**Rock vs Mine Prediction System**

*Submitted for Mini Project Report*

by

|  |  |  |
| --- | --- | --- |
| **Goli Hemanth** |  | **19BQ1A0553** |
| **Eluri Jahnavi** |  | **19BQ1A0547** |
| **Aramalla Vasundhara** |  | **19BQ1A0505** |
| **Bharata Raj Kumar** |  | **20BQ5A0505** |

Under the guidance of

**Ms.Desam Vamsi**

**Assistant Professor**



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

(B.Tech Program is Accredited by NBA)

**VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY**

Permanently Affiliated to JNTU Kakinada, Approved by AICTE

Accredited by NAAC with ‘A’ Grade, ISO 9001:2008 Certified

NAMBUR(V), PEDAKAKANI(M), GUNTUR-522 508

Tel no: 0863-2118036, url:www.vvitguntur.com

June 2022



**CERTIFICATE**

This is to certify that this Mini Project Report is the bonafide work of **G. Hemanth, E. Jahnavi, A. Vasundhara, B. Raj Kumar** bearing **Reg. No. 19BQ1A0553, 19BQ1A0547, 19BQ1A0505, 20BQ5A0505** who had carried out the project entitled “**Rock vs Mine Prediction System**” under our supervision.

**Project Guide Head of the Department**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Submitted for Viva voce Examination held on \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Internal Examiner External Examiner**

**ABSTRACT**

Underwater Mine usage by the naval defense system provides great security but also possesses a threat to the marine life and submarine vessels as the mines can be easily mistaken for rocks. We need a much more accurate system to predict the object as it is very dangerous if a mistake is made. To have a great accuracy we need accurate data to generate accurate results. We worked on the data set which is provided by Gorman, R. P., and Sejnowski, T. J. (1988). The data is used to train the machine. This project presents a method for the prediction of underwater mines and rocks using Sonar signals. Sonar signals are used to record the various frequencies of underwater objects at 60 different angles. We constructed three binary classifier models according to their accuracy. Then, prediction models are used to predict the mine and rock categories. Python and Supervised Machine Learning Classification algorithms are used to construct these prediction models.

**TABLE OF CONTENTS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CH no.** | **TITLE** |  |  | **PAGE NO** |
|  | ABSTRACT |  |  |  |
|  | Contents |  |  |  |
| 1. | Introduction |  |  | 1 |
| 2. | Concepts & Methods |  |  | 2 |
|  | * 1. Existing System |  |  | 2 |
|  | * 1. Proposed Solution |  |  | 2 |
|  | * 1. System Requirements |  |  | 3 |
| 3. | Implementation |  |  | 4 |
|  | * 1. Dataset Used |  |  | 4 |
|  | * 1. Libraries used |  |  | 4 |
|  | * 1. Algorithms Used   2. Code Snippets   3. Screenshots |  |  | 8  13  17 |
| 4. | Summary (or) Conclusion |  |  | 24 |
|  | BIBLIOGRAPHY |  |  | 24z3 |

**CHAPTER -1**

1. **INTRODUCTION**

Underwater mines or naval mines are self-contained explosive devices placed in water to destroy enemies’ surface ships or submarines. Usually, mines are mistaken as rocks during their identification, as mines can have the same shape, length, and width as rocks. To avoid this confusion it is better to use a more accurate input to receive an accurate output. One of the methods in detecting the mines is SONAR.

Sound Navigation and Ranging system works on sound waves to steer and detect objects. In general, SONAR is used for acoustic mine detection that comes under Military purposes. In mine hunting, the frequencies of underwater SONARs vary between 0.1 and 1 MHz (and their range between 1 and 0.1 km, respectively). Sonar prefers ultrasonic waves instead of infrasonic, as they cannot move under the water, and even though they have long wavelengths they cannot capture much energy. We have active and passive SONAR. Passive SONAR is only used to detect noises so they are named Listening SONAR.

We have a sound transmitter and receiver in active sonar. When a sound wave from the transmitter reaches the target, it propagates and reflects an echo. The receiver decodes the echo and records the target object's frequencies. The frequency of active sonar is normally in the 20KHz range. We utilize the frequencies obtained by active sonar at 60 various angles as our input to determine if the target is a mine or a rock in this case.

**CHAPTER – 2**

1. **CONCEPTS AND METHODS**
   1. EXISTING SYSTEM

In the existing system, the detection of mines is done by explosive ordnance disposal divers, marine mammals, video cameras on mine neutralization vehicles, laser systems, etc but not by using a definite data set or equipment which can cause risk and loss to the marine life if it goes wrong. As technology improved SONAR is being used as a primary tool to detect the mines.

* 1. PROPOSED SOLUTION

We have proposed a predictive system to give accurate results and outcomes. We utilized the dataset from "Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets" by R. Paul Gorman and Terrence J. Sejnowski. They employed SONAR to perform trials in a simulated region with metal cylinders in place of mines. Sound Navigation and Ranging system works on sound waves to steer and detect objects. The object was struck with sonar signals from 60 various angles, and the results were recorded. The dataset is then trained to the evaluated models. The Sonar output frequencies are sent into the predictive system as input. We use classification machine learning techniques to predict if the object is a Rock or a Mine. Python and Supervised Machine Learning Classification algorithms are used to construct these prediction models

* 1. **SYSTEM REQUIREMENTS** 
     1. *SOFTWARE USED****:***

This section describes the software requirements which are used for developing the project.

|  |  |  |
| --- | --- | --- |
| S.No | Software | Requirement |
| 1. | Operating system | Windows/Mac/Linux/Chrome Os |
| 2. | Anaconda | Latest version |
| 3. | python | Latest Version |

* + 1. *HARDWARE USED****:***

This section describes the hardware requirements which are used for developing the project.

|  |  |  |
| --- | --- | --- |
| S.No | Hardware | Requirement |
| 1. | Processor | 2.4 GHz Processor speed |
| 2. | RAM | minimum 4GB |
| 3. | disk space | 8 GB of available disk space minimum |

**CHAPTER - 3**

1. **IMPLEMENTATION:**
   1. DATASET USED:-

The file "sonar.mines" contains 111 patterns obtained by bouncing sonar signals off a metal cylinder at various angles and under various conditions. The file "sonar.rocks" contains 97 patterns obtained from rocks under similar conditions. The transmitted sonar signal is a frequency-modulated chirp, rising in frequency. The data set contains signals obtained from a variety of different aspect angles, spanning 90 degrees for the cylinder and 180 degrees for the rock.

Each pattern is a set of 60 numbers in the range 0.0 to 1.0.  Each number represents the energy within a particular frequency band, integrated over a certain period of time.  The integration aperture for higher frequencies occur later in time, since these frequencies are transmitted later during the chirp.

The label associated with each record contains the letter "R" if the object is a rock and "M" if it is a mine (metal cylinder).  The numbers in the labels are in increasing order of aspect angle, but they do not encode the angle directly.

* 1. **LIBRARIES USED**

1. Numpy
2. Pandas
3. Matplotlib
4. Sklearn
   * 1. ***Numpy*** :-

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It is open-source software. It contains various features including these important ones:

1. A powerful N-dimensional array object
2. Sophisticated (broadcasting) functions
3. Tools for integrating C/C++ and Fortran code
4. Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

Installation :

pip install numpy

* + 1. ***Pandas***:-

Pandas is an open-source library that is made mainly for working with relational or labeled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. This library is built on top of the NumPy library. Pandas is fast and it has high performance & productivity for users.

The first step of working in pandas is to ensure whether it is installed in the Python folder or not.  If not then we need to install it in our system usingpip command. Type cmd command in the search box and locate the folder using cd command where python-pip file has been installed.  After locating it, type the command:

pip install pandas

After the pandas have been installed into the system, you need to import the library. This module is generally imported as:

import pandas as pd

Here, pd is referred to as an alias to the Pandas. However, it is not necessary to import the library using the alias, it just helps in writing less amount code every time a method or property is called.

Pandas generally provide two data structures for manipulating data, They are:

1. Series
2. DataFrame
   * + 1. ***Series*** :-

Pandas Seriess is a one-dimensional labeled array capable of holding data of any type (integer, string, float, python objects, etc.). The axis labels are collectively called indexes. Pandas Series is nothing but a column in an excel sheet. Labels need not be unique but must be a hashable type. The object supports both integer and label-based indexing and provides a host of methods for performing operations involving the index.

In the real world, a Pandas Series will be created by loading the datasets from existing storage, storage can be SQL Database, CSV file, an Excel file. Pandas Series can be created from the lists, dictionary, and from a scalar value etc.

* + - 1. ***DataFrame*** :-

Pandas DataFrame is a two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns. Pandas DataFrame consists of three principal components, the data, rows, and columns.

In the real world, a Pandas DataFrame will be created by loading the datasets from existing storage, storage can be SQL Database, CSV file, an Excel file. Pandas DataFrame can be created from the lists, dictionary, and from a list of dictionaries, etc.

* + 1. ***Matplotlib*** :-

Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy. As such, it offers a viable open source alternative to MATLAB. Developers can also use matplotlib’s APIs (Application Programming Interfaces) to embed plots in GUI applications. A Python matplotlib script is structured so that a few lines of code are all that is required in most instances to generate a visual data plot. The matplotlib scripting layer overlays two APIs:

1. The pyplot API is a hierarchy of Python code objects topped by *matplotlib.pyplot*
2. An OO (Object-Oriented) API collection of objects that can be assembled with greater flexibility than pyplot. This API provides direct access to Matplotlib’s backend layers.

The pyplot API has a convenient MATLAB-style stateful interface. In fact, matplotlib was originally written as an open source alternative for MATLAB. The OO API and its interface is more customizable and powerful than pyplot, but considered more difficult to use. As a result, the pyplot interface is more commonly used, and is referred to by default in this article.

Understanding matplotlib’s pyplot API is key to understanding how to work with plots:

1. *matplotlib.pyplot.figure: Figure* is the top-level container. It includes everything visualized in a plot including one or more *Axes*.
2. *matplotlib.pyplot.axes: Axes* contain most of the elements in a plot**:***Axis, Tick, Line2D, Text,*etc., and sets the coordinates. It is the area in which data is plotted. Axes include the X-Axis, Y-Axis, and possibly a Z-Axis, as well.
   * 1. ***SKlearn*** :-

Scikit-learn (formerly scikits.learn and also known as sklearn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support-vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. Scikit-learn is a NumFOCUS fiscally sponsored project.

Scikit-learn is a library in Python that provides many unsupervised and supervised learning algorithms. It’s built upon some of the technology you might already be familiar with, like NumPy, pandas, and Matplotlib!

The functionality that scikit-learn provides include:

1. Regression, including Linear and Logistic Regression
2. Classification, including K-Nearest Neighbors
3. Clustering, including K-Means and K-Means++
4. Modelselection
5. Preprocessing, including Min-Max Normalization
   1. **ALGORITHMS USED :-**
   2. Logistic Regression
   3. Decision tree
   4. Linear Discriminant Analysis
   5. K-Nearest Neighbor
      1. ***Logistic Regression*** :-

Logistic Regression is one of the most simple and commonly used Machine Learning algorithms for two-class classification. It is easy to implement and can be used as the baseline for any binary classification problem. Its basic fundamental concepts are also constructive in deep learning. Logistic regression describes and estimates the relationship between one dependent binary variable and independent variables.

Logistic regression is a statistical method for predicting binary classes. The outcome or target variable is dichotomous in nature. Dichotomous means there are only two possible classes. For example, it can be used for cancer detection problems. It computes the probability of an event occurrence.

It is a special case of linear regression where the target variable is categorical in nature. It uses a log of odds as the dependent variable. Logistic Regression predicts the probability of occurrence of a binary event utilizing a logit function.

Linear Regression Equation:

http://res.cloudinary.com/dyd911kmh/image/upload/f_auto,q_auto:best/v1534281880/image1_ga8gze.png

***Fig 3.1 Linear Regression Equation***

Where, y is dependent variable and x1, x2 ... and Xn are explanatory variables.

* + 1. ***Decision tree*** :-

Decision Tree is a **Supervised learning technique**that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where**internal nodes represent the features of a dataset, branches represent the decision rules**and**each leaf node represents the outcome.** In a Decision tree, there are two nodes, which are the DecisionNode andLeafNode**.** Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches. The decisions or the test are performed on the basis of features of the given dataset. It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions***.***



***Fig 3.2 Structure of Decision Tree Example***

3.2.2.1 *How Decision Tree Algorithm Works* :-

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

* Step-1: Begin the tree with the root node, says S, which contains the complete dataset.
* Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM).
* Step-3: Divide the S into subsets that contains possible values for the best attributes.
* Step-4: Generate the decision tree node, which contains the best attribute.
* Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

3.2.2.2 *Attribute Selection Measures*

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as Attribute selection measure or ASM.By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

1. Information Gain
2. Gini Index

Information Gain :-

Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute. It calculates how much information a feature provides us about a class. According to the value of information gain, we split the node and build the decision tree. A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

Information Gain= Entropy(S)- [(Weighted Avg) \*Entropy(each feature)

Entropy**:** Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:

Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)

Where,

S= Total number of samples

P(yes)= probability of yes

P(no)= probability of no

Gini Index :-

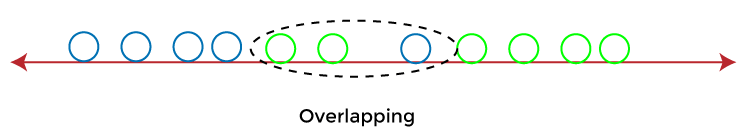
Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm. An attribute with the low Gini index should be preferred as compared to the high Gini index. It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits. Gini index can be calculated using the below formula:

Gini Index= 1- ∑jPj2

* + 1. ***Linear Discriminant Analysis*** :-

Linear Discriminant Analysis (LDA) is one of the commonly used dimensionality reduction techniques in machine learning to solve more than two-class classification problems. It is also known as Normal Discriminant Analysis (NDA) or Discriminant Function Analysis (DFA).

Linear Discriminant analysis is one of the most popular dimensionality reduction techniques used for supervised classification problems in machine learning. It is also considered a pre-processing step for modeling differences in ML and applications of pattern classification. Whenever there is a requirement to separate two or more classes having multiple features efficiently, the Linear Discriminant Analysis model is considered the most common technique to solve such classification problems. For e.g., if we have two classes with multiple features and need to separate them efficiently. When we classify them using a single feature, then it may show overlapping.



***Fig 3.3 Overlapping of classes***

To overcome the overlapping issue in the classification process, we must increase the number of features regularly.

Logistic Regression is one of the most popular classification algorithms that perform well for binary classification but falls short in the case of multiple classification problems with well-separated classes. At the same time, LDA handles these quite efficiently. LDA can also be used in data pre-processing to reduce the number of features, just as PCA, which reduces the computing cost significantly. LDA is also used in face detection algorithms. In Fisher faces, LDA is used to extract useful data from different faces. Coupled with eigenfaces, it produces effective results.

* + 1. ***K-Nearest Neighbor :-***

K-Nearest Neighbor is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems. K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data. It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset. KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

The K-NN working can be explained on the basis of the below algorithm:

Step-1: Select the number K of the neighbors

Step-2: Calculate the Euclidean distance of K number of neighbors

Step-3**:** Take the K nearest neighbors as per the calculated Euclidean distance.

Step-4: Among these k neighbors, count the number of the data points in each category.

Step-5: Assign the new data points to that category for which the number of the neighbor is maximum.

Step-6: Our model is ready.

* 1. **Code Snippets :-**
     1. ***importing necessary modules***

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from pickle import dump

from pickle import load

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import train\_test\_split

from sklearn.model\_selection import KFold

from sklearn.model\_selection import cross\_val\_score

from sklearn.model\_selection import GridSearchCV

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import accuracy\_score

from numpy import set\_printoptions

pd.set\_option('display.width', 100)

pd.set\_option('precision', 5)

%matplotlib inline

* + 1. ***Loading the Dataset :-***

filename = 'sonar data.csv'

df = pd.read\_csv(filename, header=None)

print("DataSet Loaded Successfully")

* + 1. ***Visualizing the dataset :-***

*Univariate Plots:-*

# Histograms

df.hist(figsize=(16,12));

# Density plots

df.plot(kind='density',subplots=True,layout=(8,8),sharex=False,figsize=(18, 14));

*Multi Variate Plots:-*

correlations = df.corr()

fig = plt.figure(figsize=(18,16))

ax = fig.add\_subplot(111)

cax = ax.matshow(correlations, vmin=-1, vmax=1, interpolation='none')

fig.colorbar(cax)

ticks = np.arange(0, 60, 1)

ax.set\_xticks(ticks)

ax.set\_yticks(ticks)

plt.show()

* + 1. ***Logistic Regression Model training :-***

model = LogisticRegression()

model.fit(X\_train, Y\_train)

X\_train\_prediction = model.predict(X\_train)

training\_data\_accuracy = accuracy\_score(X\_train\_prediction, Y\_train)

X\_test\_prediction = model.predict(X\_test)

test\_data\_accuracy = accuracy\_score(X\_test\_prediction, Y\_test)

* + 1. ***Decision tree Model Training:-***

X\_train, X\_test,y\_train, y\_test = train\_test\_split(X,Y,test\_size=0.1, random\_state=1)

classifier = DecisionTreeClassifier(criterion='entropy', max\_depth=3, random\_state=0)

classifier.fit(X\_train, y\_train)

X\_train\_prediction = classifier.predict(X\_train)

train\_data\_accuracy = accuracy\_score(X\_train\_prediction,y\_train)

X\_test\_prediction = classifier.predict(X\_test)

test\_data\_accuracy = accuracy\_score(X\_test\_prediction,y\_test)

* + 1. ***K-Nearest Neighbor Model Training:-***

X\_train, X\_test,y\_train, y\_test = train\_test\_split(X,Y,test\_size=0.3)

knn = KNeighborsClassifier(n\_neighbors=5, metric='euclidean')

knn.fit(X\_train, y\_train)

X\_train\_prediction = knn.predict(X\_train)

train\_data\_accuracy=accuracy\_score(y\_train,X\_train\_prediction)

X\_test\_prediction = knn.predict(X\_test)

test\_data\_accuracy=accuracy\_score(y\_test,X\_test\_prediction)

confusion\_matrix(y\_test, X\_test\_prediction)

* + 1. ***Linear Discriminant Analysis Model Training:-***

X\_train, X\_test,y\_train, y\_test = train\_test\_split(X,Y,test\_size=0.1)

lda = LinearDiscriminantAnalysis()

lda.fit(X\_train,y\_train)

x\_train\_prediction = lda.predict(X\_train)

print(accuracy\_score(y\_train,x\_train\_prediction))

x\_test\_prediction = lda.predict(X\_test)

print(accuracy\_score(y\_test,x\_test\_prediction))

confusion\_matrix(y\_test,x\_test\_prediction)

* + 1. ***Making a Predictive System:-***

input\_data=(0.0307,0.0523,0.0653,0.0521,0.0611,0.0577,0.0665,0.0664,0.1460,0.2792,0.3877,0.4992,0.4981,0.4972,0.5607,0.7339,0.8230,0.9173,0.9975,0.9911,0.8240,0.6498,0.5980,0.4862,0.3150,0.1543,0.0989,0.0284,0.1008,0.2636,0.2694,0.2930,0.2925,0.3998,0.3660,0.3172,0.4609,0.4374,0.1820,0.3376,0.6202,0.4448,0.1863,0.1420,0.0589,0.0576,0.0672,0.0269,0.0245,0.0190,0.0063,0.0321,0.0189,0.0137,0.0277,0.0152,0.0052,0.0121,0.0124,0.0055)

# 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=knn.predict(input\_data\_reshaped)

print(prediction)

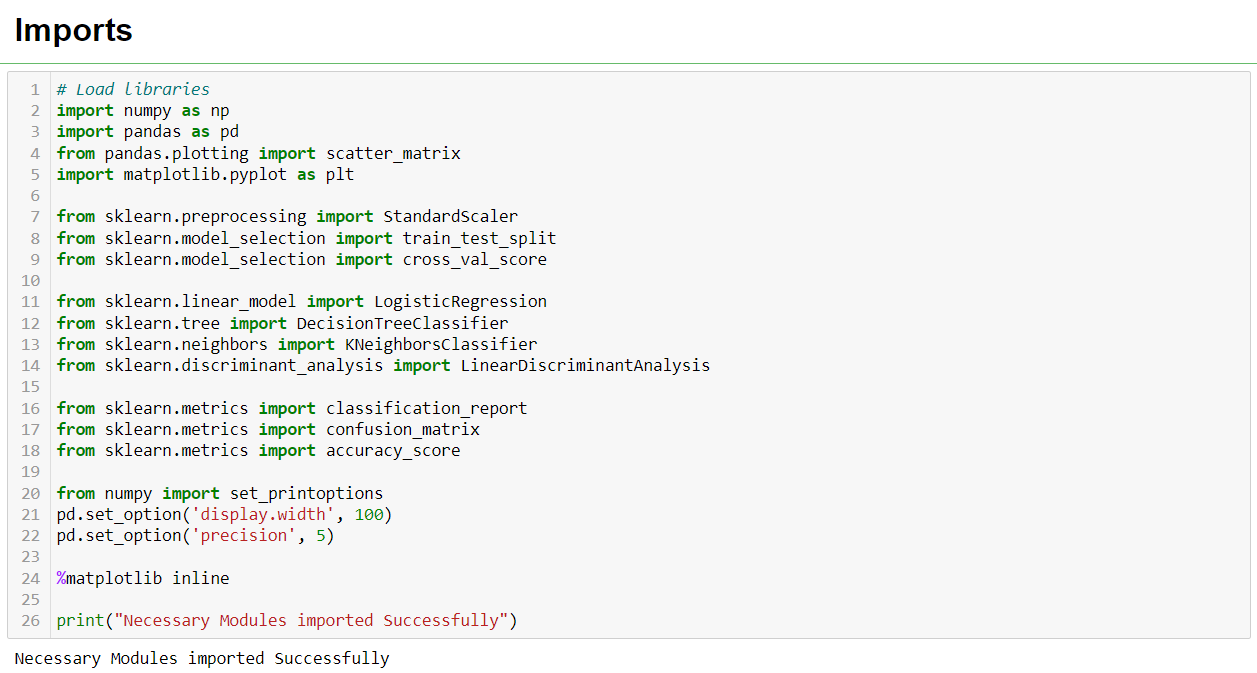
if (prediction[0]=='0'):

print('The object is a Rock')

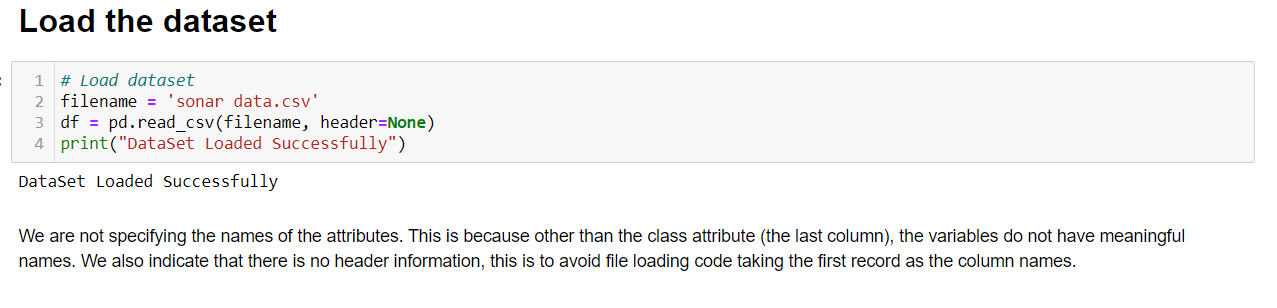
else:

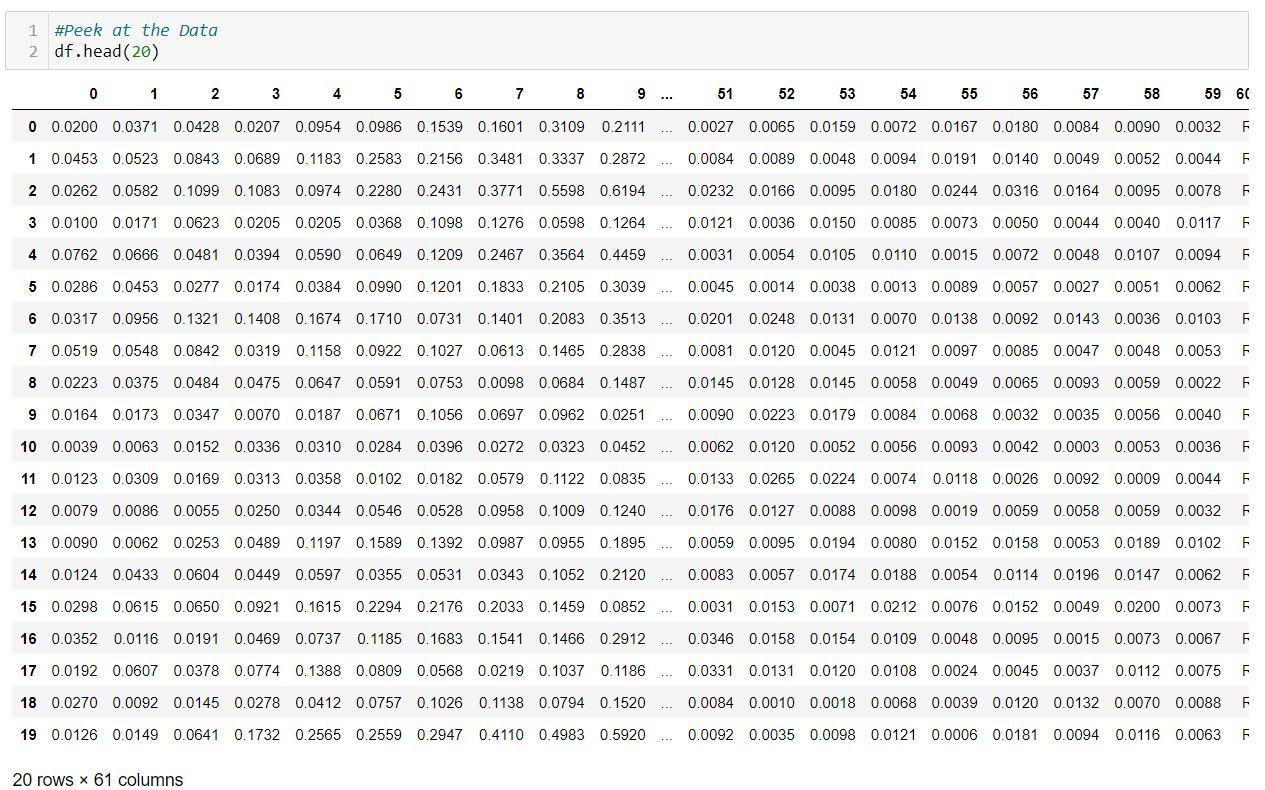
print('The object is a mine')

* 1. **Screenshots:-**

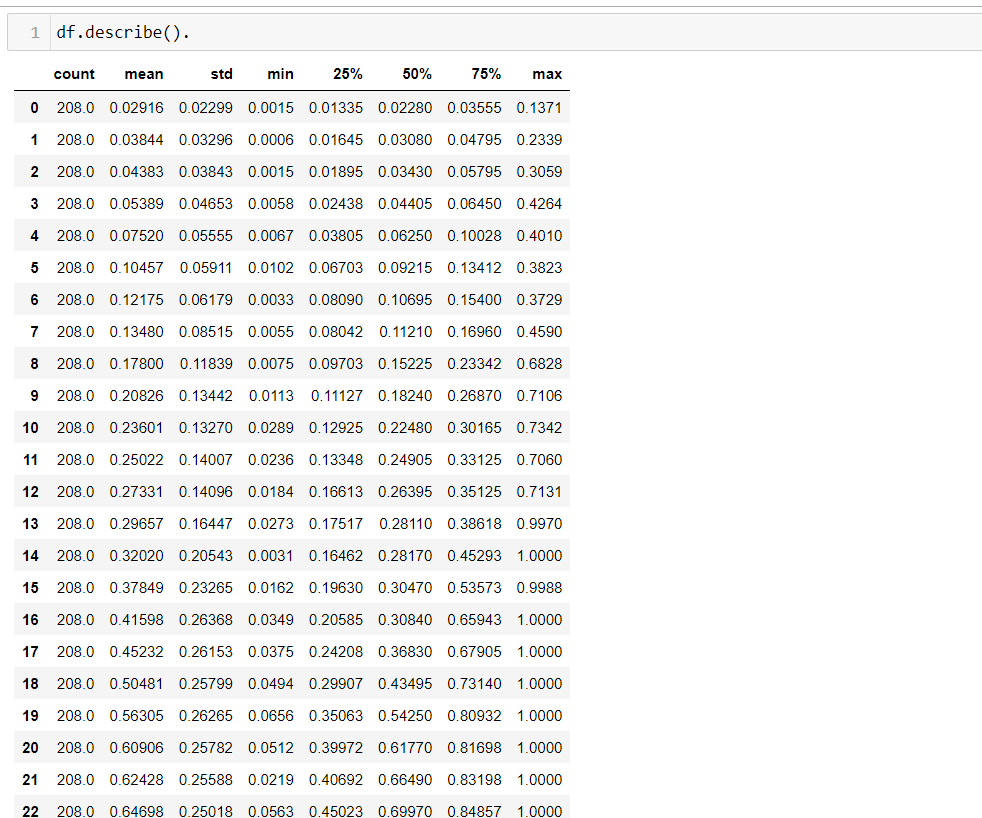
****

***Fig 3.1 Importing modules***

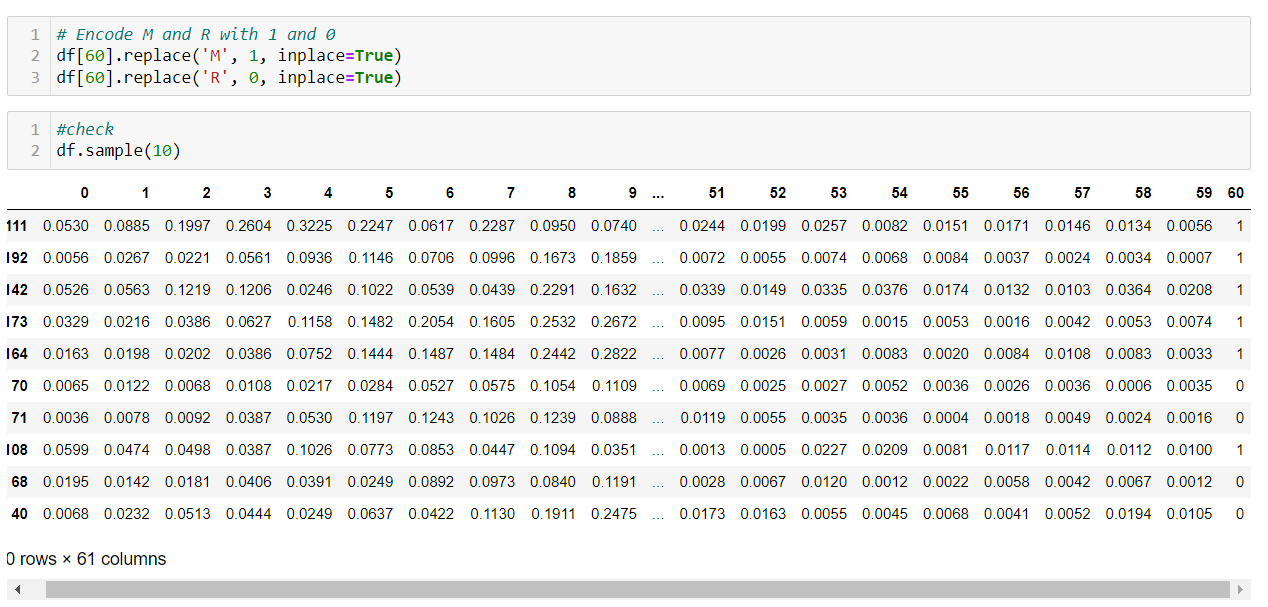
** *Fig 3.2 Loading Dataset***

****

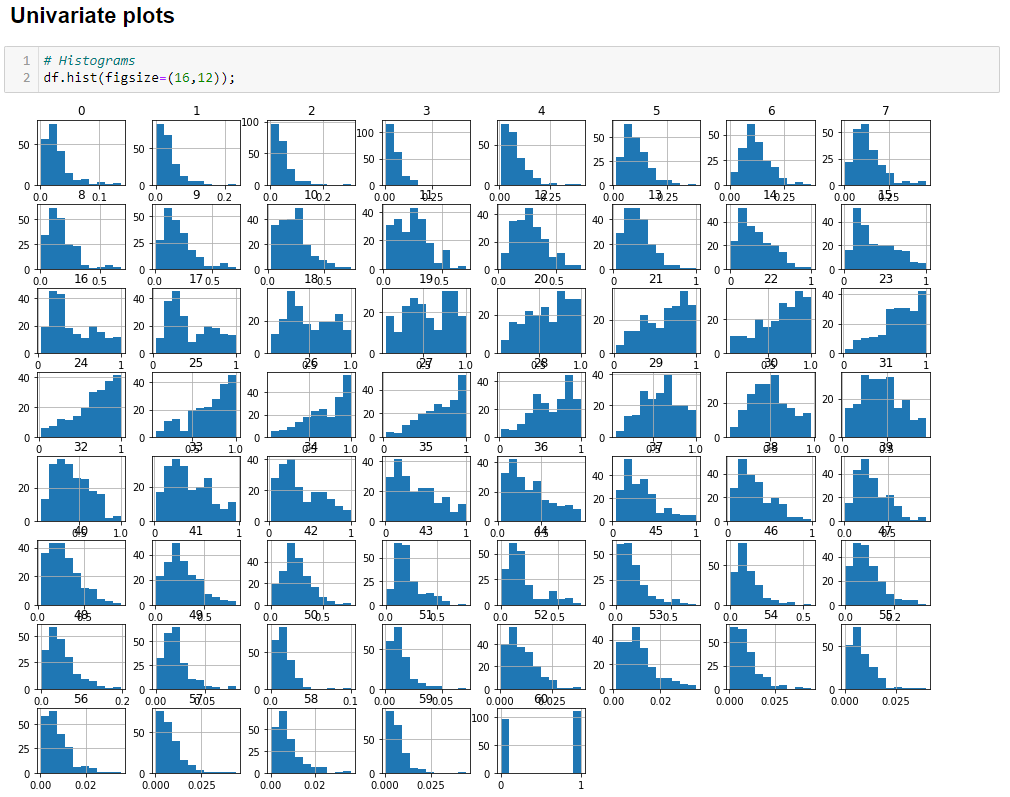
***Fig 3.3 Summarize dataset***

****

***Fig 3.4 Describing Data Statistically***

****

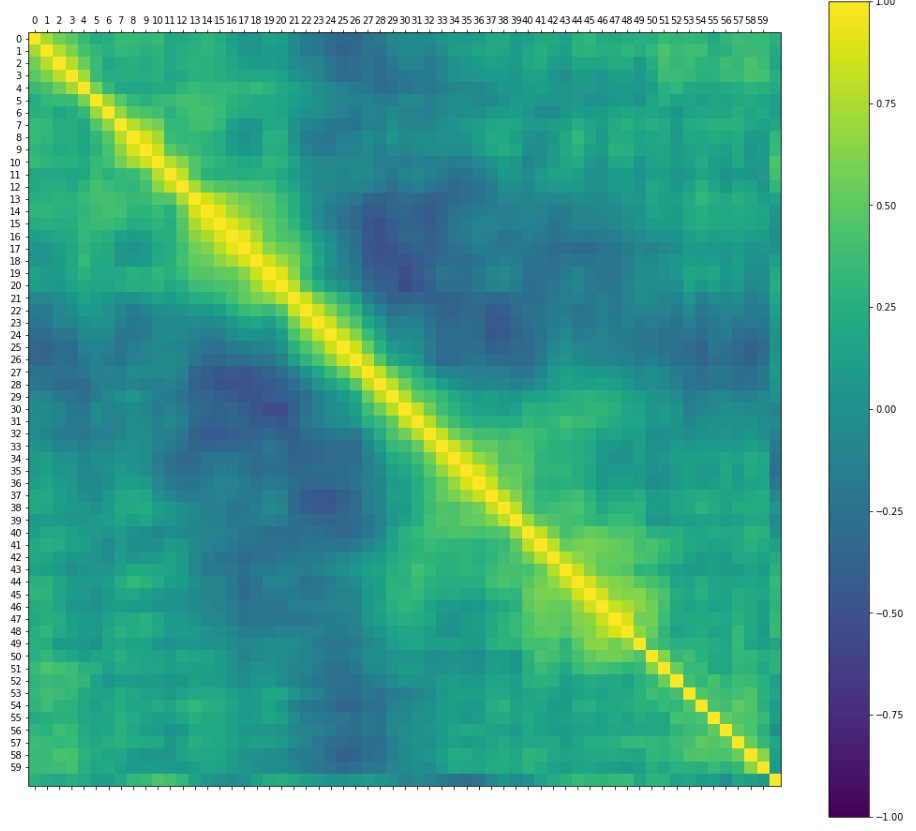
***Fig 3.5 Replacing M,R with 0,1***

****

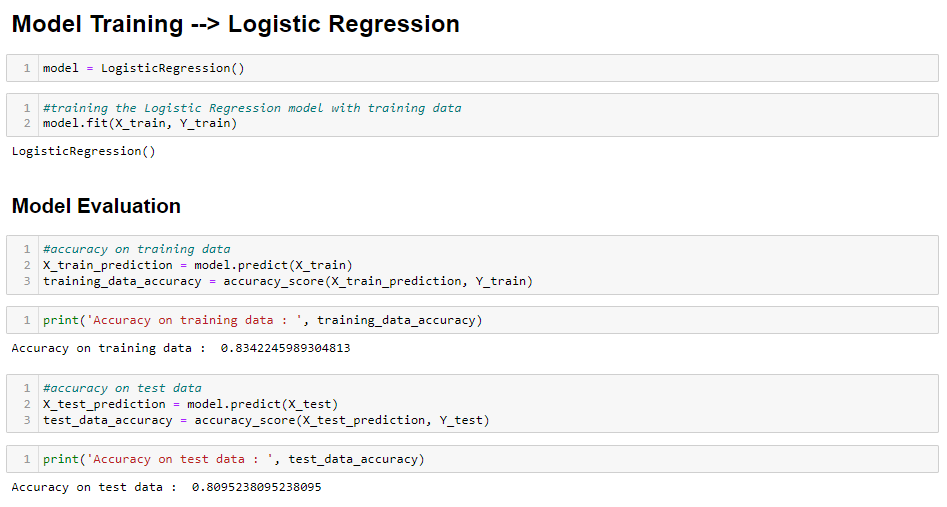
***Fig 3.6 univariate plot***

****

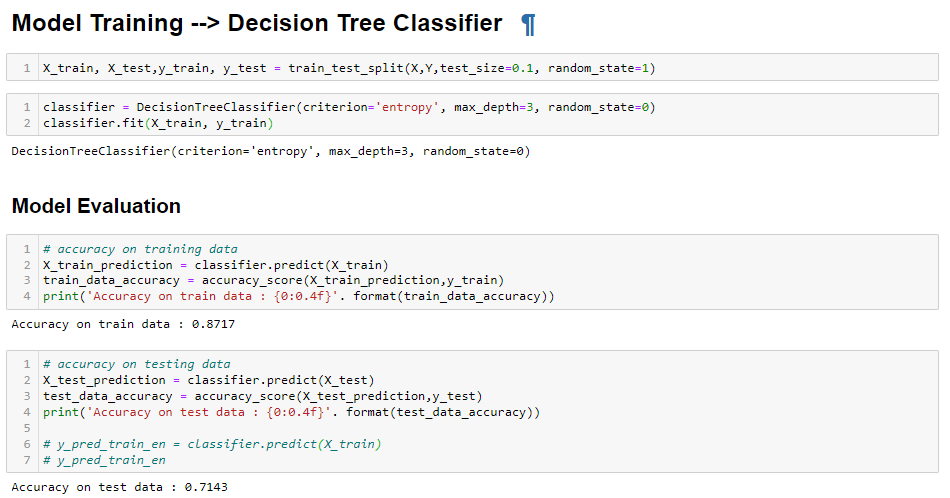
***Fig 3.7 Density Plots***

****

***Fig 3.8 Multivariate Plot***

****

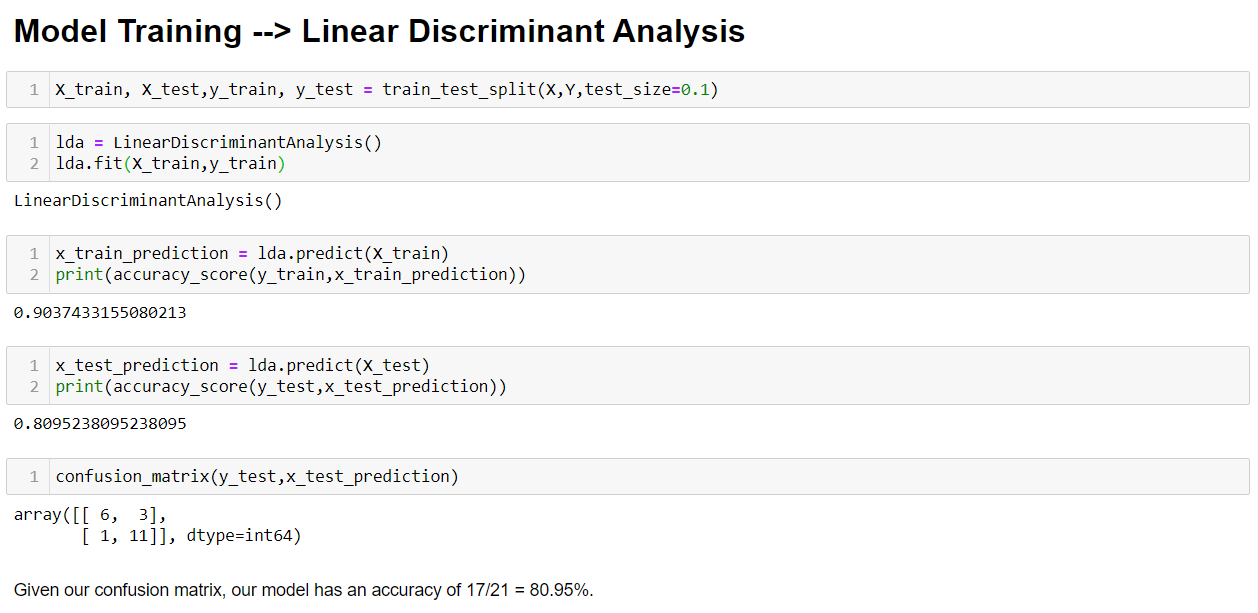
***Fig 3.9 Logistic Regression***

****

***Fig 3.10 Decision Tree Classifier***

****

***Fig 3.11 K-Nearest Neighbor***

****

***Fig 3.12 Linear Discriminant Analysis***

****

***Fig 3.13 Predictive System***

**CHAPTER - 4**

1. **CONCLUSION**

Our project “Rock vs Mine prediction by the evaluation of machine learning algorithms” are used to detect rocks and mines in the ocean bed. Naval mines are an effective method for blocking ships and restricting naval operations which result in significant negative economic and environmental impacts. There are two existing ways to detect a mine, one by using sonar signals and the other by using manpower. Using Sonar signals has been a better option as the risk for the latter is more. The data is collected and stored in a CSV file. By using different machine learning techniques we can observe and understand the nature of the predictive system. By the evaluation of algorithms, we get to check and compare the accuracies to build a better performing prediction model. A python is open-source software and the machine computation is also faster than many others and the cost might decrease dependently. Through this project, we want to make the process a bit easy and simple to achieve and use.

**BIBLIOGRAPHY:-**

<https://numpy.org/doc/stable/>

<https://pandas.pydata.org/docs/>

<https://matplotlib.org/stable/api/index>

<https://scikit-learn.org/stable/modules/classes.html>

<https://www.datacamp.com/tutorial/decision-tree-classification-python>

<https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761>

<https://towardsdatascience.com/lda-linear-discriminant-analysis-how-to-improve-your-models-with-supervised-dimensionality-52464e73930f>

<https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148>