**CLASSIFICATION OF EARTHQUAKE EARLY WARNING**

A project dissertation submitted to Bharathidasan University

in partial fulfillment of the requirements

for the award of the Degree of

**MASTER OF SCIENCE IN DATA SCIENCE**

***Submitted by***

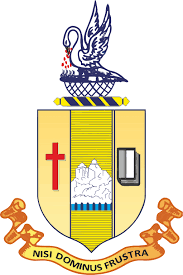
**HARIHARAN M**

**195229107**

***Guided by***

**Dr. K.RAJKUMAR,M.Sc,M.Phil,P.hD**

**Associate Professor**



**DEPARTMENT OF DATA SCIENCE**

**BISHOP HEBER COLLEGE (AUTONOMOUS)**

(Nationally Reaccredited at the ‘A’ Grade by NAAC with the CGPA of 3.58 out of 4)

(Recognized by UGC as “College with Potential for Excellence”)

(Affiliated to Bharathidasan University)

**TIRUCHIRAPPALLI 620017**

**APRIL 2021**

## DECLARATION

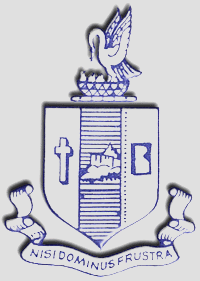
I hereby declare that the project work presented is originally done by me under the guidance of **Dr. K.RAJKUMAR, M.Sc,M.Phil,P.hD, Department of Data Science, Bishop Heber College (Autonomous), Tiruchirappalli 620017**, and has not been included in any other thesis/project submitted for any other degree.

**Name of the Candidate** : HARIHARAN M

**Register Number** : 195229107

**Batch** : 2019-2021

Signature of the Candidate

**DEPARTMENT OF DATA SCIENCE**

**BISHOP HEBER COLLEGE (AUTONOMOUS)**

(Nationally Reaccredited at the ‘A’ Grade by NAAC with the CGPA of 3.58 out of 4)

(Recognized by UGC as “College with Potential for Excellence”)

(Affiliated to Bharathidasan University)

TIRUCHIRAPPALLI 620017



Date: 31-01-2021

Course Title: Project Course Code: P19DS4PJ

BONAFIDE CERTIFICATE

This is to certify that the project work titled **“CLASSIFICATION OF EARTHQUAKE EARLY WARNING ”** is a bonafide record of the project work done by **Hariharan, 195229107,** in partial fulfillment of the requirements for the award of the degree of **MASTER OF SCIENCE IN DATA SCIENCE** during the period **2019 - 2021.**

The Viva-Voce examination for the candidate Hariharan, 195229107, was held on 31-01-2021.

Signature of the HOD Signature of the Guide

Examiners:

1.

2.

**ACKNOWLEDGEMENTS**

I sincerely thank **Dr. D. PAUL DHAYABARAN, M.Sc., M .Phil., PGDCSA., Ph.D**., **Principal,** Bishop Heber College, Trichy, for providing me with sufficient facilities which contributed to the successful completion of the project.

I wish to place on record my gratitude to **Dr. K. RAJKUMAR, M.Sc., M.Phil., Ph.D., Associate Professor and Head, Department of Data Science**  (S.F), for his motivation and guidance through the course of my project work.

I am blessed to have Dr. **K. RAJKUMAR, M.Sc., M.Phil., Ph.D., Associate Professor and Head, Department of Data Science** Bishop Heber College, and my research adviser. I wholeheartedly thank him because this is possible only because of patience, support and ready guidance. I wish to thank for being kind and considerate in all respects towards me right from the beginning. His untiring help during my difficult moments, his motivation and kindness helped me a lot for completing my project work successfully.

I am grateful to my parents for their blessing, love and sacrifice in which my life was built. I also thank my friend especially for their help and support in shaping the project.

**HARIHARAN M**

**ABSTRACT**

Our system aims to improve the accuracy of Earthquake Early Warning (EEW) systems by means of machine learning. EEW systems are designed to detect the different characteristics of (medium and large) earthquakes before their damaging effects. Traditional EEW methods based on seismometers fail to accurately identify large earthquakes due to their sensitivity. The recently introduced high-precision GPS stations are ineffective to identify medium earthquakes. In addition, GPS stations and seismometers may be deployed in large numbers across different locations and may produce an enormous amount of data, this will affect the response time of the traditional EEW systems. 

The existing system builds on a geographically distributed infrastructure(sensor stations), but it tooks time to detect and characterize the earthquake. Our experiments show that EEW is more accurate than the traditional seismometer-only approach. EEW system can be seen as a classification problem in machine learning. Sensor data is the input and earthquake character will be the result. The characters such as large , small or medium earthquakes. Each sensor has a different drawback with them by finding the earthquakes, this system combines the sensor data and uses the machine learning approach to detect the earthquake little faster than the sensor.

Early earthquake warning system combines the sensor data from the two different types of sensors(seismometers and GPS). Our system tries to improve the accuracy of the EEW to detect the earthquake early. Traditional EEW methods based on seismometers fail to accurately identify large earthquakes due to their sensitivity to the ground motion velocity.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Chapter** | **Title** | **Page No** |
| **1**  1.1  1.2  1.3 | **Introduction**  Motivation  Existing Systems and Solutions  Product Needs and Proposed System | 07 |
| 1.4 | Product Development Timeline |  |
| **2**  2.1  2.2  2.3 | **Related Work and Solutions Review**  Related Work 1  Related Work 2  Related Work 3 | 10 |
| **3**  3.1  3.2 | **Data Collection**  Description of the Data  Source and Methods of Collecting Data | 13 |
| **4**  4.1  4.2  4.3 | **Preprocessing and Feature Selection**  Overview of Preprocessing Methods  Overview of Feature Selection Methods  Preprocessing and Feature Selection Steps | 14 |
| **5**  5.1  5.2  5.3 | **Model Development**  Model Architecture  Algorithms Applied  Training Overview | 16 |
| **6**  6.1  6.2 | **Experimental Design and Evaluation**  Experimental Design  Experimental Evaluation | 19 |
| 6.3 | Customer Evaluation and Feedback |  |
| **7**  7.1  7.2 | **Model Optimization**  Overview of Model Tuning and Best Parameter Selection  Model Tuning Process and Experiments | 23 |
| **8**  8.1  8.2  8.3 | **Product Delivery and Deployment**  User Manuals  Delivery Schedule  Deployment Process | 26 |
| **9**  9.1  9.2 | **Conclusion**  Summary  Limitation and Future Work | 31 |
|  | **References** | 32 |
|  | Appendix-A: Data Set | 33 |
|  | Appendix-B: Source Code | 34 |
|  | Appendix-C: Output Screenshots | 38 |

**Chapter 1**

**INTRODUCTION**

**1.1 Motivation**

Earthquakes and floods are the major disasters that cause many life loss, injuries and damage to the environment across the world. Mainly earthquakes tend to cause the tsunami, landslides and fires and many disasters. There were many earthquake early detector stations placed across the world, that work all days, months and years to detect the earthquake and produce the data for every second from the sensors. But these sensors took some time to detect the earthquake from the sensors.

Earthquake early warning can be assimilated as a classification problem, where the input is sensor data and the output is a class (normal activity/medium earthquake/large earthquake). The machine learning approaches designed to combine large volumes of data from multiple data sources(sensors).

Earthquakes and floods cause more environmental damage than any other disaster. Earthquakes can be detected by the Seismometers and GPS sensors. Seismometer detects the medium earthquakes and GPS will detect the large earthquakes. Traditionally these are the methods used to detect the earthquakes across the world by placing the sensors in the ground level.

**1.2 Existing Systems and Solutions**

Seismometers are able to detect the medium earthquakes which are under the scale of 5< magnitude <6 , Richter scale, but it has the difficulty to detect the large earthquakes which are >6 magnitude. GPS sensors are able to detect the large earthquakes but not medium. Thus our classification of machine learning approach combines these sensor data and gives better accuracy than the traditional systems.

**1.3 Product Needs and Proposed System**

**Product Needs:**

Earthquake early warning can stop many disasters and loss of human life. Traditional methods take more time to detect before an earthquake. To mitigate the disastrous effects, a number of Earthquake Early Warning (EEW) systems have been built around the world (Allen and Melgar 2019). These critical systems, operating 24/7, are expected to automatically detect and characterize the earthquake and to deliver alerts before the ground motion actually reaches sensitive areas so that protective measures could be taken.

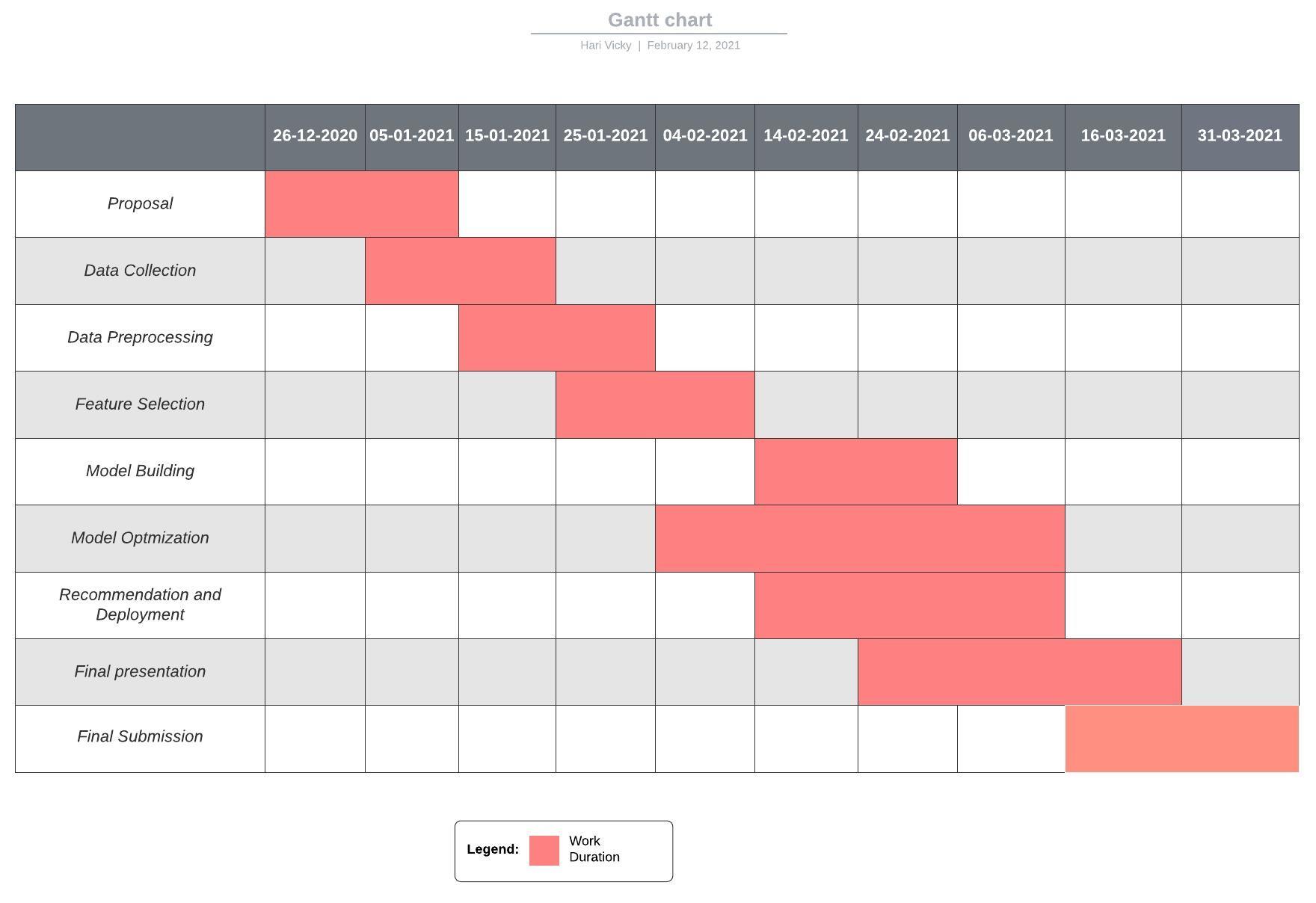
**Proposed system:**

The proposed system combines the sensor data and provides accuracy and decreases the time for detection. This proposed system gives better accuracy than the existing system. This proposed system uses a supervised machine learning classification approach such as support vector classifier(SVC), Random forest classifier, KNearest Neighbor(KNN) to classify the type of the earthquake.

**Requirements of Tools and Libraries in Proposed system**

* Notebook Editor (Jupyter or Spyder)
* Python Programming
* Scikit Learn-Machine Learning-Supervised Learning
* Visualization in Tableau
* Flask-Web application
* Heroku-Deployment

**1.4 Product Development Timeline**

**Figure1.1(Product development Timeline)**

**Chapter 2**

**RELATED WORK AND SOLUTIONS REVIEW**

**2.1 Related Work1**

**The Model:**

This related work Earthquake Early Warning and Tsunami Warning, provides data and details about the attributes.

**Features and Missing Features:**

In this paper I took the geographical dataset collected from the real time sensors. The dataset contains attributes related to earthquake GPS stations location, scale of the earthquake and so on. The dataset attributes will be present in the following chapters. This related work only gives the accuracy of the model but not with multiple sensor data.

**2.2 Related Work2**

**The Model:**

This related work was based on implementation of different machine learning models using sklearn library. In this work I took machine learning techniques used to handle the sensor data using the machine learning approach.

**Features and Missing Features:**

In this paper I took the stratified kfold method to train and test the data into equal ratio. The dataset contains many classes, so this stratified kfold method will help the model understand the data. But this work gives a lower accuracy rate than the proposed system.

**2.3 Related Work3**

**The Model:**

This work has different time duration for past 9years data on earthquakes. This sensor data gives the detailed information of earthquake location, scale of the earthquakes time and date of the earthquakes.

**Features and Missing Features:**

In this related work I took different attributes that help to learn more about the data. These attributes such as station latitude, station longitude, station name will add more information to the data to identify the earthquake.

**Proposed system Different from related work:**

As the problem is the classification problem, the proposed system would use supervised classification algorithms, such as Support Vector classifier, Random Forest classifier, and KNearest Neighbor model. I would apply them separately and choose the one which performs the best based on evaluation metrics.

Existing system has model accuracy around 60%. Our proposed system tries to improve the accuracy better than the existing system, so that model can work well on validation. Our proposed system uses the stratified k-fold validation technique from this existing system and also cross validation.

This model will take the input feature as described above in the dataset and inputs section in addition to this some technical indicators like event\_depth, event\_magnitude, event\_latitude, event\_longitude.

**Steps to be followed during Project:**

* Obtain the data
* Scrubbing the data
* Exploratory data analysis
* Feature selection
* Split train test data
* Various type of Model building
* Predict the output
* Evaluation Metrics
* Select best Model
* Create Flask server
* Develop web application through Flask
* Heroku deployment

**FLOW DIAGRAM**

**WORKFLOW-1:**

****

**WORKFLOW-2:**

**WORKFLOW-3:**

****

**WORKFLOW-4:**

****

**WORKFLOW-5:**

****

**Chapter 3**

**DATA COLLECTION**

**3.1 Description of the Data**

**About Data:**

This dataset is a historical data that contains GPS stations and seismometers multivariate time series data associated with three types of events (normal activity / medium earthquakes / large earthquakes). This data contains the following attributes.

Data for this study is collected for the 2001-2018 period from different stations. The data has 45320 rows and 15 columns. These attributes include the latitude, longitude of the station, magnitude, depth of the earthquake, recorded station and type of the earthquake.

**Size of Data:**

In Proposed system 45320 records in the dataset and 15 columns including direction, depth, station, type of the earthquake which is our target variable.

**Features of Data:**

* event\_id: unique ID of an event. Dataset is composed of 269 events
* event\_time: timestamp of the event occurrence
* event\_magnitude: magnitude of the earthquake
* event\_latitude: latitude of the event recorded (degrees)
* event\_longitude: longitude of the event recorded (degrees)
* event\_depth: distance below Earth's surface where earthquake happened (km)
* station: sensor name (GPS station or seismometer)
* station\_latitude: sensor (GPS station or seismometer) latitude (degrees)
* station\_longitude: sensor (GPS station or seismometer) longitude (degrees)
* label : describes which type of earthquake.

**Target Variable:**

* label : predict the type of the earthquake

**3.2 Source and Methods of Collecting Data**

This historical dataset was produced by the IRIS(Incorporated Research Institutions for Seismology) and as mentioned early in the (2.1Related Work1). GPS stations and seismometers data are obtained from the archive file. The URL of the data source is (<https://figshare.com/articles/dataset/Earthquake_Early_Warning_Dataset/9758555> ,<http://ds.iris.edu/wilber3/find_event> ).

This dataset includes 29 large earthquakes. The number of medium earthquakes is calculated by the ratio of medium over large earthquakes during the past 10 years. The complete dataset is attached in the Appendix-A.

**Chapter 4**

**PREPROCESSING AND FEATURE SELECTION**

**4.1 Overview of Preprocessing Methods**

**Preprocessing:**

In data collection not needed every data is clean so we remove some noise through data preprocessing.

**Data Preprocess method are used in this Proposed system:**

* **Combine all the data**
* **Handling Missing values**
* **Scale the numerical columns**
* **Handling categorical values**
* **Drop unwanted columns**

**Why are the above methods needed?**

**1.Combain all the data**

In the Proposed system collect some kind of commodity data like magnitude, depth of the earthquake these are all collected from various sources so finally combine all the data in

one new data frame with the help of merge.

**2.Handling Missing values**

If suppose the dataset has some null values that are the main disadvantage of predicting the rate. So it must handle filling null values or drop null values.

**3.Scale the numerical columns**

Seismometers data are available as digital signals, which is speciﬁc for each sensor. Therefore, each sensor column must be scaled before the train and test.

**4.Handling categorical values**

In machine learning algorithms don’t support categorical data so the proposed

system converts the categorical data to numerical data with the help of the label encoder

library and also it can be changed manually.

**5**.**Drop unwanted columns**

The dataset contains some unwanted columns which does not give any information, for that columns we can drop the columns for the better model performance.

**4.2 Overview of Feature Selection Methods**

Performed the StandardScaler for the GPS data to normalize and keep the outliers, which are large earthquake readings. Encoding the categorical data columns using rename method and label encoder library.

**4.3 Preprocessing and Feature Selection Steps**

**Handling Null values:**

This Proposed system has some null values but here use the filling method because if suppose drop null values that are not an efficient way of future prediction so that here proposed system using fill method with the help of pandas.

**Handling Categorical data:**

In this Proposed system the commodity and date column had an object type but the proposed system convert date column with help of date time function another one important feature is commodity that has three columns.

from sklearn.preprocessing import LabelEncoder

Le=LabelEncoder()

data\_gps[‘station’]=Le.fit\_transform(data\_gps[‘station’])

data\_gps

**Scaling the columns:**

Here the numerical values will be scaled using the standard scalar() library. This helps to change the sensor data into the normal distributed data.

from sklearn.preprocessing import standardscaler

sc=StandardScaler()

scale=['event\_magnitude','event\_latitude','event\_longitude','event\_depth','mts\_id','station','station\_latitude','station\_longitude','timestamp','dimension\_E','dimension\_N','dimension\_Z']

data\_gps[scale]=sc.fit\_transform(data\_gps[scale])

**Select input and output features:**

x=data\_gps.drop(columns=['label','event\_time','timestamp'],axis=1)

y=data\_gps['label']

**Feature Selection:**

Compute the importance of the feature by using the extratree regression() library. We can easily find which columns are more important and which do not give any information by using this method.

model = ExtraTreesClassifier()

model.fit(x,y)

**Chapter 5**

**MODEL DEVELOPMENT**

**5.1 Model Architecture**

The proposed system combines the sensor data and provides accuracy and decreases the time for detection. This proposed system gives better accuracy than the existing system.

We would use the train test split function in scikit-learn to split the data into the training set and testing set, it could split the whole dataset into several packs and in each pack, the indices of the testing set would be higher than the training set.

**Model Architecture**

****

**5.2 Algorithms Applied**

This Proposed system has used different machine learning algorithms to see if they can accurately classify the type of earthquake.

I have used 5 different regression machine learning algorithms that are shown in below.

* Logistic regression
* Support vector classifier
* KNearest neighbour
* Decision tree classifier
* Random Forest classifier

Finally, we ensemble top three best performing algorithms and compare their

performance with other algorithms.

**1.Logistic Regression**:

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is dichotomous, which means there would be only two possible classes

Logistic Regression= 1\1+e^-z

Just as ordinary least square regression is the method used to estimate coefficients for the best fit line in linear regression, But here in this proposed system used classification algorithm so that, logistic regression uses maximum likelihood estimation (MLE) to obtain the model coefficients that relate predictors to the target

**2. Support Vector Classifier:**

The linear SVM classifier works by drawing a straight line between two classes. All the data points that fall on one side of the line will be labeled as one class and all the points that fall on the other side will be labeled as the second.

The equation of a line is n . [x y] + b = 0, where n is the unit normal vector, b is the distance from the origin, and [x y] is the vector from the origin to the point. All points further from the origin than the line, satisfy the equation n . [x y] + b > 0; all those closer to the origin, satisfy the equation n

**3. Decision Tree Classifier:**

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. There are two types of Decision Tree available in Machine learning one is classification another one is regression but in this proposed system outcome is 0 or 1 that means chance of earthquake or not so obviously machine goes to use classification algorithm.

**4. Random Forest Classifier:**

Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction

It provides higher accuracy through cross validation. Random forest classifier will handle the missing values and maintain the accuracy of a large proportion of data.

**5. K-Nearest Neighbour Classifier:**

KNN algorithms use data and classify new data points based on similarity measures (e.g. distance function). Classification is done by a majority vote to its neighbors.

KNN is a non-parametric, lazy learning algorithm. Its purpose is to use a database in which the data points are separated into several classes to predict the classification of a new sample point. Just for reference, this is “where” KNN is positioned in the algorithm list of scikit learn.

**5.3 Training Overview**

**Training methods:**

The proposed system split the data into training and testing through train\_test\_split function. Here the proposed system used 0.8 data as a training process and the remaining 0.2 data is testing process.

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

**Training Process:**

The process of training an ML model involves providing an ML algorithm that is, the learning algorithm with training data to learn from. The term ML model refers to the model artifact that is created by the training process.

The proposed system is under supervised machine learning algorithms so dependent data can learn from independent data. So that machine can learn from that historical data. In this proposed system use three types of train test split in the first experiment use 80% of training and 20% of testing through sklearn train\_test\_split

**X\_train, X\_test, Y\_train, Y\_test= train\_test\_split(X,y,test\_size=0.2)**

In this proposed system use three types of train test split in the second experiment use 70% of training and 30% of testing through sklearn train\_test\_split.

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

**Chapter 6**

**EXPERIMENTAL DESIGN AND EVALUATION**

**6.1 Experimental Design**

In the existing system type of the earthquake is classified by the use of historical data. And finally, the result can compare by various algorithms to select which one is the highest accuracy that model will be applicable in user usage.

**Design of Experiment:**

This experiment design by the web app and deployed through Heroku so that users can give the attributes of earthquake such as magnitude, depth, longitude, latitude of the earthquake through the web page then click the classify button user can classify the type of the earthquake and this is the experiment design of this product.

**Experiment-1: Train test Feature selection and Select X and Y :**

After data scrubbing obviously, data is ready to involve the machine learning model but we must select the best feature before model building.

This proposed system can select the best features and select x and y then go to build the model.

In this proposed system use three types of train test split in first experiment use 80% of

training and 20% of testing through sklearn train\_test\_split.

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)

**Experiment-2: Model Build and training**

After select x and y that data is ready to build the model here choose the algorithm and fit

the model through fit() method after the fitting system will predict the data through .predict()

method. In this proposed system train the model through various types of machine learning

algorithm when using different types of algorithms that give different types of prediction.

**Different types of algorithms:**

* Logistic regression
* Support vector classifier
* KNearest neighbour

**Experiment-3: Given input and get output**

The training and the testing process almost over but when new data comes to the model

it can predict the data correctly now , hence product testing is almost good. Here test data

given by the various types of input.

test=[[0, 6.0 , 35.815, -120.37, 7.9, 0, 704, 35.9394, -120.4337, 33100.0, -5800.0, 44400.0]]

sc.fit\_transform(test)

classifier.predict(test)

**6.2 Experimental Results**

**Algorithm result variance in experiment:**

In this proposed system used various types algorithm that algorithm give different types

of algorithms. In below shown various outputs through plotly visualization.

**Experiment-1: Logistic Regression**

In this proposed system train the model used logistic regression. In this logistic regression train the model through fit()method when developer uses the logistic regression without any tuning, algorithm result is not efficient but algorithm train with default tuning like logistic regression work with n\_jobs = -1.

**Experiment-2: Support Vector Classifier**

The linear SVM classifier works by drawing a straight line between two classes. Here kernel = 'rbf', random\_state = 0 parameters helps the model to learn about the data.

**Experiment-3: KNearest neighbour**

KNN algorithms use data and classify new data points based on similarity measures. Classification is done by a majority vote to its neighbors. Here n\_neighbors=7 was the best neighbor value for this data. It takes much computation time for the larger datasets.

**Validation-1: Predict the normal activity**

Through this data we can classify three types of stages(normal,medium,large). Normal shows that there is no peak change in the attributes and it is a normal activity of earth. Large and medium implies that there is an abnormal change in the activity and there is abnormal activity on the earth.

test=[[0,6.0,35.815,-120.37,7.9,0,704,35.9394,-120.4337,33100.0,-5800.0,44400.0]]

model.predict(test)

The above data is the example for normal activity, the result will be in the form of numeric values, which say the type of activity.

**Validation-2: Predict the medium activity**

The following data depends on the medium activity, when we try to predict the data through the model then it classifies as the medium activity.

test=[[0,1.0,66.998,-300.39,9.9,6,900,56.9394,-100.4558,33100.0,-8870.6,98900.0]]

model.predict(test)

**Validation-3**: **Predict the large activity**

The following data depends on the large activity, when we try to predict the data through the model then it classifies as the large activity which means its a large earthquake .

test=[[0,1.0,66.998,-300.39,9.9,6,900,56.9394,-100.4558,33100.0,-8870.6,98900.0]]

model.predict(test)

**Evaluation Metrics**

Evaluation metrics means to analyze the error of the product in this proposed system can analyze the evaluation metrics through Metrics library this metrics library provides mean square error, root mean square error, accuracy in this product full and fully regression problem so that obviously we can go to calculate MSE and RMSE.

**Accuracy**

Accuracy is one metric for evaluating classification models. Informally, accuracy is the fraction of predictions our model got right.

**accuracy = no.of correct predictions / total number of predictions**

**Confusion Matrix**

Confusion Matrix is a summary of predicted results in a specific table layout that allows visualization of the performance measure of the machine learning model for a binary classification problem (2 classes) or multi-class classification problem (more than 2 classes). This will be calculated by “True positive, False positive, False negative, False positive”.

**Precision**

Precision is the fraction of the correctly classified instances from the total classified instances.

**precision= true positive / true positive + false positive**

**Recall**

Recall is the fraction of the correctly classified instances from the total classified instances.

**precision= true positive / true positive + false negative**

**F1 Score**

This does the work of both precision and recall, F1 score is the harmonic mean of precision and recall.

**F1\_score= 2\*precision\*recall / precision + recall**

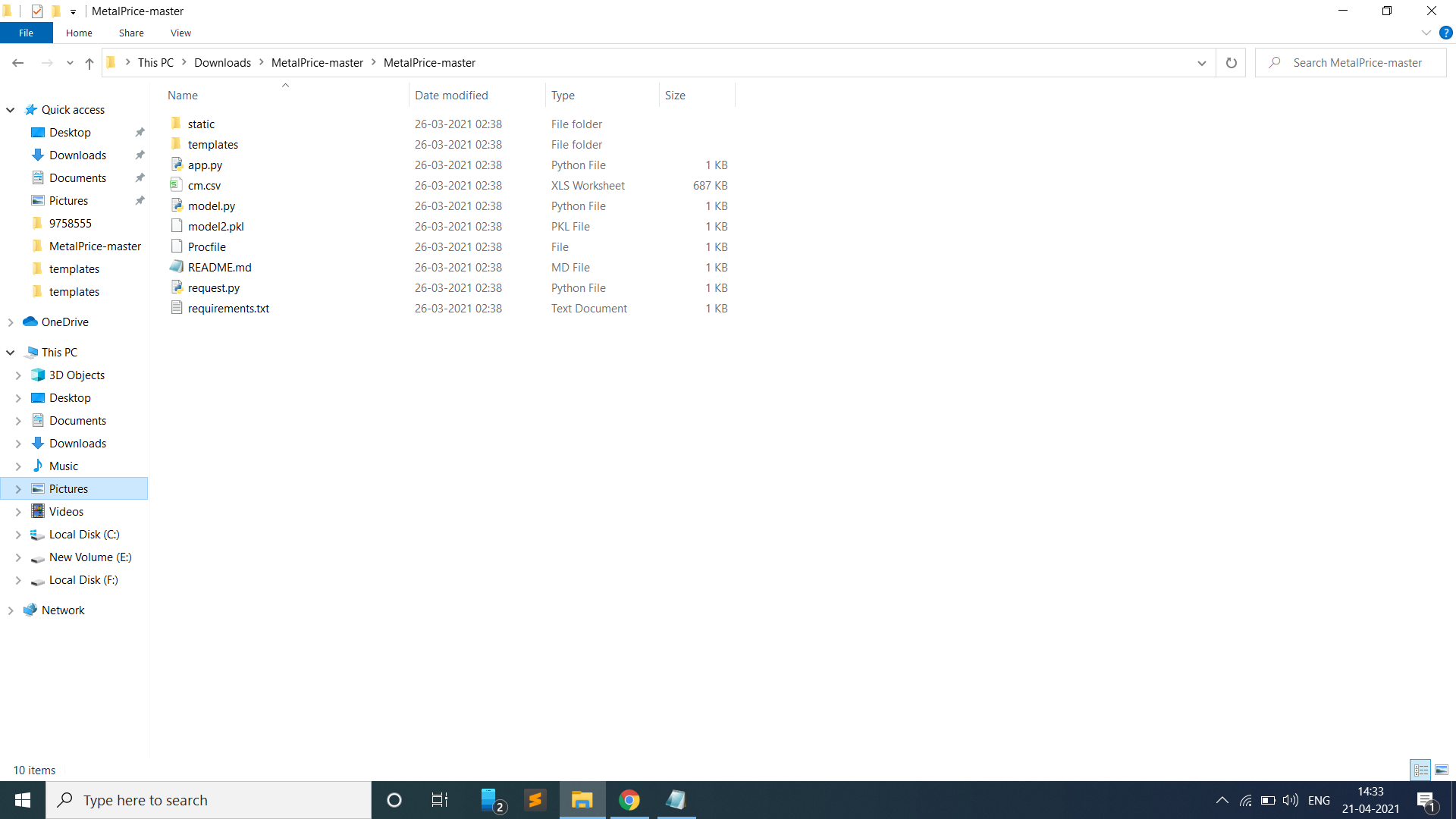
**Evaluation metrics report**

|  |  |
| --- | --- |
| **Model Name** | **Accuracy** |
| **Logistic Regression** | 44 |
| **Support vector classifier** | 92 |
| **K Nearest neighbor** | **73** |

**Develop pickle file**

In after the prediction we can convert this python file to pickle file because when we

develop web applications now pickle file can handle all the data through app.py file.

****

**6.3 Customer Evaluation and Feedback**

**Evaluation**

When the product is delivered to the customer, a one-time developer gives the demo now the customer evaluates the products if the customer's requirements satisfy or not. During the evaluation customer evaluates step-by-step requirements like designing styles in pages, then checks given demo data that data will predict or not if the user gives measurements of the earthquake machine will be approximately predicted correctly. These are all the things evaluated by the customer.

**Feedback**

After evaluation by the customer if it doesn't meet customer requirements, the customer gives some feedback. After getting the feedback, the developer solves it quickly. And if suppose the customer requirements were satisfied as well as they give good feedback and what improvements are needed in the next product.

**Chapter 7**

**MODEL OPTIMIZATION**

**7.1 Overview of Model Tuning and Best Parameters Selection**

**Model Tuning:**

When Our default parameter is not efficient so obviously the developer goes to tune the parameter in this proposed system and applies some parameter tuning. Here use various kinds of classification algorithms but all algorithms are not effective when applying model tuning algorithms with high accuracy**.**

**Best Parameter Selection:**

In any dataset have many features but no need to use all features. I supposed to use all features or unwanted features, prediction is not effective. If a user wants high and correct accuracy feature selection is a must in product development.

In this proposed system also select the best features this product predicts the future analysis of the earthquake so firstly the system can extract the features.

**Feature selection:**

The system can easily select which features depend on the predicted price columns. Here In this proposed system select given below feature, these feature all are under from best feature selection

* event\_magnitude: magnitude of the earthquake
* event\_latitude: latitude of the event recorded (degrees)
* event\_longitude: longitude of the event recorded (degrees)
* event\_depth: distance below Earth's surface where earthquake happened (km)
* station: sensor name (GPS station or seismometer)
* station\_latitude: sensor (GPS station or seismometer) latitude (degrees)
* station\_longitude: sensor (GPS station or seismometer) longitude (degrees)

**7.2 Model Tuning Process and Experiments**

**Model Tuning Process:**

In this proposed system tune the support vector classifier algorithm In support vector hyperparameter is given below.

A Machine Learning model is defined as a mathematical model with a number of parameters that need to be learned from the data. However, there are some parameters, known as Hyperparameters and those cannot be directly learned

Support Vector Machine also have some hyper-parameters (like what C or gamma values to use) and finding optimal hyper-parameter is a very hard task to solve. GridSearchCV takes a dictionary that describes the parameters that could be tried on a model to train it. The grid of parameters is defined as a dictionary, where the keys are the parameters and the values are the settings to be tested.

**GridSearchCV**

In this proposed system improve the accuracy through GridSearchCV. Here use different types of Gamma and Kernel that code is given below.

from sklearn.model\_selection import GridSearchCV

# defining parameter range

param\_grid = {'C': [0.1, 1, 10, 100, 1000],

'gamma': [1, 0.1, 0.01, 0.001, 0.0001],

'kernel': ['rbf']}

grid = GridSearchCV(SVC(), param\_grid, refit = True, verbose = 3)

# fitting the model for grid search

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

One of the great things about GridSearchCV is that it is a meta-estimator. It takes an estimator like SVC, and creates a new estimator, that behaves exactly the same – in this case, like a classifier.

What fit does is a bit more involved than usual. First, it runs the same loop with cross-validation, to find the best parameter combination.

Once it has the best combination, it runs fit again on all data passed to fit (without cross-validation), to build a single new model using the best parameter setting.

we can easily find best parameters found by GridSearchCV in the best\_params\_ attribute, and the best estimator in the best\_estimator\_ attribute

**# print best parameter after tuning**

**print(grid.best\_params\_)**

**# print how our model looks after hyper-parameter tuning**

**print(grid.best\_estimator\_)**

**Chapter 8**

**PRODUCT DELIVERY AND DEPLOYMENT**

**8.1 User Manuals**

**Need of the User manual:**

After developing the proposed system, developers obviously deliver the product. When delivery the product user manual is mandatory. Because users have no prior experience of the product because they are end users. If suppose any issue when a user uses this product, they can easily solve that issue through this user manual.

**What details are available in the User manual?**

In user manual file have process of developing and deploying but importantly user manual have how to use this product like how to get URL and paste the URL, how to handle the main file like how to know earthquake because this product full and fully develop by machine learning so machine cannot accept the categorical data but end user when give the categorical like station that means which area, application will be raise an error. Such issues are the same so developers can give solutions through the user manual.

In below what are the problem solving are available in that manual:

* Value error when click predict
* Application error
* Parse error
* Internal server error

**Value error When Click Predict**

If suppose entering the station instead of 1 system cannot accept data because the product is full and fully developed by machine learning. And developers solve this type of issue through manuals and very easily through our design page.

That means when user enter the station page will be design by select the dropdown in dropdown show many stations user can select the station then machine can be converted categorical to numerical

**Application error:**

If suppose users have some prior knowledge about development, they update some features they are solving some issue so developers give solutions to this type of issue.

This type of issue is nothing but version mismatching of some of the libraries missing when deploying the product so users go to change or update the requirements.txt file. These all solutions are provided by user manuals.

**Parse and Internal Server:**

This type of error when the user changes the data or file so the user handles the file very securely and carefully, the user directly uses a given URL without a file but the developer gives the file also.

**8.2 Delivery Schedule**

Machine learning project is cube of D that means Development Deploy and Delivery so so delivery is final stage of success in that product here developer given the timeline of product delivery

|  |  |
| --- | --- |
| **Date and No of Days** | **Which one is Deliver the customer (user)** |
| First week (3-4 days) | Project Proposal |
| Second week | Designing |
| Third week | User accept design then develop code |
| Fifth week | Styling changes in Design |
| Sixth week | Run in local server |
| Seventh week | Complete design and Processing |
| Tenth week | Deployment and solving user Problem |
| Final Stage | Product Delivery in Market |

Table 8.1 Delivery Process Schedule

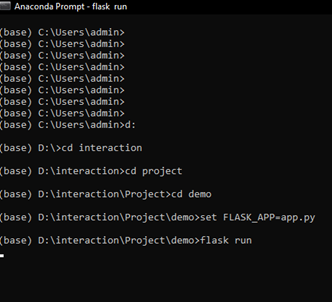
**8.3 Deployment Process**

When the product is ready to run on a local server then obviously go to deploy the project because when deployed the product user can easily use.

In this proposed system done by a local server then upload the code and files in GitHub when GitHub is done then create a Heroku account. Because of this product deployed through Heroku. Heroku provides one URL that will be used by the user.

Local Server running:

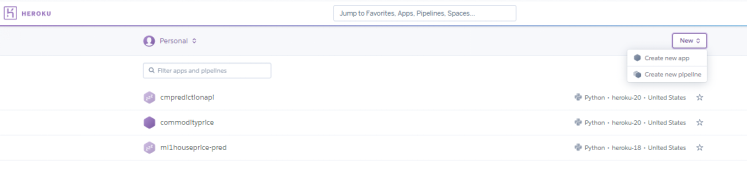
In local server running we can run localhost Running on (<http://127.0.0.1:5000/>)



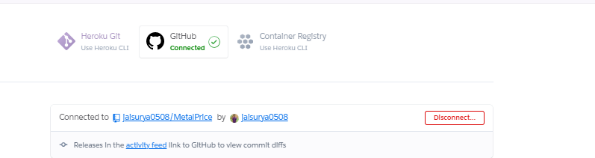
**Deploy Processing in Heroku**



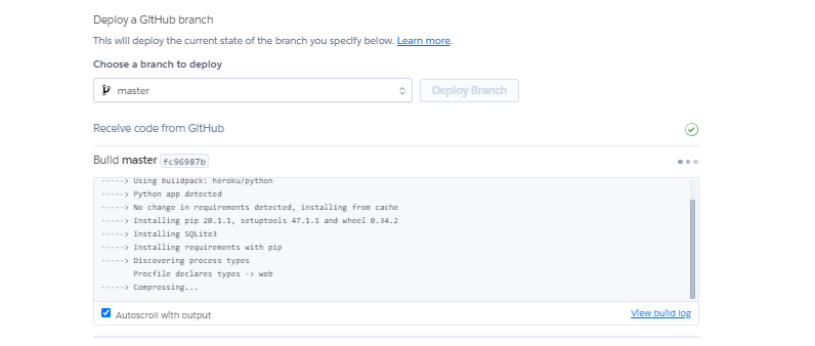
**Create new app**

****

**Connect Github repository**

****

**Deploy the Github branch**

****

**Chapter 9**

**CONCLUSION**

**9.1 Summary**

Earthquake early warning can stop many disasters and loss of human life. Traditional methods take more time to detect before an earthquake.One area of development where it demonstrated promising results is earthquake early warning (EEW), i.e. the characterization of an earthquake before it reaches sensitive areas.

Current state-of-the-art methods based on seismometers data only demonstrated an applicability limited to medium earthquakes. This proposed system uses a supervised machine learning classification approach such as support vector classifier(SVC), Random forest classifier, KNearest Neighbor(KNN) to classify the type of the earthquake.

**9.2 Limitations and Future Work**

**Limitations:**

This proposed system have some limitations such as given below:

* This method won’t perform well on both sensor data/multi sensor data.
* This method does not perform well for all locations, and classifies abnormal for it.
* The proposed method only identifies the specific stations that are available in the dataset.

**Future Work:**

In above discussing some limitation of this system in future outcomes they are overcome

all the above issues.

**Future Outcome-1:**

In the feature, time series methods will be implemented and graphs will displayed, along with that the accuracy for multi sensor data will be improved.

**Future Outcome-2:**

This process will be converted to a mobile application that will be available on the play store so the users can easily access the application.

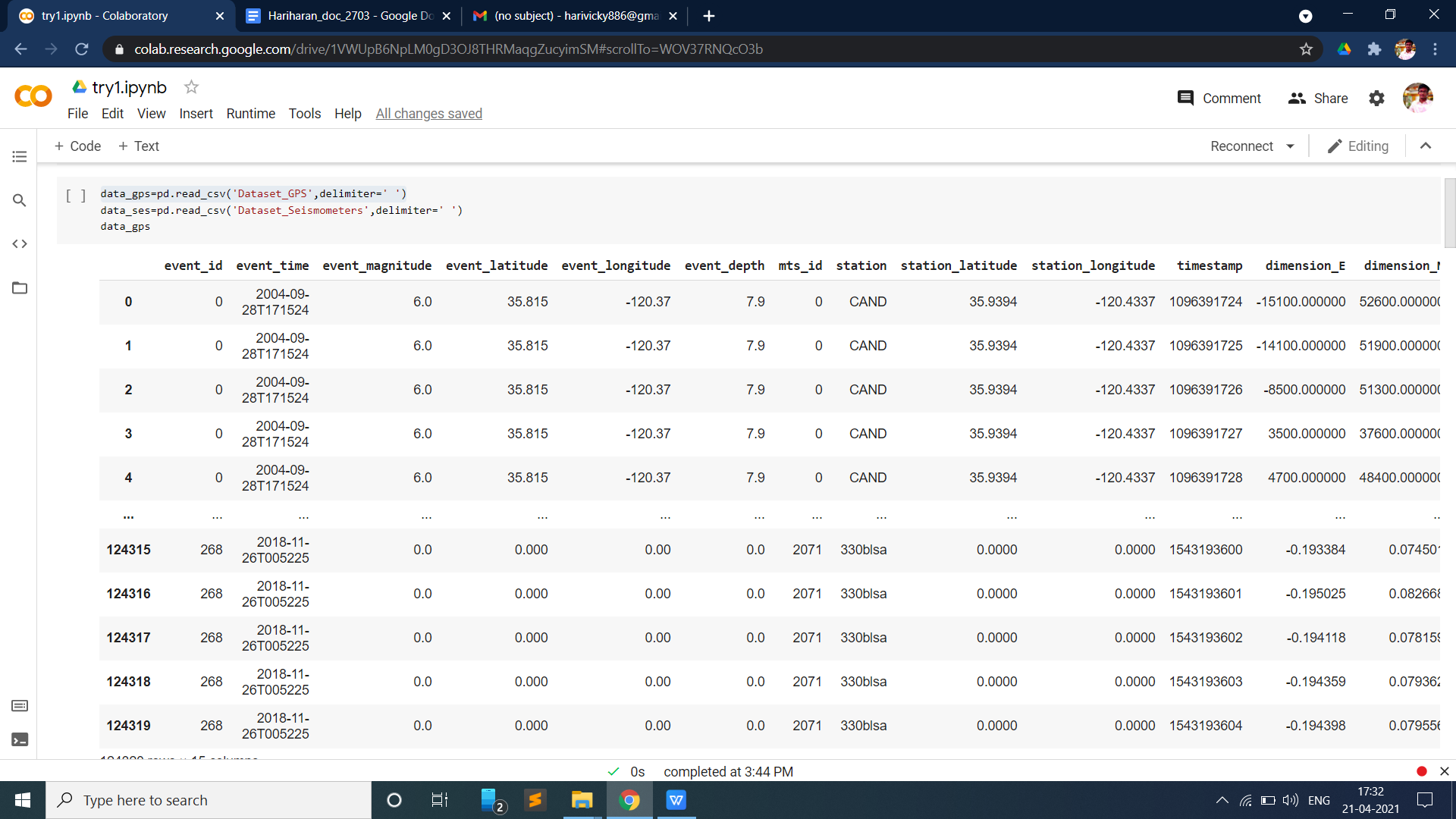
**REFERENCES**

1. Allen, R. M., and Melgar, D. 2019. Earthquake Early Warning: Advances, Scientifific Challenges, and Societal Needs. Annual Review of Earth and Planetary Sciences47:361–388.
2. Fauvel, K.; Balouek-Thomert, D.; Melgar, D.; Silva, P.; Simonet, A.; Antoniu, G.; Costan, A.; Masson, V.; Parashar,M.; Rodero, I.; and Termier, A. 2019. Earthquake Early Warning Dataset. figshare.
3. Hoshiba, M., and Ozaki, T. 2012. Earthquake Early Warning and Tsunami Warning of JMA for the 2011 off the Pacifific Coast of Tohoku Earthquake. Journal of the Seismological Society of Japan 64:155–168.
4. Manish Parashar,2 Ivan Rodero,2 Alexandre Termier Univ Rennes, Inria, CNRS, IRISA, Rennes, France. A Distributed Multi-Sensor Machine Learning Approach to Earthquake Early Warning

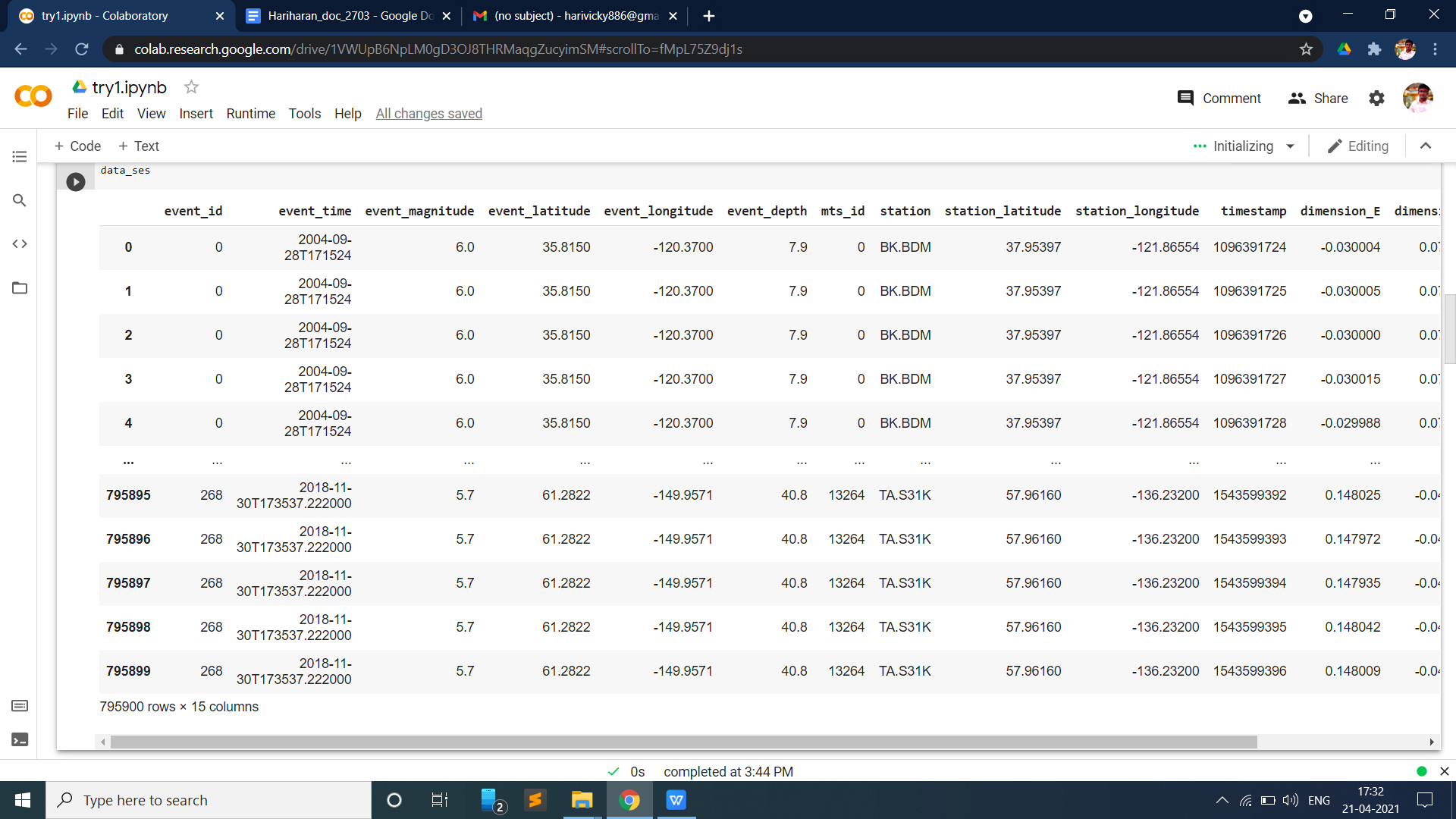
**APPENDIX-A**

**DATA SET**

The data set that containing the gps data

****

The data set that containing the seismometer data

****

**APPENDIX-B**

**SOURCE CODE**

In the source code folder given below structure because in this proposed system deliver the product through web application. The template folder has designing files.

**Main folder** **sub folder**



**app.py**

from flask import Flask, request, jsonify, render\_template

import pickle

import numpy as np

app=Flask(\_\_name\_\_)

model=pickle.load(open('model2.pkl','rb'))

@app.route('/')

def home():

return render\_template('index.html')

@app.route('/predict',methods=['POST'])

def predict():

int\_features =[int(x) for x in request.form.values()]

final\_features=[np.array(int\_features)]

prediction=model.predict(final\_features)

output=round(prediction[0],2)

return render\_template('index.html',prediction\_text='Type of the earthquake is {}'.format(output))

@app.route('/predict\_api',methods=['POST'])

def predict\_api():

data= request.get\_json(force=True)

prediction= model.predict([np.array(list(data.values()))])

output=prediction[0]

return jsonify(output)

if \_\_name\_\_ ==”\_\_main\_\_”:

app.run(debug=True)

**Model.py**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import StandardScaler

from sklearn.preprocessing import LabelEncoder

data\_gps=pd.read\_csv('Dataset\_GPS',delimiter=' ')

le=LabelEncoder()

data\_gps['station']=le.fit\_transform(data\_gps['station'])

sc=StandardScaler()

scale=['event\_magnitude','event\_latitude','event\_longitude','event\_depth','mts\_id','station','station\_latitude','station\_longitude','timestamp','dimension\_E','dimension\_N','dimension\_Z']

data\_gps[scale]=sc.fit\_transform(data\_gps[scale])

x=data\_gps.drop(columns=['label','event\_time','timestamp'],axis=1)

y=data\_gps['label']

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

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.15)

from sklearn.svm import SVC

classifier = SVC(kernel = 'rbf', random\_state = 0)

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

#saving model to disk

pickle.dump(classifier, open('model2.pkl','wb'))

#loading model to compare results

model=pickle.load(open('model2.pkl','rb'))

test=[[0,6.0,35.815,-120.37,7.9,0,704,35.9394,-120.4337,33100.0,-5800.0,44400.0]]

classifier.predict(test)

**index.html**

<!DOCTYPE html>

<html>

<head>

<title>Classification of earthquake</title>

<link rel="stylesheet" type="text/css" href="{{ url\_for('static', filename='css/styles.css') }}">

</head>

<body background-image=url("gold.jpg")>

<div class= "ml-container">

<h1>Classification of earthquake early warning</h1>

<form action="{{url\_for('predict')}}"method="post">

<input type="text" name="e\_mag" placeholder="event\_magnitude" required="required"/><br><br>

<input type="text" name="e\_lat" placeholder="event\_latitude" required="required"/><br><br>

<input type="text" name="e\_log" placeholder="event\_longitude" required="required"/><br><br>

<input type="text" name="e\_d" placeholder="event\_depth" required="required"/><br><br>

<input type="text" name="stn" placeholder="station" required="required"/><br><br>

<input type="text" name="stn\_lat" placeholder="station\_latitude" required="required"/><br><br>

<input type="text" name="stn\_log" placeholder="station\_longitude" required="required"/><br><br>

<button type="submit" value="predict"></button><br>

</form><br> <br>

{{prediction\_text}}</div></body>

</html>

**requirements.txt**

Flask==1.1.1

gunicorn==19.9.0

itsdangerous==1.1.0

jinja2==2.10.1

MarkupSafe==1.1.1

Werkzeug==0.15.5

numpy>=1.9.2

scipy>=0.15.1

scikit-learn>=0.18

matplotlib>=1.4.3

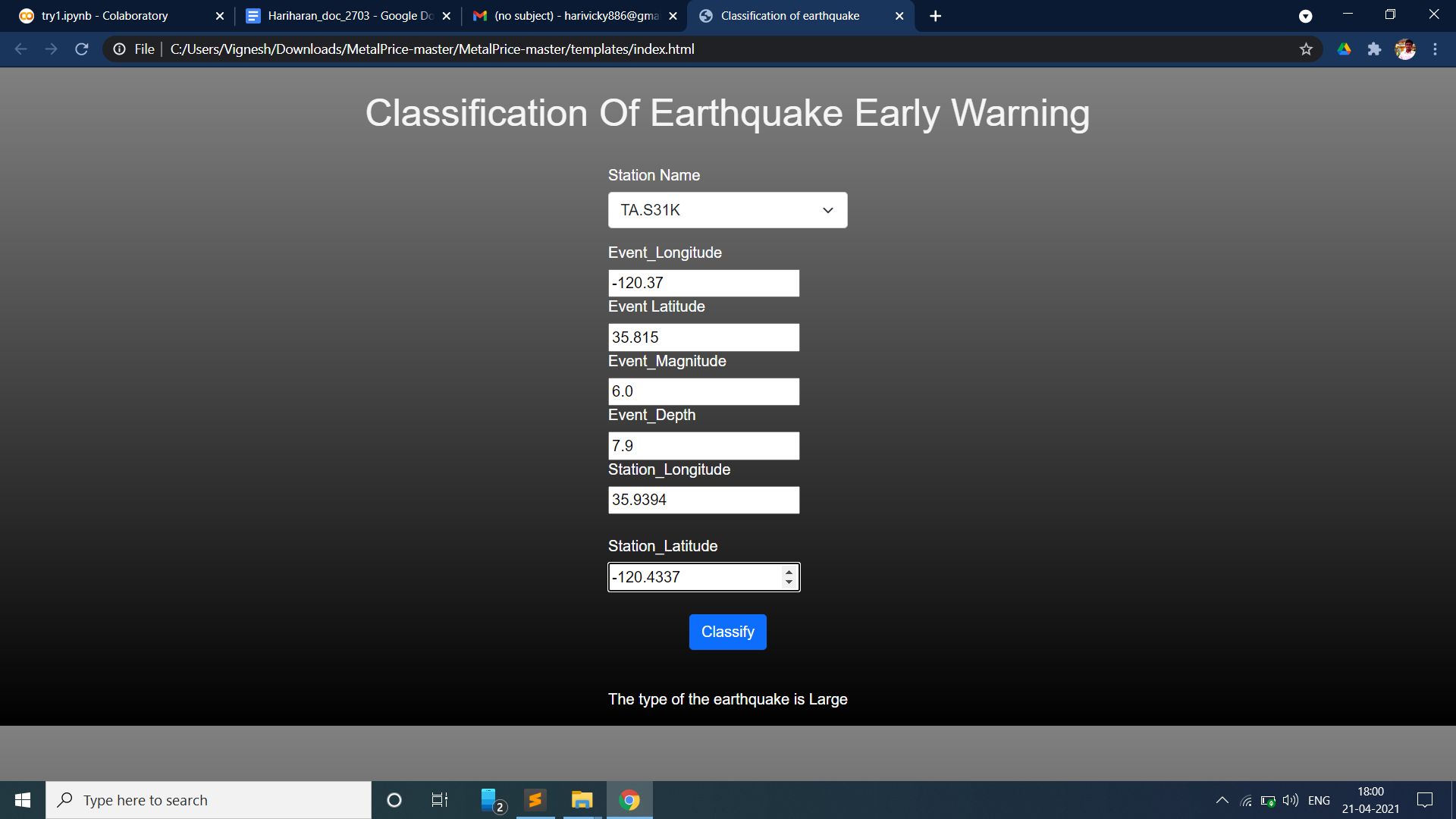
pandas>=0.19

**APPENDIX-C**

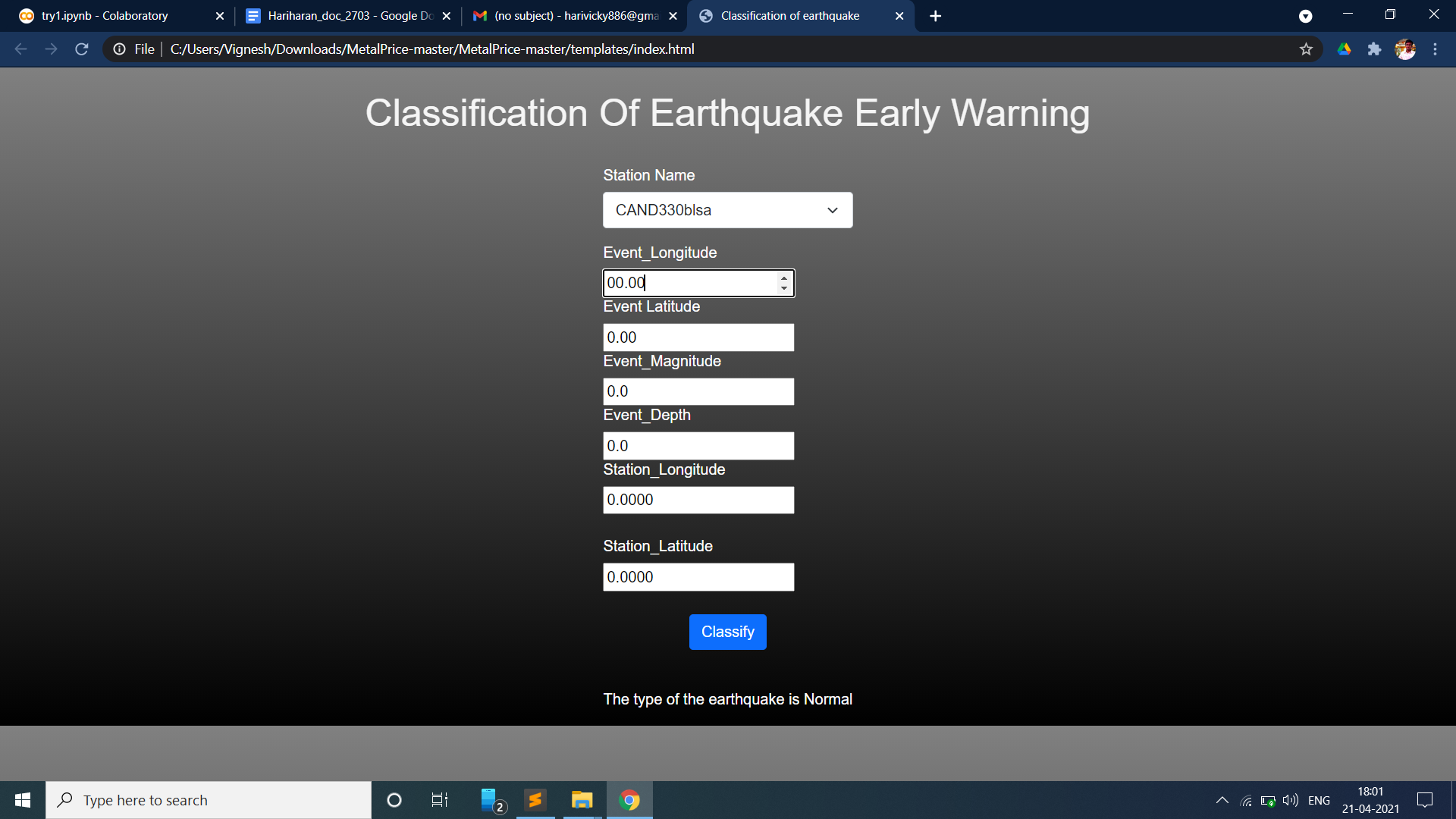
**OUTPUT SCREEN SHOTS**

**Validation and outputs**

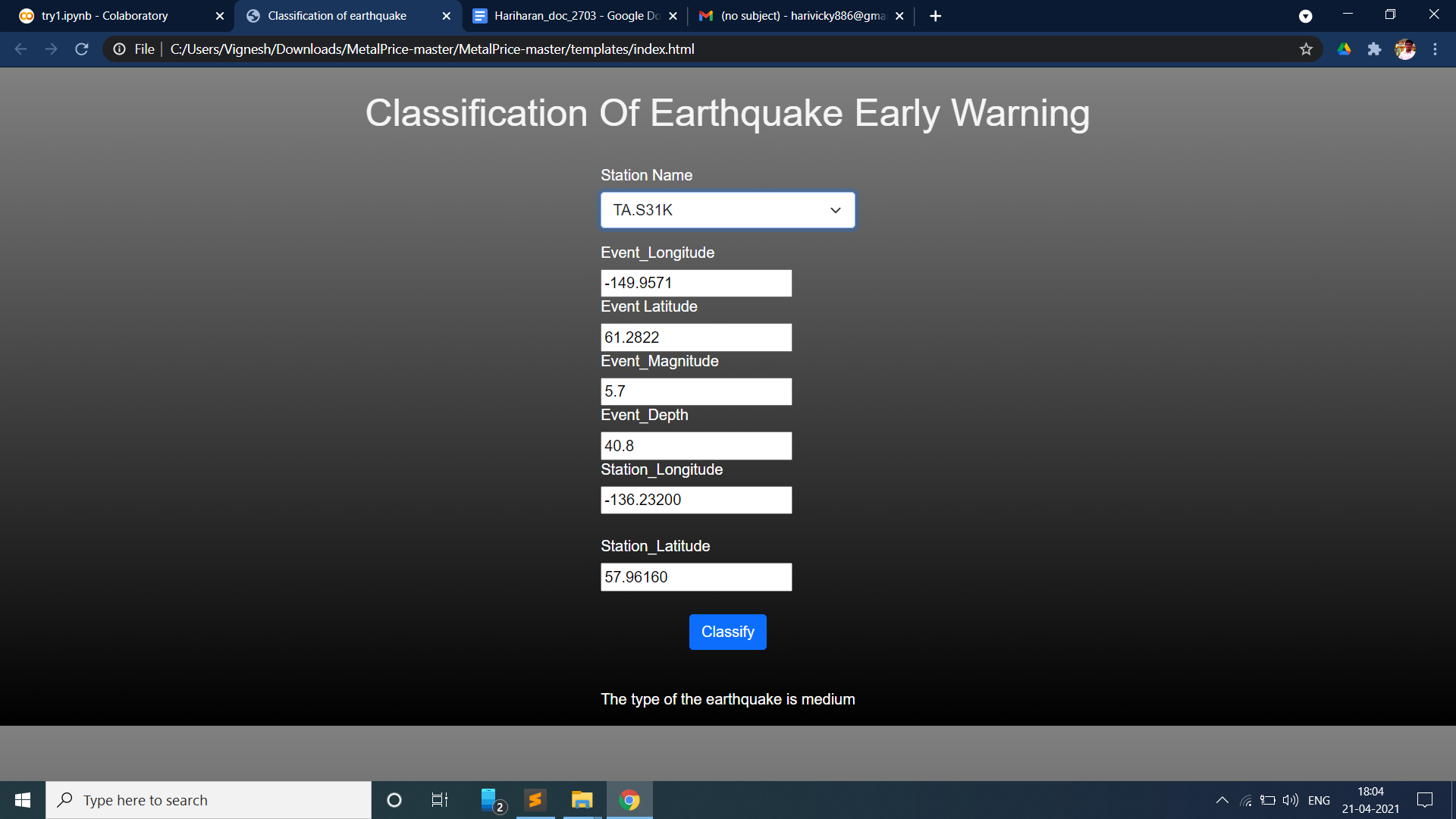
Output for large earthquake

****

Output for normal activity

****

Output for medium activity

****