**PROJECT REPORT ON MACHINE LEARNING**

Submitted By:

Name: Sukhpal

M. No: 9643636882

Email ID: [sukhpal.du.or.20@gmail.com](mailto:sukhpal.du.or.20@gmail.com)

Submitted To:

Company Name: Credicxo

Email ID: [hr@credicxo.com](mailto:hr@credicxo.com)

**Abstract:**

In this project, we were asked to experiment with a given dataset, and to explore how machine learning algorithms can be used to find the patterns in data. We were expected to gain experience using a common machine learning library and were expected to submit a report about the dataset and the algorithms used. After performing the required tasks on a dataset of my choice, herein lies my final report.

**Introduction:**

**Machine Learning** is the field of study that gives computers the capability to learn without being explicitly programmed. ML is one of the most exciting technologies that one would have ever come across. As it is evident from the name, it gives the computer that which makes it more similar to humans: The ability to learn. Machine learning is actively being used today, perhaps in many more places than one would expect.

**Deep learning** is a subset of machine learning in artificial intelligence (AI) that has networks capable of learning unsupervised from data that is unstructured or unlabeled. Also known as deep neural learning or deep neural network.

**Dataset:**

The given dataset contains details about organic chemical compounds including their chemical features, isomeric conformation, names and the classes in which they are classified. The compounds are classified as either ‘Musk’ or ‘Non-Musk’ compounds. The dataset has 170 columns and 6598 observations.

**Language used:-**

Python language: Python is an interpreted , high-level , general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python’s design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear , logical code for small and large-scale projects.

**Softwares used:**

1. Spyder (Python 3.7)
2. Jupyter Notebook

**Library used:**

**pandas :** pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming.

**numpy:** numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

**matplotlib.pyplot:** matplotlib.pyplot is a collection of command style functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

**sklearn.model\_selection.train\_test\_split:** splitting a dataset into training set and test set.

Python Code and Results:

# Part 1 - Data Preprocessing

# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

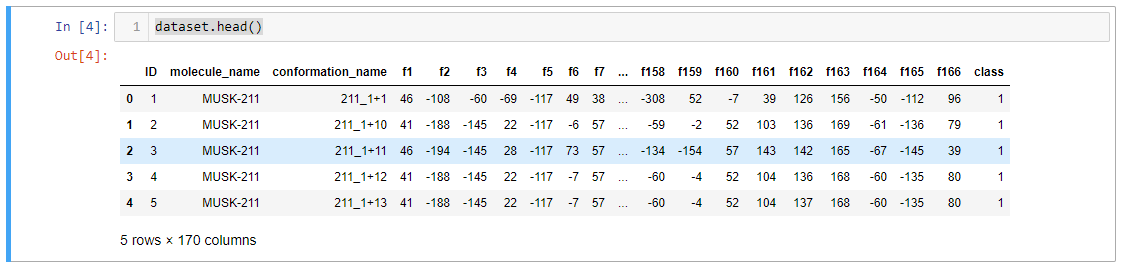
# Importing the dataset

dataset = pd.read\_csv('musk\_csv.csv')

X = dataset.iloc[:, 3:-1].values

y = dataset.iloc[:, -1].values

dataset.head()



“To take a review of given data on that I will apply Machine Learning Algorithms.”

dataset.info()



“For checking no. of columns and no. of rows and data types of each column”

dataset['class'].value\_counts()



“To count values categorically”

print(dataset.isnull().sum())

“To check null values in dataset. There is no null value in dataset.”

# Splitting the dataset into the Training set and Test set

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 0)

“To split in a 80:20 ratio for training and validation datasets.”

# Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

“To Standardization of data”

# Part 2 - Now let's make the ANN!

# Importing the Keras libraries and packages

import keras

from keras.models import Sequential

from keras.layers import Dense

“Keras is a powerful and easy-to-use free open source **Python** library for developing and evaluating deep learning models. It wraps the efficient numerical computation libraries Theano and TensorFlow and allows you to define and train neural network models in just a few lines of code.”

# Initialising the ANN

classifier = Sequential()

“Artificial neural networks or connectionist systems are computing systems vaguely inspired by the biological neural networks that constitute animal brains. Such systems "learn" to perform tasks by considering examples, generally without being programmed with task-specific rules.

I think ANN is most appropriate algorithm for this type of data So I am choosing this“

# Adding the input layer and the first hidden layer

classifier.add(Dense(output\_dim = 83, init = 'uniform', activation = 'relu', input\_dim = 166))

“A dense layer is a classic fully connected neural network layer : each input node is connected to each output node. A dropout layer is similar except that when the layer is used, the activations are set to zero for some random nodes. This is a way to prevent overfitting.”

# Adding the second hidden layer

classifier.add(Dense(output\_dim = 83, init = 'uniform', activation = 'relu'))

# Adding the output layer

classifier.add(Dense(output\_dim = 1, init = 'uniform', activation = 'sigmoid'))

# Compiling the ANN

classifier.compile(optimizer = 'adam', loss = 'binary\_crossentropy', metrics = ['accuracy'])

# Fitting the ANN to the Training set

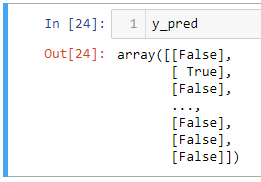
history=classifier.fit(X\_train, y\_train, validation\_split =0.2, epochs=100)

# Part 3 - Making the predictions and evaluating the model

# Predicting the Test set results

y\_pred = classifier.predict(X\_test)

y\_pred = (y\_pred > 0.5)

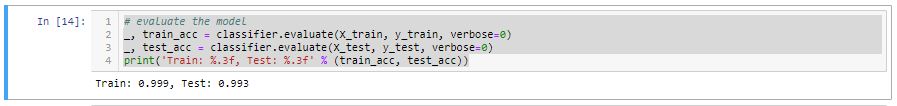


# evaluate the model

\_, train\_acc = classifier.evaluate(X\_train, y\_train, verbose=0)

\_, test\_acc = classifier.evaluate(X\_test, y\_test, verbose=0)

print('Train: %.3f, Test: %.3f' % (train\_acc, test\_acc))



“Testing Accuracy of the model”

# Plot loss during training

plt.subplot(211)

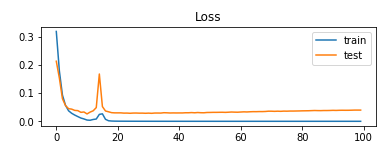
plt.title('Loss')

plt.plot(history.history['loss'], label='train')

plt.plot(history.history['val\_loss'], label='test')

plt.legend()

plt.show()



# Plot accuracy during training

plt.subplot(212)

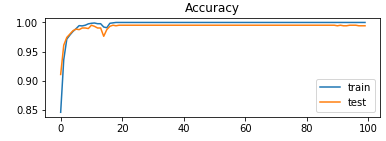
plt.title('Accuracy')

plt.plot(history.history['acc'], label='train')

plt.plot(history.history['val\_acc'], label='test')

plt.legend()

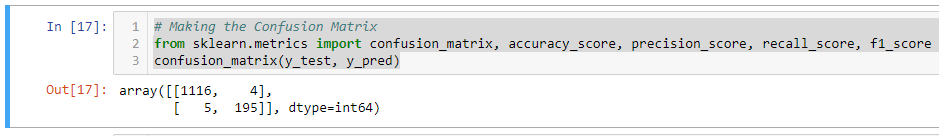
plt.show()



# Making the Confusion Matrix

From sklearn.metrics import confusion\_matrix, accuracy\_score, precision\_score, recall\_score, f1\_score

confusion\_matrix(y\_test, y\_pred)

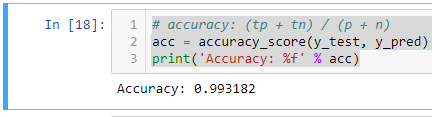


“A confusion matrix is a table that is often used to describe the performance of a classification model (or “classifier”) on a set of test data for which the true values are known. It allows the visualization of the performance of an algorithm.”

# accuracy: (tp + tn) / (p + n)

acc = accuracy\_score(y\_test, y\_pred)

print('Accuracy: %f' % acc)

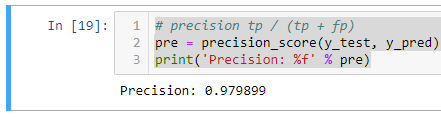


“**Accuracy** is one metric for evaluating classification models. Informally, **accuracy** is the fraction of predictions our model got right. Formally, **accuracy** has the following definition: **Accuracy** = Number of correct predictions / Total number of predictions.”

# precision tp / (tp + fp)

pre = precision\_score(y\_test, y\_pred)

print('Precision: %f' % pre)

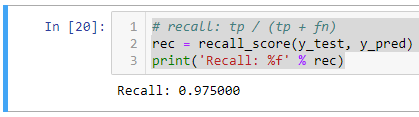


“In pattern recognition, information retrieval and classification (machine learning), precision (also called positive predictive value) is the fraction of relevant instances among the retrieved instances.”

# recall: tp / (tp + fn)

rec = recall\_score(y\_test, y\_pred)

print('Recall: %f' % rec)

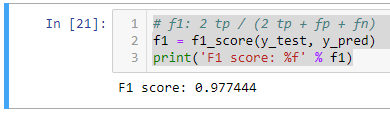


“recall (also known as [sensitivity](https://en.wikipedia.org/wiki/Sensitivity_and_specificity)) is the fraction of the total amount of relevant instances that were actually retrieved. Both precision and recall are therefore based on an understanding and measure of [relevance](https://en.wikipedia.org/wiki/Relevance).”

# f1: 2 tp / (2 tp + fp + fn)

f1 = f1\_score(y\_test, y\_pred)

print('F1 score: %f' % f1)



“The F1 Score is the 2\*((precision\*recall)/(precision+recall)). It is also called the F Score or the F Measure. Put another way, the F1 score conveys the balance between the precision and the recall.”

**Conclusion:**

First of all, I understood data what actually data is doing and make decisions and apply some data preprocessing algorithms that are important and the I apply Artificial Neural Network (ANN) Algorithm of Deep Learning and check Accuracy of the model and make some graphs that are requirements of the task. So the model accuracy I got, is 0.993182, precision is 0.979899, recall 0.975000 and f1\_score is 0.977444.

Thank You