4       5       6       7       8   \
         115  0.0414  0.0436  0.0447  0.0844  0.0419  0.1215  0.2002  0.1516  0.0818
         38   0.0123  0.0022  0.0196  0.0206  0.0180  0.0492  0.0033  0.0398  0.0791
         56   0.0152  0.0102  0.0113  0.0263  0.0097  0.0391  0.0857  0.0915  0.0949
         123  0.0270  0.0163  0.0341  0.0247  0.0822  0.1256  0.1323  0.1584  0.2017
         18   0.0270  0.0092  0.0145  0.0278  0.0412  0.0757  0.1026  0.1138  0.0794
         ..      ...     ...     ...     ...     ...     ...     ...     ...     ...
         140  0.0412  0.1135  0.0518  0.0232  0.0646  0.1124  0.1787  0.2407  0.2682
         5    0.0286  0.0453  0.0277  0.0174  0.0384  0.0990  0.1201  0.1833  0.2105
         154  0.0117  0.0069  0.0279  0.0583  0.0915  0.1267  0.1577  0.1927  0.2361
         131  0.1150  0.1163  0.0866  0.0358  0.0232  0.1267  0.2417  0.2661  0.4346
         203  0.0187  0.0346  0.0168  0.0177  0.0393  0.1630  0.2028  0.1694  0.2328

                  9   ...      50      51      52      53      54      55      56  \
```

Figure 9 Splitting the data into training data and test data.

According to Fig. 9:

- (X, Y, test_size, Stratify, random_state)
- We need 10% of the data to be test data, so test_size is kept as 0.1.
- Stratify: We need to split the data equally based on number of rock and mine.
- Random_state: To split our data in a particular way.
- 187 training data and 21 test data.

MODEL TRAINING

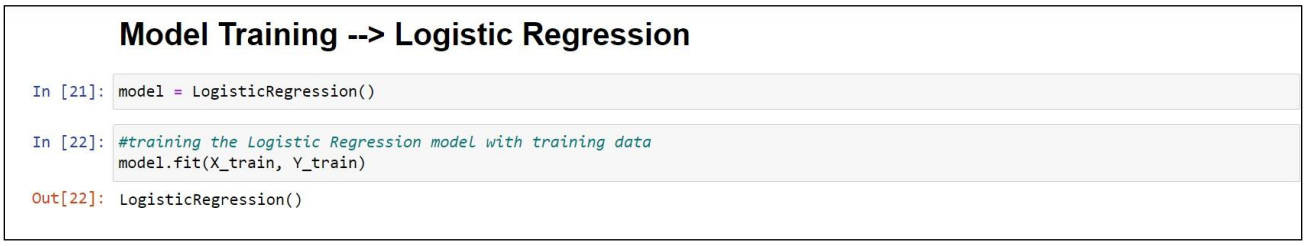


Figure 10 Model training

Model training in machine language is the process of feeding an ML algorithm with data to help identify and learn good values for all attributes involved.
For our model training we used **Logistic regression**.

Logistic regression is a process of modeling the probability of a discrete outcome given an input variable. The most common logistic regression models a binary outcome; something that can take two values such as true/false, yes/no, and so on. For our case it's R or M that is a rock or a mine.

According to Fig. 10:

- We use the training data to train our model.
- .fit(X_train, Y_train) to train our model.
- The fit() method takes the training data as arguments, which can be one array in the case of unsupervised learning, or two arrays in the case of supervised learning.

MODEL EVALUATION

```
In [23]: #accuracy on training data
         X_train_prediction = model.predict(X_train)
         training_data_accuracy = accuracy_score(X_train_prediction, Y_train)

In [24]: print('Accuracy on training data : ', training_data_accuracy)

         Accuracy on training data :  0.8342245989304813
```

Figure 11 Accuracy of training data

- The training data accuracy and the test data accuracy are both calculated and both are more than 70%, so the accuracy is good.
- If we would have used more data somewhat like 1000-2000 the accuracy would be little higher.
- Model.predict() – prediction of training data accuracy score.
- 83.42 % accuracy for training.

```
In [25]: #accuracy on test data
         X_test_prediction = model.predict(X_test)
         test_data_accuracy = accuracy_score(X_test_prediction, Y_test)

In [26]: print('Accuracy on test data : ', test_data_accuracy)

         Accuracy on test data :  0.7619047619047619
```

Figure 12 Accuracy of test data.

- Now we will check accuracy on test data. The model has not seen this data yet.
- 76.19 % accuracy for testing.
- So out of 100 time 76 times the model can predict the correct object.

MAKING UP THE PREDICTIVE SYSTEM

- We take the input data and convert in np array to reduce the time complexity.
- We feed a sample data to check whether the model is working fine or not.
- We need to reshape the array to avoid confusion from the model side as we are predicting for one instance.
- Calling the model functions in the model.predict().
- Returns R or M.

```
In [29]: #input_data = (0.0307,0.0523,0.0653,0.0521,0.0611,0.0577,0.0665,0.0664,0.1460,0.2792,0.3877,0.4992,0.4981,0.4972,0.5607,0.7339,0
         #input_data = (0.0200,0.0371,0.0428,0.0207,0.0954,0.0986,0.1539,0.1601,0.3109,0.2111,0.1609,0.1582,0.2238,0.0645,0.0660,0.2273,0
         input_data = (0.0164,0.0627,0.0738,0.0608,0.0233,0.1048,0.1338,0.0644,0.1522,0.0780,0.1791,0.2681,0.1788,0.1039,0.1980,0.3234,0.

         # changing the input_data to a numpy array
         input_data_as_numpy_array = np.asarray(input_data)

         # reshape the np array as we are predicting for one instance
         input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

         prediction = model.predict(input_data_reshaped)
         print(prediction)

         if (prediction[0]=='R'):
           print('The object is a Rock')
         else:
           print('The object is a mine')

         ['M']
         The object is a mine
```

Figure 13 Testing our prediction system with one Mine Data

```
In [30]: #input_data = (0.0307,0.0523,0.0653,0.0521,0.0611,0.0577,0.0665,0.0664,0.1460,0.2792,0.3877,0.4992,0.4981,0.4972,0.5607,0.7339,0.
         input_data = (0.0200,0.0371,0.0428,0.0207,0.0954,0.0986,0.1539,0.1601,0.3109,0.2111,0.1609,0.1582,0.2238,0.0645,0.0660,0.2273,0.3
         #input_data = (0.0164,0.0627,0.0738,0.0608,0.0233,0.1048,0.1338,0.0644,0.1522,0.0780,0.1791,0.2681,0.1788,0.1039,0.1980,0.3234,0.

         # changing the input_data to a numpy array
         input_data_as_numpy_array = np.asarray(input_data)

         # reshape the np array as we are predicting for one instance
         input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

         prediction = model.predict(input_data_reshaped)
         print(prediction)

         if (prediction[0]=='R'):
           print('The object is a Rock')
         else:
           print('The object is a mine')

         ['R']
         The object is a Rock
```

Figure 14 Testing our prediction system with one Rock Data.

FUTURE PROSPECTS

In the ocean bed there are many more unwanted materials other than rock which can affect the prediction procedure of our model. Radioactive wastes, plastic wastes and many other kinds of mines are also present there. For such a sensitive calculation the accuracy should be very high like 85-90%. This machine learning model which we made needs a lot more of improvement and improvisation to properly determine the type of material that is encountered. For future work more, complex data will be handled using big data Hadoop framework. With random forest algorithm, the results are further optimized by feature selection to get the accuracy of 91.15% The project we did majorly focuses on the backend working of the sonar the frontend development need the knowledge of flask or Django framework. After acquiring that knowledge, we would be able to develop the frontend and think of deployment.

Classifier	Accuracy	Error	TP Rate	FP Rate	F Score	MCC	AUC
Decision tree	75.48	24.51	0.75	0.24	0.75	0.50	0.78
Adaboost	71.64	28.36	0.71	0.28	0.71	0.42	0.84
Random forest	83.17	16.82	0.83	0.17	0.83	0.66	0.92
SVM	73.07	26.92	0.73	0.28	0.72	0.45	0.70
Logistic	72.58	27.40	0.72	0.27	0.72	0.44	0.77
Neural network	71.63	28.36	0.71	0.28	0.71	0.42	0.84
Decision stump	73.07	26.92	0.73	0.28	0.73	0.45	0.70
J48	71.15	18.84	0,71	0.28	0.71	0.42	0.74
Naive bayes	67.78	32.21	0.67	0.30	0.67	0.36	0.80
Bayes net	80.28	19.71	0.80	0.20	0.80	0.60	0.88
Optimized result	**91.15**	8.84	0.90	0.14	0.89	0.79	0.92

Figure 15 Sample of Real SONAR Data

A fully developed SONAR system has to take into consideration several kinds of Data and these are a part of marine engineering which is out of the scope of this projects and to reach at our supreme accuracy we need to collaborate with other students from different streams and make it interdisciplinary.

CONCLUSION

An adequate prediction miniature, united with the machine learning classifying features, is proposed which can conclude if the target of the sound wave is either a rock or a mine.

REFERENCES

- Hands–On Machine Learning with Scikit–Learn and TensorFlow, Aurelien Geron.
- https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc
- https://www.ibm.com/topics/logistic-regression
- https://www.javatpoint.com/logistic-regression-in-machine-learning
- https://www.javatpoint.com/supervised-machine-learning
- https://www.javatpoint.com/supervised-machine-learning
- Dura, Esther, et al. "Active learning for detection of mine-like objects in side-scan sonar imagery." IEEE Journal ofOceanic Engineering 30.2: 360-371 (2005).
- https://www.researchgate.net/publication/330958762_Prediction_of_Underwater_Surface_Target_through_SONAR_A_Case_Study_of_Machine_Learning

GITHUB LINK:

https://github.com/Dg8258/Sonar-Mine-Detection

https://github.com/anujshukla0406/Sonar_Mine_detection