B.TECH PROJECT REPORT

Predicting seismic hazard in an Underground mine using Machine Learning and Deep Learning Algorithms

Under the guidance of:

Dr. Nawal Kishore
(Assistant Professor)
(Dept. of Mining Engineering IIT BHU Varanasi)

Submitted by:

Nikhil Kumar
(Roll No. 17155051)
(B.Tech Part 3)
(Dept. of Mining Engineering IIT BHU Varanasi)

Department of Mining Engineering
IIT (BHU), Varanasi
Session: 2019- 2020

Acknowledgements

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

I am highly indebted to **Dr. Nawal Kishore** for their guidance and constant supervision as well as for providing necessary information regarding the project and useful critiques of this research work even at the time of global pandemic COVID19 through virtual meetings and telephonic conversations.

I would like to express my gratitude towards my parents for their kind co-operation and encouragement which help me in completion of this project.

I would like to express my special gratitude and thanks to industry persons for giving me such attention and time.

My thanks and appreciations also go to my colleague in developing the project and people who have willingly helped me out with their abilities.

INDIAN INSTITUTE OF TECHNOLOGY (BHU), VARANASI

CERTIFICATE

This is to certify that Nikhil Kumar (Roll no. 17155051), a student of Department of Mining Engineering- Batch 2017, has successfully completed the B.Tech project "**Predicting seismic hazard in an Underground mine using Machine Learning and Deep Learning Algorithms**" under my guidance during the session 2019- 2020.

Date: Dr. Nawal Kishore
IIT (BHU), Varanasi

Contents

Abstract

When an event or accident occurs suddenly and unexpectedly, a crisis requires immediate attention. Despite technological advances, natural disasters such as earthquakes, floods, avalanches, hurricanes, volcanoes, and fires are all the more extraordinary. Mining operations are always associated with risks associated with mine. Earthquake danger in underground mines is a threat to human life. Seismic hazards are more difficult to detect than the natural hazards of earthquakes. Advanced seismic seismicity and the use of seismo acoustic prediction. The accuracy of the information generated is not correct.

Seismology has not yet determined the magnitude of the earthquakes; How and when, it should search for the best model for predicting seismic oceans. This paper describes the statistical rules of an experiment for testing seismic forecasts. The primary objectives of the tests described by B-Lo are to estimate physical models for earthquakes, to ensure that the sources of earthquake hazard and risk studies are compatible with seismic data, and to provide quantitative measures to weight the model, which may or may not be assigned. Estimated to fit specific areas.

Introduction

What are Seismic Hazards?

An earthquake hazard is the probability of earthquakes occurring in a given geographical area, in a given window, and in a given range with the intensity of the earth's velocity. Therefore, with the perceived threat, risk can be assessed and included in areas such as building codes for standard buildings, large buildings and infrastructure projects, land use plans, and determining insurance rates. . Two standard measurements of ground velocity derived from seismic risk studies are also briefly MCE.

The most common potential maximum earthquakes (or events) used in standard building codes, and the more detailed and decisive maximum fidelity earthquakes, are included in the design of large buildings and in civilian infrastructure such as dams or bridges. It is important to clarify which MCE is being discussed.

Causes and Distributions of Seismic Hazard :

Evolution has been observed in seismic risk analysis for the past few decades and is a perspective demonstrated on current issues and new developments. For the past 30 years, there have been methods and data sophisticated, but our overall understanding of seismic hazards in many areas applies to specific buildings signs haven't changed much. Our seismic risk zoning maps are very simple and transparent the result of historical seismic models. Over time, we have improved our understanding of where we are and why earthquakes occur and as a result land movements and their

characteristic improvement opportunities.
We are beginning to understand the important role of uncertainty in seismic risk analysis.
However, there are still significant drawbacks to our uncertainty treatment. Lack of knowledge our uncertainty of that threat also prevents that uncertainty from being properly determined. On Current and new geo-motion data are available in real-time to provide better understanding of earthquakes ground Speed Construction and Propagation and Creating Ground Work for Real Time Earthquake Hazard information system that will be developed in the future.

Detection and Advancements :

Developments in seismic hazard analysis over the last few decades are overviewed, and a perspective is presented on current issues and new developments. Over the last 30 years, methods and data have been refined, but our overall understanding of seismic hazards in most regions, as applied to typical building codes, has not changed very much. Our seismic hazard zoning maps are a relatively simple and transparent consequence of the patterns of historical seismicity. Over time, we have refined our understanding of where and why earthquakes occur, and improved our characterization of the resulting ground motions and their probabilities. We have begun to understand the important role of uncertainty in seismic hazard analysis. However, there are still significant shortcomings in our treatment of uncertainty. The same lack of knowledge that causes our uncertainty of the hazard also prevents us from accurately quantifying that uncertainty. At present, new ground-motion data, available in near-real time, are allowing better insight into earthquake ground motion generation and propagation, and are laying the groundwork for real-time seismic hazard information systems that will be developed in the future.

Objective

The objective of this project is to forecast the likelihood of seismic hazard in the next shift (8 hours period) using Machine learning and Deep Learning Algorithms given the seismic data of previous shifts.

Why Machine Learning?

Seismic hazeard is the hardest detectable and predictable natural hazard as compared to other hazards like hurricane, cyclone etc.
There are many advanced seismic and seismoacoustic monitoring systems that help us better understand the rock mass process and hazards but there accuracy is very low. Complexity of seismic processes and big differences between the number of low-energy and high energy (greater than 10000 joules) seismic events causes statistical techniques to be insufficient to predict seismic hazard.
So, we need to find better methods for hazard detections which has greater accuracy and using Machine Learning methods we can do that.

Machine Learning and Classifications

What is Machine Learning?

Machine Learning is a application of Artificial Intelligence which provides the systems an ability to learn and improve from experience without being explicitly programmed. It's algorithms use statistics to find patterns in massive amounts of data.

How to Classify using Machine Learning?

In machine learning, classification is a supervised learning concept which basically categorizes a set of data into classes. There are many different types of classification tasks that you may encounter in machine learning and specialized approaches to modeling that may be used for each.From a modeling perspective, classification requires a training dataset with many examples of inputs and outputs from which to learn. In this Project we use classification techniques of Machine Learning which are explained Later.

Dataset and Description

The dataset used for this project is Seismic-Bumps Dataset and it desribes the problem of high energy (higher than 10^4 J) seismic bumps forecasting in a coal mine. Data come from two of longwalls located in a Polish coal mine. This data was generated by Silesian University of Technology, Poland and Institute of Innovative Technologies EMAG, Poland.

Features description :

The multivariate dataset here contains 19 different features and 2584 instances of this features. The available features are real and categorical forms.

Features are described as:

1. **Seismic**: result of shift seismic hazard assessment in the mine working obtained by the seismic method (a - lack of hazard, b - low hazard, c - high hazard, d - danger state)
2. **Seismoacoustic**: result of shift seismic hazard assessment in the mine working obtained by the seismoacoustic method.
3. **Shift**: information about type of a shift (W - coal-getting shift, N -preparation shift).
4. **genergy**: seismic energy recorded within previous shift by the most active geophone (GMax) out of geophones monitoring the longwall.
5. **gpuls**: a number of pulses recorded within previous shift by GMax.
6. **gdenergy**: a deviation of energy recorded within previous shift by GMax from average energy recorded during eight previous shifts.
7. **gdpuls**: a deviation of a number of pulses recorded within previous shift by GMax from average number of pulses recorded during eight previous shifts.

8. **ghazard**: result of shift seismic hazard assessment in the mine working obtained by the seismoacoustic method based on registration coming form GMax only.
9. **nbumps**: the number of seismic bumps recorded within previous shift.
10. **nbumps2**: the number of seismic bumps (in energy range [10^2,10^3)) registered within previous shift.
11. **nbumps3**: the number of seismic bumps (in energy range [10^3,10^4)) registered within previous shift.
12. **nbumps4**: the number of seismic bumps (in energy range [10^4,10^5)) registered within previous shift.
13. **nbumps5**: the number of seismic bumps (in energy range [10^5,10^6)) registered within the last shift.
14. **nbumps6**: the number of seismic bumps (in energy range [10^6,10^7)) registered within previous shift.
15. **nbumps7**: the number of seismic bumps (in energy range [10^7,10^8)) registered within previous shift.
16. **nbumps89**: the number of seismic bumps (in energy range [10^8,10^10)) registered within previous shift.
17. **Energy**: total energy of seismic bumps registered within previous shift.
18. **Maxenergy**: the maximum energy of the seismic bumps registered within previous shift.
19. **Class**: the decision attribute - '1' means that high energy seismic bump occurred in the next shift ('hazardous state'), '0' means that no high energy seismic bumps occurred in the next shift.

Balanced and Unbalanced Data :

A balanced data set is a set that contains all elements observed in all time frame. Whereas unbalanced data is a set of data where certain years, the data category is not observed.The presented data set is characterized by unbalanced distribution of positive and negative examples. In the data set there are only 170 positive examples representing class 1. An imbalanced datset provides us a very good accuracy only if it is handled properly using different

metrics and graphs like ROC-AUC curve, F1 score etc.

Machine Learning Modelling

This project is coded using Python Language on Google Colaboratory platform. For performing deep Learning algorithm we used TensorFlow as a backend.

Importing required Libraries :

In our first step we import required libraries which mainly include: Pandas for data handling, Numpy for numerical computations, seaborn and matplotlib for visualisation purposes, sklearn for machine learning modelling and keras for deep learning modelling.

```
##--CODE--##
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasClassifier
#from keras.utils import np_utils

from keras import optimizers
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import roc_auc_score, roc_curve, accuracy_score

%matplotlib inline

##--OUTPUT--##
Using TensorFlow backend.
```

Reading and describing the dataset :

Here we read the dataset which is in .txt format with the help of pandas on a seismic_df dataframe. We also print a snippet of data.

```
##--CODE--##
path='drive/My Drive/Datasets/'
seismic_df = pd.read_csv(path+'dataset.txt')
seismic_df.head()
```

##--OUTPUT--##

	seismic	seismoacoustic	shift	genergy	gpul	gdenergy	gdpuls	ghazard	nbumps	nbumps2	nbumps3	nbumps4	nbumps5	nbumps6	nbumps7	nbumps89	energy	maxenergy	class
0	a	a	N	15180	48	-72	-72	a	0	0	0	0	0	0	0	0	0	0	0
1	a	a	N	14720	33	-70	-79	a	1	0	1	0	0	0	0	0	2000	2000	0
2	a	a	N	8050	30	-81	-78	a	0	0	0	0	0	0	0	0	0	0	0
3	a	a	N	28820	171	-23	40	a	1	0	1	0	0	0	0	0	3000	3000	0
4	a	a	N	12640	57	-63	-52	a	0	0	0	0	0	0	0	0	0	0	0

We find the values like count, mean, deviation, min, max and quartiles. Which helps us to quickly analyse all the numerical features.

```
##--CODE--##
#checking for data shape and data type information
seismic_df.shape
seismic_df.info()
#checking for Missing Values
seismic_df.isnull().sum()
#description of the dataset
seismic_df.describe()
```

##--OUTPUT--##

	genergy	gpul	gdenergy	gdpuls	nbumps	nbumps2	nbumps3	nbumps4	nbumps5	nbumps6	nbumps7	nbumps89	energy	maxenergy	class
count	2.584000e+03	2584.000000	2584.000000	2584.000000	2584.000000	2584.000000	2584.000000	2584.000000	2584.000000	2584.0	2584.0	2584.0	2584.000000	2584.000000	2584.000000
mean	9.024252e+04	538.579334	12.375774	4.508901	0.859520	0.393576	0.392802	0.067724	0.004644	0.0	0.0	0.0	4975.270898	4278.850619	0.065789
std	2.292005e+05	562.652536	80.319051	63.166556	1.364616	0.783772	0.769710	0.279059	0.068001	0.0	0.0	0.0	20450.833222	19357.454882	0.247962
min	1.000000e+02	2.000000	-96.000000	-96.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.0	0.000000	0.000000	0.000000
25%	1.166000e+04	190.000000	-37.000000	-36.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.0	0.000000	0.000000	0.000000
50%	2.548500e+04	379.000000	-6.000000	-6.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.0	0.000000	0.000000	0.000000
75%	5.283250e+04	669.000000	38.000000	30.250000	1.000000	1.000000	1.000000	0.000000	0.000000	0.0	0.0	0.0	2600.000000	2000.000000	0.000000
max	2.595650e+06	4518.000000	1245.000000	838.000000	9.000000	8.000000	7.000000	3.000000	1.000000	0.0	0.0	0.0	402000.000000	400000.000000	1.000000

Handling Non-numerical features :

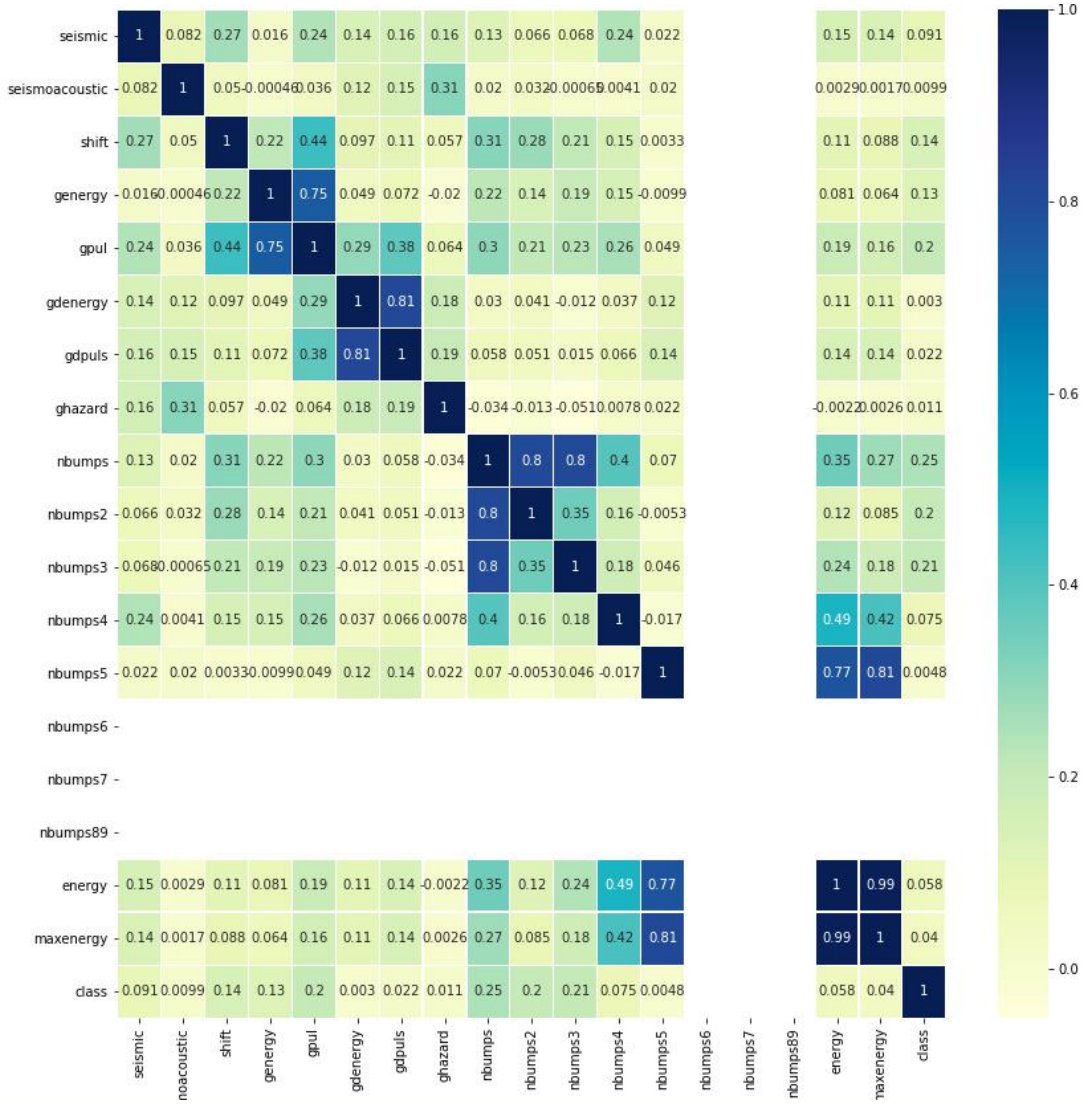
Since python cannot work with non-numerical features and from the previous code we get that some features are not in numerical form and that will create a loss of data. So, we convert those features in numerical form so that computations can be done on them.

```
##--CODE--##
def handle_non_numerical_data(df):
    columns = df.columns

    for column in columns:
        text_digit_vals = {}
        def convert_to_int(val):
            return text_digit_vals[val]

        if df[column].dtype != np.int64 and df[column].dtype != np.float64:
            column_contents = df[column].values.tolist()
            unique_elements = set(column_contents)
            x = 0
            for unique in unique_elements:
                if unique not in text_digit_vals:
                    text_digit_vals[unique] = x
                    x+=1

            df[column] = list(map(convert_to_int, df[column]))

    return df

seismic_df=handle_non_numerical_data(seismic_df)
```

Checking the correlation between the features and class :

We calculate the correlation value i.e, how any feature is correlated with another feature. It helps us to find important features which will be considered in the model.

```
##--CODE--##
plt.figure(figsize=(14,14))
corr_mat=seismic_df.corr()
sns.heatmap(corr_mat,cmap='YlGnBu',annot=True,linewidths=0.2)
plt.show()


##--OUTPUT--##
```

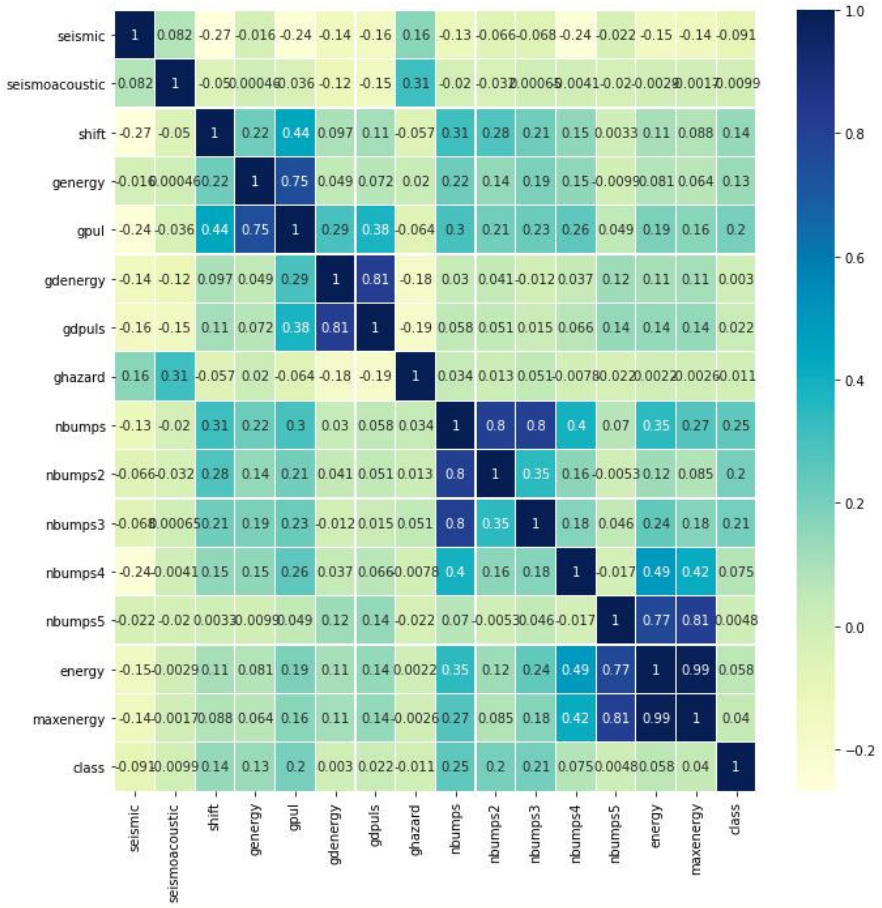


Removing unnecessary columns :

From previous correlation table we see that some values have zero correlation and so they will only increase the computation time of the model and won't contribute in the accuracy of the model so we drop those columns from the dataframe and display new correlation table.

```
##--CODE--##
drop_col=['nbumps6','nbumps7','nbumps89']
seismic_df.drop(columns=drop_col,inplace=True)
Print(seismic_df.columns)
plt.figure(figsize=(10,10))
corr_mat=seismic_df.corr()
```

```
sns.heatmap(corr_mat,cmap='YlGnBu',annot=True,linewidths=0.2)
plt.show()

##--OUTPUT--##
Index(['seismic', 'seismoacoustic', 'shift', 'genergy', 'gpul', 'gdenergy',
'gdpuls', 'ghazard', 'nbumps', 'nbumps2', 'nbumps3', 'nbumps4', 'nbumps5',
'energy', 'maxenergy', 'class'], dtype='object'
```



Splitting the data into train and test set :

We divide the data into train and test set where train set contains 70% of the data and test set contains 30% of the data. This data is used for training the machine learning model and testing it's accuracy.

```
##--CODE--##
X = seismic_df.values[:,0:16]
y = seismic_df.values[:,-1]

print(X.shape)
print(y.shape)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
 random_state=101)


##--OUTPUT--##
(2584, 16)
(2584,)
```

###Machine Learning Models###

What is Cross Validation?

Cross-validation is a statistical method used to estimate the skill of machine learning models.

It is commonly used in applied machine learning to compare and select a model for a given predictive modeling problem because it is easy to understand, easy to implement, and results in skill estimates that generally have a lower bias than other methods.

The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. When a specific value for k is chosen, it may be used in place of k in the reference to the model, such as k=10 becoming 10-fold cross-validation.

Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model.

What is Confusion matrix and AUC-ROC curve ?

Confusion Matrix: In the field of machine learning and specifically the problem of statistical classification, a confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one.

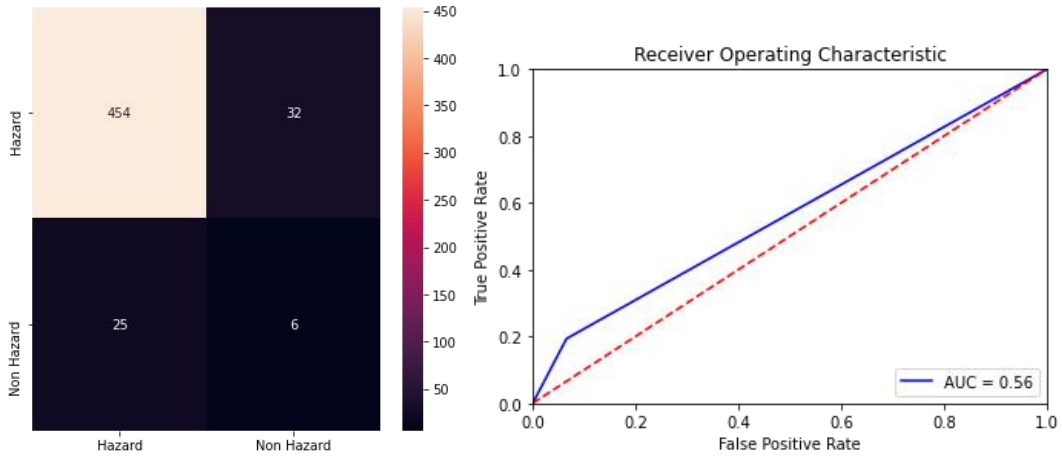
AUC ROC curve: In Machine Learning, performance measurement is an essential task. So when it comes to a classification problem, we can count on an AUC - ROC Curve. When we need to check or visualize the performance of the multi - class classification problem, we use AUC (Area Under The Curve) ROC (Receiver Operating Characteristics) curve. It is one of the most important evaluation metrics for checking any classification model's performance. It is also written as AUROC (Area Under the Receiver Operating Characteristics)

Gaussian Naive Bayes Algorithm :

Naive Bayes classifiers are a collection of classification algorithms based on **Bayes' Theorem**. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.
We apply this algorithm and get the **accuracy** to be **90.4%.**

```
##--CODE--##
from sklearn.naive_bayes import GaussianNB
classifier1 = GaussianNB()

trained_model = classifier1.fit(X_train, y_train)
y_pred = trained_model.predict(X_test)


#Finding accuracy using 10 fold cross-validation

accuracies = cross_val_score(estimator = classifier1, X = X_train, y =
y_train, cv = 10)
mn=accuracies.mean()
sd=accuracies.std()

print ("\n mean Accuracy:  ")
print (mn)
print ("\n Standard Deviation:")
print (sd)

print ("\n Confusion Matrix : ")
# Making the Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
plt.figure(figsize=(7,7))
sns.heatmap(cm,annot=True,fmt="0000.0f",xticklabels=["Hazard","Non Haza
rd"],yticklabels=["Hazard","Non Hazard"])
plt.show()

fpr, tpr, threshold = roc_curve(y_test, y_pred)
roc_auc=roc_auc_score(y_test, y_pred)
print ("\n AUC ROC score ",roc_auc)

import matplotlib.pyplot as plt
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()

##--OUTPUT--##
mean Accuracy: 0.9046831762112472
Standard Deviation: 0.017362429663327655
Confusion Matrix : [[454 32] [ 25 6]]
```



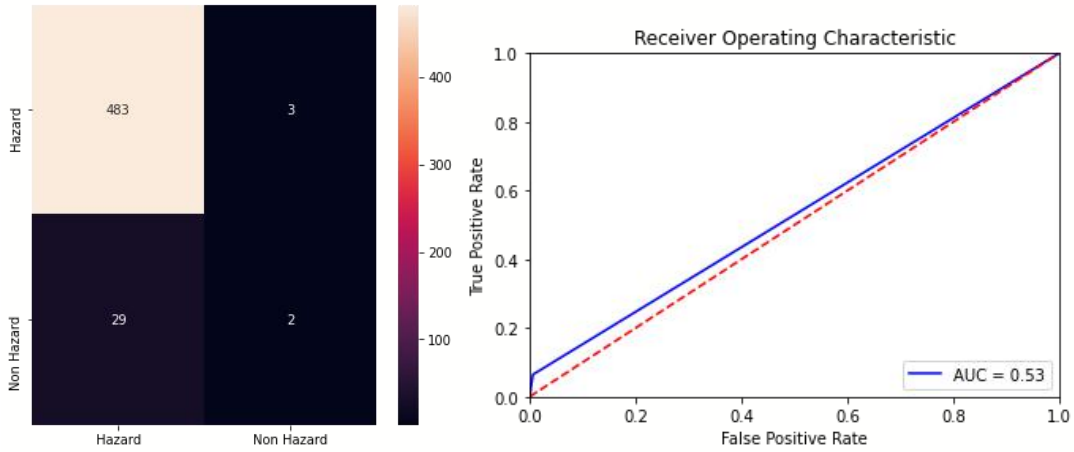
K Nearest Neighbours classifier :

K Nearest Neighbours is a non-parametric method proposed by Thomas Cover used for classification and regression. In both cases, the input consists of the *k* closest training examples in the feature space. In *k-NN classification*, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its *k* nearest neighbors (*k* is a positive integer, typically small). If *k* = 1, then the object is simply assigned to the class of that single nearest neighbor.

We apply this algorithm and get the **accuracy** to be **92.5%.**

```
##--CODE--##
#%%

from sklearn.neighbors import KNeighborsClassifier
classifier2 = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski
', p = 2)
trained_model = classifier2.fit(X_train, y_train)
y_pred = trained_model.predict(X_test)
accuracies = cross_val_score(estimator = classifier2, X = X_train, y =
y_train, cv = 10)
mn=accuracies.mean()
sd=accuracies.std()

print ("\n mean Accuracy: ")
print (mn)
print ("\n Standard Deviation:")
print (sd)

print ("\n Confusion Matrix : ")
#Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)

plt.figure(figsize=(7,7))
sns.heatmap(cm,annot=True,fmt="0000.0f",xticklabels=["Hazard","Non Haza
rd"],yticklabels=["Hazard","Non Hazard"])
plt.show()

fpr, tpr, threshold = roc_curve(y_test, y_pred)
roc_auc=roc_auc_score(y_test, y_pred)
print("AUC ROC")
```

```
print(roc_auc)

import matplotlib.pyplot as plt
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()

##--OUTPUT--##
mean Accuracy: 0.9250105529759394
Standard Deviation: 0.00623050116225987
Confusion Matrix :
[[483   3]
 [ 29   2]]
```



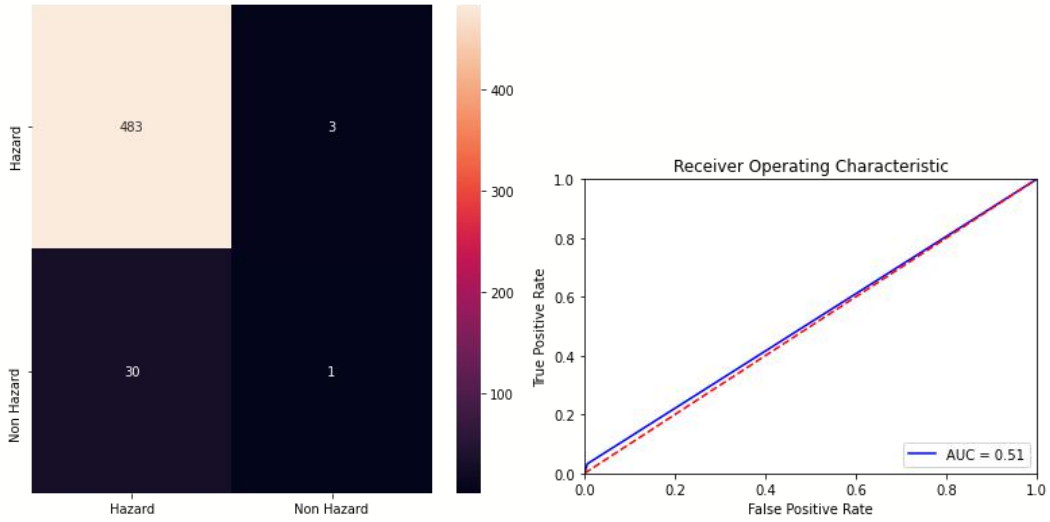
Logistic Regression :

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary regression).We apply this algorithm and get the **accuracy** to be **91.8%.**

```
##--CODE--##
from sklearn.linear_model import LogisticRegression   #class
classifier3  = LogisticRegression(random_state = 101,max_iter=2000)

trained_model = classifier3.fit(X_train, y_train)
y_pred = trained_model.predict(X_test)

accuracies = cross_val_score(estimator = classifier3, X = X_train, y =
y_train, cv = 10)
mn=accuracies.mean()
sd=accuracies.std()

print (" \n Logistic Regression :-  ")
print ("\n mean Accuracy: ")
print (mn)

print ("\n Standard Deviation:")
print (sd)

print ("\n Confusion Matrix : ")
#Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)

plt.figure(figsize=(7,7))
sns.heatmap(cm,annot=True,fmt="0000.0f",xticklabels=["Hazard","Non Haza
rd"],yticklabels=["Hazard","Non Hazard"])
plt.show()

fpr, tpr, threshold = roc_curve(y_test, y_pred)
roc_auc=roc_auc_score(y_test, y_pred)
print (roc_auc)

import matplotlib.pyplot as plt
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
```

```
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

```
##--OUTPUT--##
mean Accuracy: 0.9187139439988744
Standard Deviation:0.015466347834405257
Confusion Matrix :
[[483   3]
 [ 30   1]]
```

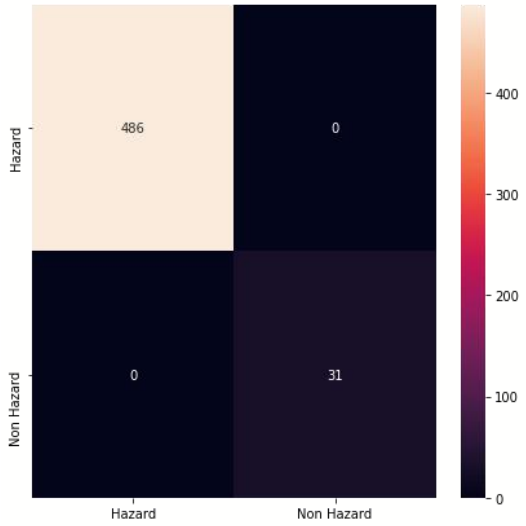


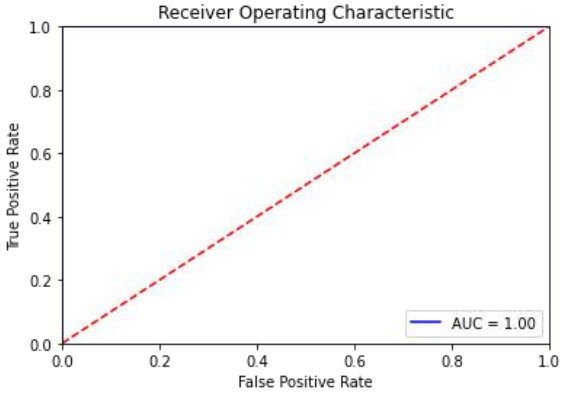
Decision Tree Classifier :

The decision tree classifier creates the classification model by building a decision tree. Each node in the tree specifies a test on an attribute, each branch descending from that node corresponds to one of the possible values for that attribute.We apply this algorithm and get the **accuracy** to be **100%.**

```
##--CODE--##
%%
from sklearn.tree import DecisionTreeClassifier
classifier4 = DecisionTreeClassifier(criterion = 'entropy', random_stat
e = 0)
trained_model = classifier4.fit(X_train, y_train)
y_pred = trained_model.predict(X_test)

accuracies = cross_val_score(estimator = classifier4, X = X_train, y =
y_train, cv = 10)
mn=accuracies.mean()
sd=accuracies.std()

print ("\n Decision Tree :-  ")
print ("\n mean Accuracy: ")
print (mn)

print ("\n Standard Deviation:")
print (sd)

print ("\n Confusion Matrix : ")
# Making the Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)

plt.figure(figsize=(7,7))
sns.heatmap(cm,annot=True,fmt="0000.0f",xticklabels=["Hazard","Non Haza
rd"],yticklabels=["Hazard","Non Hazard"])
plt.show()

fpr, tpr, threshold = roc_curve(y_test, y_pred)
roc_auc=roc_auc_score(y_test, y_pred)
print (roc_auc)

import matplotlib.pyplot as plt
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
```

```
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()

##--OUTPUT--##
mean Accuracy: 1.0
Standard Deviation:0.0
Confusion Matrix :
[[486   0]
 [  0  31]]
```





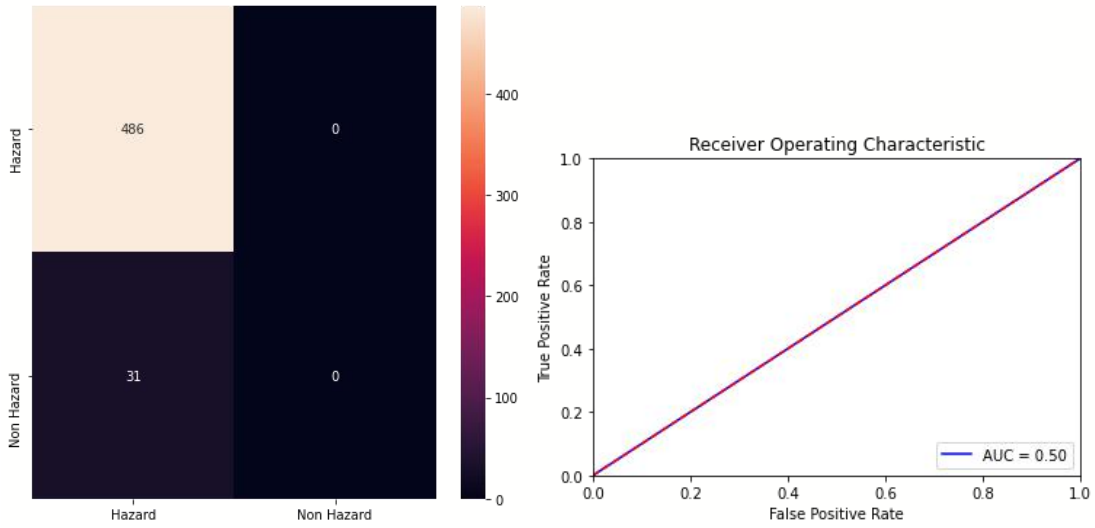
Support Vector Classifier :

support-vector machines (SVMs, also support-vector networks[1]) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). We apply this algorithm and get the **accuracy** to be **93.3%.**

```
##--CODE--##
from sklearn.svm import SVC
classifier5 = SVC()
trained_model = classifier5.fit(X_train, y_train)
y_pred = trained_model.predict(X_test)

accuracies = cross_val_score(estimator = classifier5, X = X_train, y =
y_train, cv = 10)
mn=accuracies.mean()
sd=accuracies.std()

print (" \n Support Vector machines :-  ")
print ("\n mean Accuracy: ")
print (mn)

print ("\n Standard Deviation:")
print (sd)

print ("\n Confusion Matrix : ")
# Making the Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)

plt.figure(figsize=(7,7))
sns.heatmap(cm,annot=True,fmt="0000.0f",xticklabels=["Hazard","Non Haza
rd"],yticklabels=["Hazard","Non Hazard"])
plt.show()

fpr, tpr, threshold = roc_curve(y_test, y_pred)
roc_auc=roc_auc_score(y_test, y_pred)
print("\nAUC ROC ",roc_auc)
```

```
import matplotlib.pyplot as plt
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

```
##--OUTPUT--##
mean Accuracy: 0.9327540922095586
Standard Deviation:0.0013857670281515956
Confusion Matrix :
[[486   0]
 [ 31   0]]
```

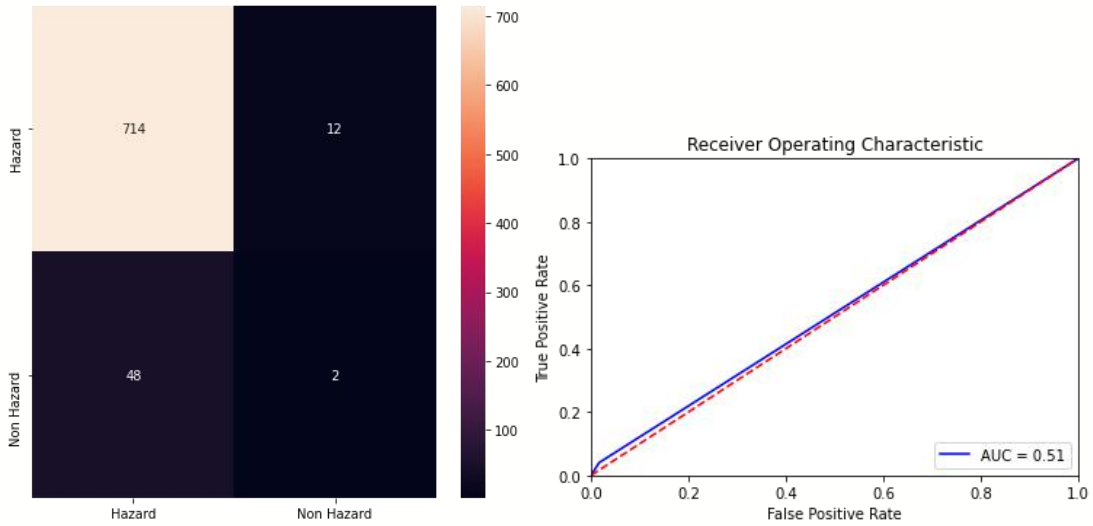


Random Forest Classifier :

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. We apply this algorithm and get the **accuracy** to be **93.2%.**

```
##--CODE--##
#%%
from sklearn.ensemble import RandomForestClassifier

seed = 7
num_trees = 150
max_features = 10

classifier6 = RandomForestClassifier(n_estimators=num_trees, max_featur
es=max_features)
trained_model = classifier6.fit(X_train, y_train)
y_pred = trained_model.predict(X_test)



accuracies = cross_val_score(estimator = classifier6, X = X_train, y =
y_train, cv = 10)
mn=accuracies.mean()
sd=accuracies.std()

print ("\n Random Forest : Randomly selected 10 features and 100 forest
 and then mean accuracy :-  ")
print ("\n mean Accuracy: ")
print (mn)

print ("\n Standard Deviation:")
print (sd)

cm = confusion_matrix(y_test, y_pred)
print(cm)

plt.figure(figsize=(7,7))
sns.heatmap(cm,annot=True,fmt="0000.0f",xticklabels=["Hazard","Non Haza
rd"],yticklabels=["Hazard","Non Hazard"])
plt.show()

fpr, tpr, threshold = roc_curve(y_test, y_pred)
```

```
roc_auc=roc_auc_score(y_test, y_pred)
print (roc_auc)

import matplotlib.pyplot as plt
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()


##--OUTPUT--##
mean Accuracy: 0.9319674647022712
Standard Deviation:0.006576405939703404
Confusion Matrix:
[[714  12]
 [ 48   2]]
```



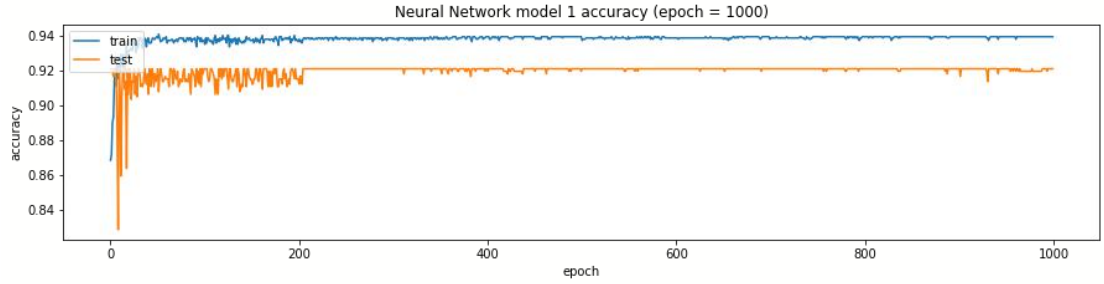
###Deep Learning Models###

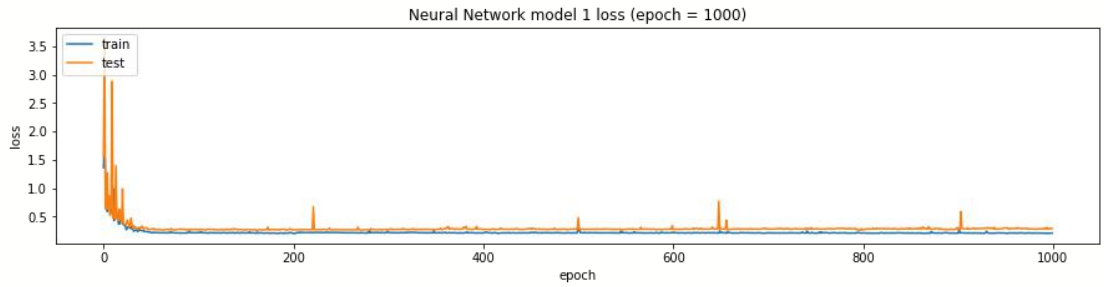
Artificial Neural networks :

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. We apply this algorithm and get the **accuracy** to be **94%.**

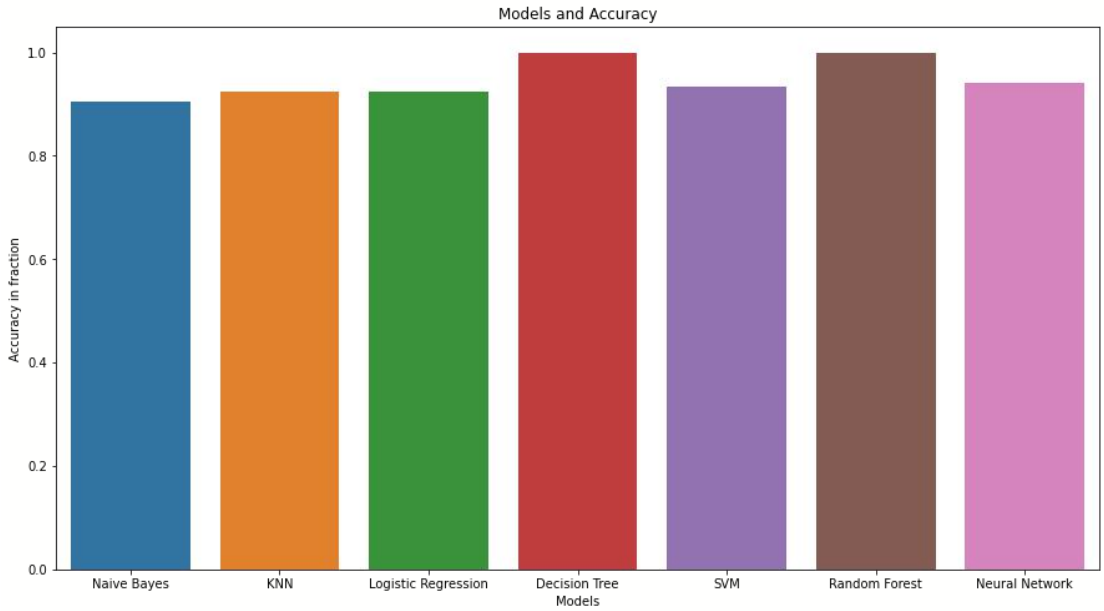
```
##--CODE--##
#%%
model = Sequential()
model.add(Dense(17, input_dim=16, kernel_initializer='normal', activati
on='relu'))
model.add(Dense(9, kernel_initializer='normal', activation='relu'))
model.add(Dense(1, kernel_initializer='normal', activation='sigmoid'))
# Compile model
sgd =optimizers.SGD(lr=0.01, momentum=0.8, decay=0.0, nesterov=False)
model.compile(loss='binary_crossentropy',optimizer='adam', metrics=['ac
curacy'])

#We manually provide the train and test partition
history = model.fit(X_train, y_train, validation_split=0.33, epochs=100
0, batch_size=16, verbose=2)
Y_test_set = seismic_df.values[:,17]
#Y_test_set.reshape((-1,17))
ynew = model.predict_classes(X_test)

# summarize history for accuracy
plt.figure(figsize=(15,7))

plt.subplot(2,1,1)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Neural Network model 1 accuracy (epoch = 1000)')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

plt.figure(figsize=(15,7))

plt.subplot(2,1,2)
```

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Neural Network model 1 loss (epoch = 1000)')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
print("Accuracy score: ",accuracy_score(y_test,ynew))

##--OUTPUT--##
##Epoch from 1 to 1000##
---
Epoch 995/1000
 - 0s - loss: 0.2043 - accuracy: 0.9393 - val_loss: 0.3112 - val_accuracy: 0.9209
Epoch 996/1000
 - 0s - loss: 0.2113 - accuracy: 0.9393 - val_loss: 0.2823 - val_accuracy: 0.9209
Epoch 997/1000
 - 0s - loss: 0.2074 - accuracy: 0.9393 - val_loss: 0.2866 - val_accuracy: 0.9209
Epoch 998/1000
 - 0s - loss: 0.2073 - accuracy: 0.9393 - val_loss: 0.2908 - val_accuracy: 0.9209
Epoch 999/1000
 - 0s - loss: 0.2093 - accuracy: 0.9393 - val_loss: 0.2961 - val_accuracy: 0.9209
Epoch 1000/1000
 - 0s - loss: 0.2096 - accuracy: 0.9393 - val_loss: 0.2909 - val_accuracy: 0.9209

Accuracy score:  0.9400386847195358
```





Comparing the models :



From the above table we can see that we get best accuracy for Decision Tree algorithm and Random Forest algorithm.

Result and Conclusion

We can see that we get good results for the above dataset and this model helps us in the prediction of seismic hazards for the next shift. The accuracy of the model can be further improved if we collect more data, since that will help us better train the model.
So, we can conclude that Machine Learning helps us predict results for serious hazardous problems like earthquake which is not possible to be solved through equations and statistical methods.

What Next?
Since this model is in python code and if anyone wants to use it then he will be required to have the knowledge of python. So, in order to make this project public I want to deploy it in a website where anyone can just feed the data (measurement from sensors) of current shift and get a reliable result for the occurence of earthquake in the next shift.

Advantages and Improvements

Major advantage of using this model is that it will help us predict any incoming occurence of Earthquake and will help to save lives of workers and protect useful machineries. This model is very cost efficient since deployment on a website does not cost too much and so it can be made available on public domain.

We can introduce more amount of data and features to improve the accuracy thereby making correct prediction for any instance.

References

- Link to the dataset :
  https://archive.ics.uci.edu/ml/datasets/seismic-bumps#
- Pandas Documentation :
  https://pandas.pydata.org/
- Sklearn Documentation :
  https://scikit-learn.org/stable/
- Keras Documentation :
  https://keras.io/api/
- Matplotlib Documentation :
  https://matplotlib.org/contents.html