

FORECASTING EARTHQUAKE AFTERSHOCK LOCATIONS WITH AI-ASSISTED SCIENCE

A Project Work Synopsis

Submitted in the partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

INTERNET-OF-THINGS

Submitted by:

Abhishek Singh

Shefali Yadav

Rishabh Anand

University Roll Number

19BCS4508

19BCS4524

19BCS4525

Under the Supervision of:

Mr. Nikhil Aggarwal



**CHANDIGARH
UNIVERSITY**

Discover. Learn. Empower.

CHANDIGARH UNIVERSITY, GHARUAN, MOHALI - 140413,

PUNJAB

MONTH & YEAR

Table of Contents

- 1. INTRODUCTION***
 - Problem Definition
 - Project Overview/Specifications* (page-1 and 3)
 - Hardware Specification
 - Software Specification
- 2. LITERATURE SURVEY**
- 3. PROBLEM FORMULATION**
- 4. RESEARCH OBJECTIVES**
- 5. METHODOLOGY**
- 6. TENTATIVE CHAPTER PLAN FOR THE PROPOSED WORK**
- 7. REFERENCES**
- 8. APPENDICES**

1 INTRODUCTION

From hurricanes and floods to volcanoes and earthquakes, the Earth is continuously evolving in fits and spurts of dramatic activity. Earthquakes and subsequent tsunamis alone have caused massive destruction in the last decade—even over the course of writing this post, there were earthquakes in New Caledonia, Southern California, Iran, and Fiji, just to name a few.

Earthquakes typically occur in sequences: an initial "mainshock" (the event that usually gets the headlines) is often followed by a set of "aftershocks." Although these aftershocks are usually smaller than the main shock, in some cases, they may significantly hamper recovery efforts.

Although the timing and size of aftershocks has been understood and explained by established empirical laws, forecasting the locations of these events has proven more challenging.

Aftershocks are a response to changes in stress generated by large earthquakes and represent the most common observations of the triggering of earthquakes. The maximum magnitude of aftershocks and their temporal decay are well described by empirical laws (such as Bath's law and Omori's law), but explaining and forecasting the spatial distribution of aftershocks is more difficult.

1.1 Problem definition

The present project aims at forecasting the location of aftershocks that follow the occurrence of any large earthquake. While main shocks are not (yet) predictable, aftershock statistics are sound enough for us to predict them in reasonably small magnitude-space-time windows.

1.2 Problem overview

Here we use a deep-learning approach to identify a static-stress-based criterion that forecasts aftershock locations without prior assumptions about fault orientation. We show that a neural network trained on more than 131,000 main shock–aftershock pairs can predict the locations of aftershocks in an independent test dataset of more than 30,000 main shock–aftershock pairs more accurately (area under curve of 0.849) than can classic Coulomb failure stress change (area under curve of 0.583). We find that the learned aftershock pattern is physically interpretable: the maximum change in shear stress, the von Mises yield criterion (a scaled version of the second invariant of the deviatoric stress-change tensor) and the sum of the absolute values of the independent components of the stress-change tensor each explain more than 98 per cent of the variance in the neural-network prediction. This machine-learning-driven insight provides improved forecasts of aftershock locations and identifies physical quantities that may control earthquake triggering during the most active part of the seismic cycle.

1.3 Hardware specifications

This project requires a tremendous amount of computing power. High performance graphical processing units (GPUs) are ideal because they can handle a large volume of calculations in multiple cores with copious memory available. However, managing multiple GPUs on-premises can create a large demand on internal resources and be incredibly costly to scale.

1.4 Software specification

Model Building and Optimization : tensor flow, Jupyter Notebook , Python , Matplotlib, Seaborn ,Pandas.

2 LITERATURE REVIEW

1. COULOMB FAILURE STRESS CHANGE :

The Coulomb failure stress (CFS) criterion is the most commonly used method for predicting spatial distributions of aftershocks following large earthquakes. However, large uncertainties are always associated with the calculation of Coulomb stress change. The uncertainties mainly arise due to nonunique slip inversions and unknown receiver faults; especially for the latter, results are highly dependent on the choice of the assumed receiver mechanism. Based on binary tests (aftershocks yes/no), recent studies suggest that alternative stress quantities, a distance-slip probabilistic model as well as deep neural network (DNN) approaches, all are superior to CFS with predefined receiver mechanism. To challenge this conclusion, which might have large implications, we use 289 slip inversions from SRCMOD database to calculate more realistic CFS values for a layered half-space and variable receiver mechanisms. We also analyze the effect of the magnitude cutoff, grid size variation, and aftershock duration to verify the use of receiver operating characteristic (ROC) analysis for the ranking of stress metrics.

The observations suggest that introducing a layered half-space does not improve the stress maps and ROC curves. However, results significantly improve for larger aftershocks and shorter time periods but without changing the ranking. We also go beyond binary testing and apply alternative statistics to test the ability to estimate aftershock numbers, which confirm that simple stress metrics perform better than the classic Coulomb failure stress calculations and are also better than the distance-slip probabilistic model.

Despite its frequent application for several decades, Coulomb failure stress calculations have been questioned by recent studies and shown to be outperformed by other stress scalars and state-of-the-art methods like deep neural network in forecasting aftershocks. However, the recent results are also questionable because of an artificial DNN application (Mignan & Broccardo, 2019) as well as simplified CFS calculations. As this has broad implication for this research area, we performed a comprehensive reanalysis of the previous ROC-based study. Here we include CFS metrics accounting for the variability of aftershock mechanisms and additionally taking account of the incompleteness of catalogs as well as the occurrence of background activity. In addition to the previously conducted ROC analysis for binary forecasts, we also tested forecasts of aftershock numbers. To summarize, we find that the results of the ROC analysis are dependent on the magnitude cutoff, aftershock duration, and grid sizes and that more realistic CFS calculations (OOP and VM) can significantly improve the results.

However, our analysis verifies that the stress scalars MS and VMS, and distance-slip probabilistic model (R), all of which do not rely on any specification of receiver mechanisms, outperform on average the CFS metrics in all test setups. While CFS might still be used for the evaluation of the stress changes on well-defined fault segments, our results indicate that spatial forecasts of the aftershock density might be generally improved by using von-Mises stress (VMS) instead of Coulomb stress.

2. RESEARCH BY HARVARD AND GOOGLE MACHINE LEARNING EXPERTS:

They started with a database of information on more than 118 major earthquakes from around the world. From there, they applied a neural network to analyze the relationships between static stress changes caused by the main shocks and aftershock locations. The algorithm was able to identify useful patterns.

The end result was an improved model to forecast aftershock locations and while this system is still imprecise, it's a motivating step forward. Machine learning-based forecasts may one day help deploy emergency services and inform evacuation plans for areas at risk of an aftershock. When they applied neural networks to the data set, they were able to look under the hood at the specific combinations of factors that it found important and useful for that forecast, rather than just taking the forecasted results at face value. This opens up new possibilities for finding potential physical theories that may allow us to better understand natural phenomena.

3 PROBLEM FORMULATION

We have applied a neural net to analyze the relationships between static stress changes caused by the mainshocks and aftershock locations. The algorithm was able to identify useful patterns.

The end result was an improved model to forecast aftershock locations and while this system is still imprecise, it's a motivating step forward. Machine learning-based forecasts may one day help deploy emergency services and inform evacuation plans for areas at risk of an aftershock.

4 OBJECTIVES

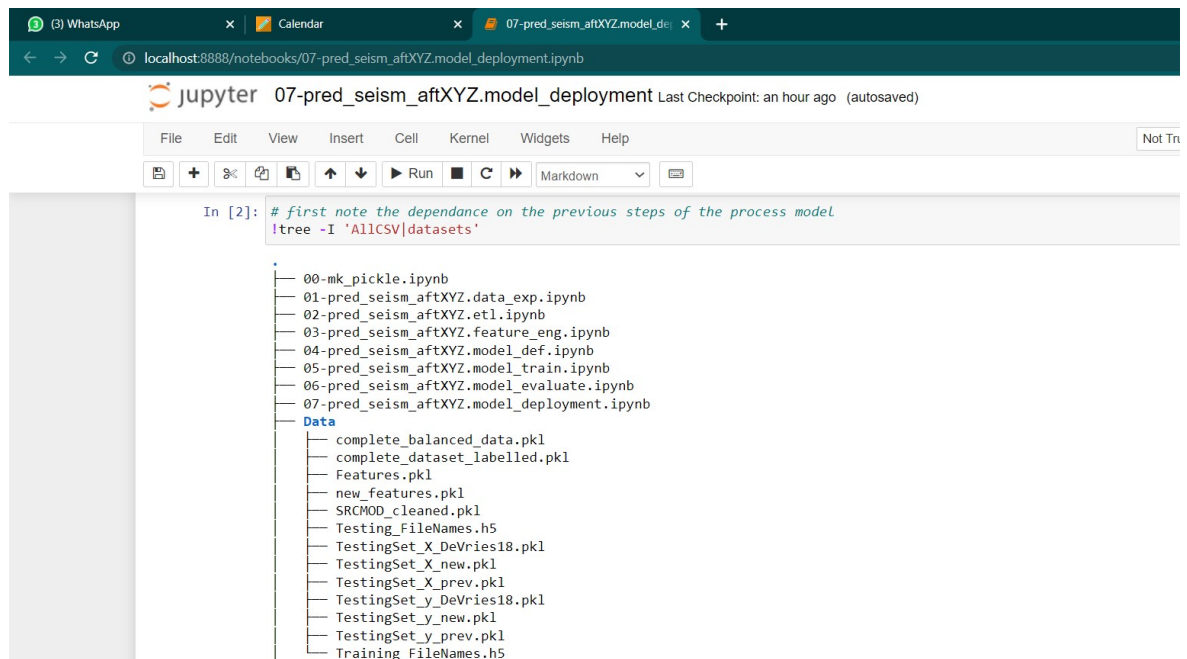
The proposed work is aimed to carry out work leading to the development of an approach for forecasting aftershock locations. The proposed aim will be achieved by dividing the work into following objectives:

1. This project aims at forecasting the location of aftershocks that follow the occurrence of any large earthquake
2. Aftershock forecasts can help a wide range of users including the public, emergency managers, and lifeline engineers prepare for, respond to, cope with, and recover from earthquake disasters.
3. By bypassing stress computation entirely, using a simple ANN topology, and only 2 mainshock-rupture-based features, we can streamline the process of aftershock pattern prediction for future post-mainshock crisis management (i.e. process faster and more transparent);

5 METHODOLOGY

The following methodology will be followed to achieve the objectives defined for proposed research work:

- The idea is to observe a volume which extends 100 km horizontally and 50 km vertically from the main shock. We then break that volume into 5km x 5km x 5km small volumes and calculate the elastic stress change tensors at each of their centroid.
- Now using that information we need to predict whether there was an aftershock in that small volume or not. In order to know the ground truth we use International Seismological Center(ISC) event catalogue, in which for each main shock we looked up for its corresponding aftershocks from 1 sec to 1 year time and using that information we created our ground truth i.e. whether there was an aftershock in that small volume or not.
- So this whole problem is now a binary classification problem, in which our neural network has to predict was there an aftershock in that small 5km x 5km x 5km region or not. The model will use deep learning techniques to predict whether there can be aftershock at a particular locations or not. I'll be taking the data provided by the SRCMOD <http://equake-rc.info/SRCMOD/searchmodels/allevnts/>.
- The format of the data will be FSP (finite-source rupture model). We'll not be working on how to process those SRCMOD files in order to create a csv. Rather we'll use already created CSVs.
- The data provided by the SRCMOD file contains the information about the hypocenter, latitude, longitude, magnitude, strike, dip, the inversion parameters and many other relevant data that is sufficient to get an insight of the earthquake.
- We'll be feeding these data to the neural network having several hidden layers (it depends on you, try and experiment with it), which will then try to extract the important features in order to predict the whether there is a chance of having an aftershock or not. we'll be using two activation functions tanh and ReLu (again experiment with it). The output layer will be a sigmoid layer, which will give us a probability between 0 - 1, as to how confident it is.

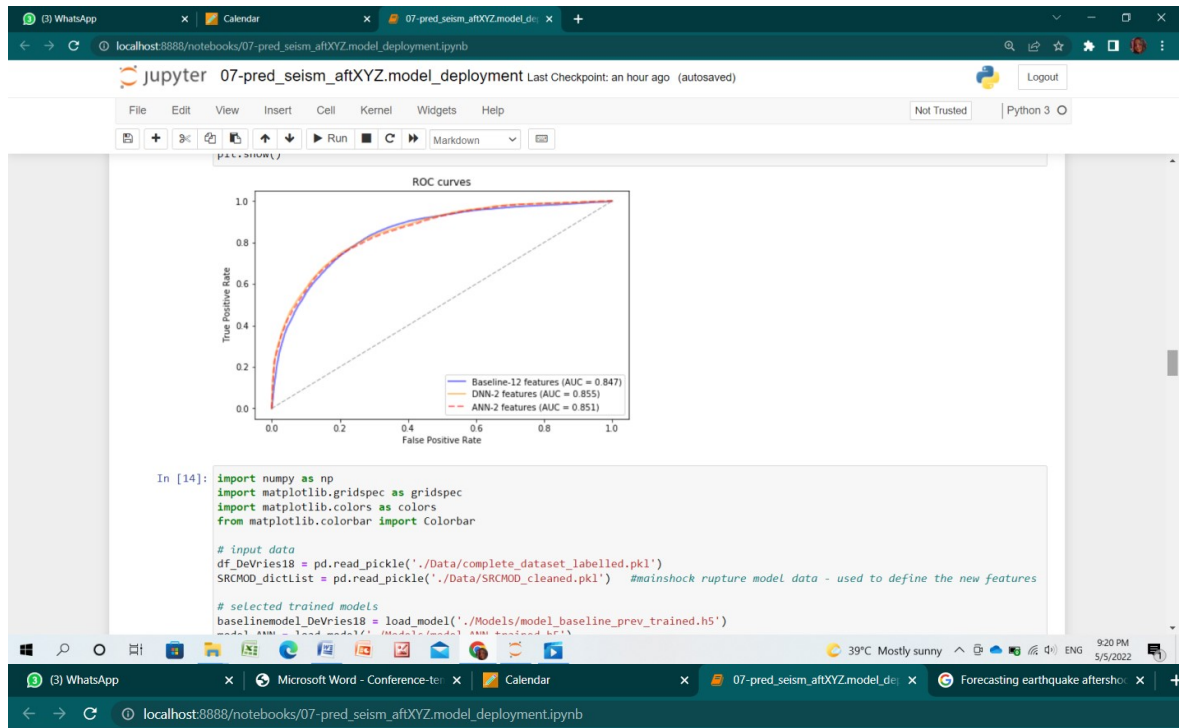


The screenshot shows a Jupyter Notebook interface with a browser window at the top displaying the URL `localhost:8888/notebooks/07-pred_seism_aftXYZ.model_deployment.ipynb`. The notebook title is `07-pred_seism_aftXYZ.model_deployment` and it shows a last checkpoint from an hour ago. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations and execution. The main area shows a code cell with the following content:

```
In [2]: # first note the dependance on the previous steps of the process model
!tree -I 'AllCSV|datasets'
```

The output of the command is a file tree structure:

```
.
├── 00-mk_pickle.ipynb
├── 01-pred_seism_aftXYZ.data_exp.ipynb
├── 02-pred_seism_aftXYZ.etl.ipynb
├── 03-pred_seism_aftXYZ.feature_eng.ipynb
├── 04-pred_seism_aftXYZ.model_def.ipynb
├── 05-pred_seism_aftXYZ.model_train.ipynb
├── 06-pred_seism_aftXYZ.model_evaluate.ipynb
├── 07-pred_seism_aftXYZ.model_deployment.ipynb
├── Data
│   ├── complete_balanced_data.pkl
│   ├── complete_dataset_labelled.pkl
│   ├── Features.pkl
│   ├── new_features.pkl
│   ├── SRCMOD_cleaned.pkl
│   ├── Testing_FileNames.h5
│   ├── TestingSet_X_DeVries18.pkl
│   ├── TestingSet_X_new.pkl
│   ├── TestingSet_X_prev.pkl
│   ├── TestingSet_y_DeVries18.pkl
│   ├── TestingSet_y_new.pkl
│   ├── TestingSet_y_prev.pkl
│   └── Training_FileNames.h5
```

Jupyter 07-pred_seism_afXYZ.model_deployment Last Checkpoint: 2 hours ago (autosaved)

```
#Loop over slip distributions
for i in range(len(eventTags)):

    #read in data
    df_eventTag = df_DeVries18.loc[(df_DeVries18['ID'] == eventTags[i])
                                   & (df_DeVries18['z'] == depth_vec[i])]
    grid_aftershock_count = np.double(df_eventTag['aftershocksyn'])

    #BASELINE: run model prediction for this slip distribution - we first define the 12 features
    df_eventTag_DeVries18_origFeatures = df_eventTag[['stresses_full_xx', 'stresses_full_xy',
                                                         'stresses_full_yy', 'stresses_full_xz', 'stresses_full_yz', 'stresses_full_zz',
                                                         'aftershocksyn', 'ID']]
    origFeatures = ['stresses_full_xx', 'stresses_full_xy', 'stresses_full_yy', 'stresses_full_xz', 'stresses_full_yz', 'stresses_full_zz',
                    'posabsxx', 'posabsxy', 'posabsyy', 'posabsxz', 'posabsyz', 'posabszz']
    negabs = ['negabsxx', 'negabsxy', 'negabsyy', 'negabsxz', 'negabsyz', 'negabszz']
    df_eventTag_DeVries18_engFeatures = pd.DataFrame()
    df_eventTag_DeVries18_engFeatures[['ID', 'aftershocksyn']] = df_eventTag_DeVries18_origFeatures[['ID', 'aftershocksyn']]
    df_eventTag_DeVries18_engFeatures[posabs] = abs(df_eventTag_DeVries18_origFeatures[origFeatures]) * 1e-6
    df_eventTag_DeVries18_engFeatures[negabs] = -abs(df_eventTag_DeVries18_origFeatures[origFeatures]) * 1e-6
    features = ['posabsxx', 'posabsxy', 'posabsyy', 'posabsxz', 'posabsyz', 'posabszz',
                'negabsxx', 'negabsxy', 'negabsyy', 'negabsxz', 'negabsyz', 'negabszz']
    target = 'aftershocksyn'
    x_test_EventTag = df_eventTag_DeVries18_engFeatures[features]
    y_test_EventTag = df_eventTag_DeVries18_engFeatures[target]
    pred_baseline = baselinemodel_DeVries18.predict(x_test_EventTag)
```

```
target = aftershocksyn
x_test_EventTag = df_NewFeatures_norm[features]
# y_test_EventTag = df_NewFeatures_norm[target]
pred_ANN = model_ANN.predict(x_test_EventTag)

# AUC estimates
fpr_baseline, tpr_baseline, _ = metrics.roc_curve(y_test_EventTag, pred_baseline)
auc_baseline = metrics.auc(fpr_baseline, tpr_baseline)
fpr_ANN, tpr_ANN, _ = sklearn.metrics.roc_curve(y_test_EventTag, pred_ANN)
auc_ANN = metrics.auc(fpr_ANN, tpr_ANN)

x_temp = np.double(df_eventTag['x'])
y_temp = np.double(df_eventTag['y'])
grid_aftershock_count_temp = np.double(df_eventTag['aftershocksyn'])

for j in range(0, 2):
    ax = plt.subplot(gs[j*2, i*rowscale])
    contour_levels = np.linspace(min_val_big, max_val_big, 100)

    if j==0: #if baseline
        field_temp = pred_baseline
    else: #if DNN
        field_temp = pred_ANN

    field_temp = field_temp[:,0]
    field_temp[np.where(field_temp>=max_val_big)] = max_val_big - 0.001
    field_temp[np.where(field_temp<min_val_big)] = min_val_big + 0.001
```

```
# slip_flt = np.asarray(SRCMOD_eventTag['slip'])
# color_slip = [str(item/255.) for item in slip_flt]
# plt.scatter(x_flt, y_flt, c=color_slip, marker='o', s=20)

# count and plot aftershocks at the depth of interest
n_cells = 0
for i_isc in range(0, len(x_temp)):
    if grid_aftershock_count_temp[i_isc] > 0:
        plt.plot(x_temp[i_isc], y_temp[i_isc], 's', color = [0.0, 0.0, 0.0], markersize=5)
        n_cells += 1

# add labels
xpos = [0.07, 0.3, 0.48, 0.67]
ypos = [0.87, 0.45]
spacing = 0.022
stringlabel = sublabels[j][i]
plt.text(xpos[i], ypos[j], stringlabel, fontweight = 'bold', fontsize=fontsize, ha='left', va='center', tr
if j == 0: plt.text(xpos[i], ypos[j]-spacing, '$\mathrm{Baseline}$', fontsize=fontsize, ha='
if j == 1: plt.text(xpos[i], ypos[j]-spacing, '$\mathrm{ANN}$', fontsize=fontsize, ha='left', va='center',
plt.text(xpos[i], ypos[j]-2*spacing, 'z = ' + str(abs(depth_vec[i])/1e3) + ' km', fontsize=fontsize, ha='l
if j == 0: plt.text(xpos[i], ypos[j]-3*spacing, 'AUC = ' + str(round(auc_baseline,2)), fontsize=fontsize,
if j == 1: plt.text(xpos[i], ypos[j]-3*spacing, 'AUC = ' + str(round(auc_ANN,2)), fontsize=fontsize, ha='l
pos = ax.get_position() # get the original position
if j == 1: ax.set_position([pos.x0, pos.y0-0.017, pos.width, pos.height])

#plot decorations
```


07-pred_seism_aftXYZ.model_deployment

```
In [14]: import numpy as np
import matplotlib.gridspec as gridspec
import matplotlib.colors as colors
from matplotlib.colorbar import Colorbar

# input data
df_DeVries18 = pd.read_pickle('./Data/complete_dataset_labelled.pkl')
SRCMOD_dictList = pd.read_pickle('./Data/SRCMOD_cleaned.pkl') #mainshock rupture model data - used to define the

# selected trained models
baselinemodel_DeVries18 = load_model('./Models/model_baseline_prev_trained.h5')
model_ANN = load_model('./Models/model_ANN_trained.h5')

#code modified from PlotThreeTestCases.py of DeVries18
def truncate_colormap(cmap, minval=0.0, maxval=1.0, n=100):
    new_cmap = colors.LinearSegmentedColormap.from_list(
        'trunc({n},{a:.2f},{b:.2f})'.format(n=cmap.name, a=minval, b=maxval),
        cmap(np.linspace(minval, maxval, n)))
    return new_cmap

#define figure parameters
min_val_big = 0.2
max_val_big = 0.8
fontsize = 18 #22
```

07-pred_seism_aftXYZ.model_deployment

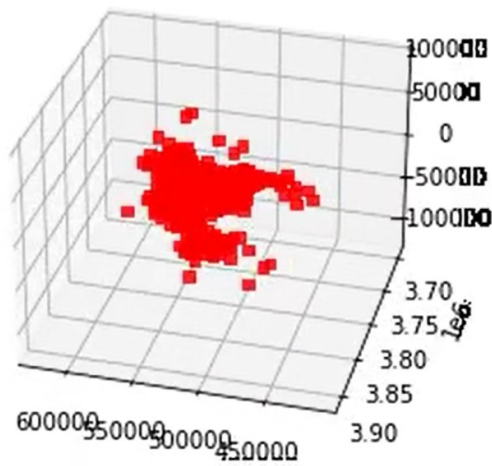
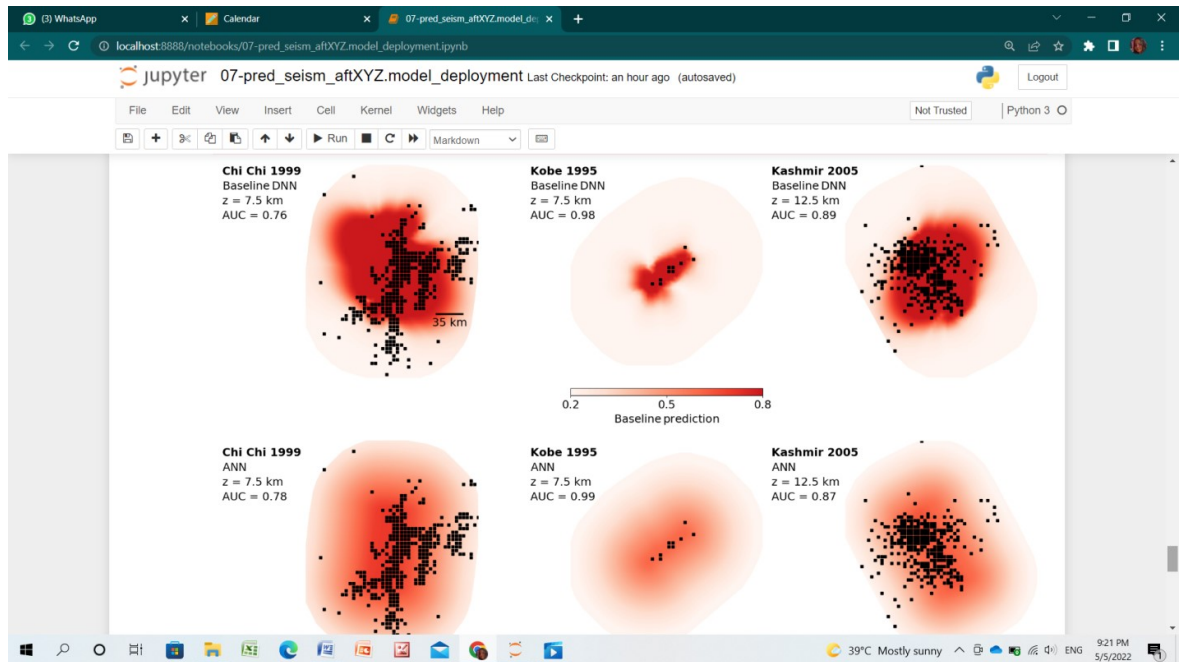
```
height_ratios=[30, 1, 30, 1]
)

rowscale = 2
cmap = plt.get_cmap('Reds')
new_cmap = truncate_colormap(cmap, 0.0, 0.75)

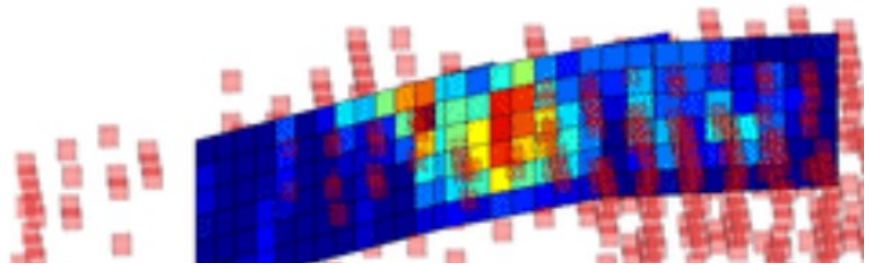
#Loop over slip distributions
for i in range(len(eventTags)):

    #read in data
    df_eventTag = df_DeVries18.loc[(df_DeVries18['ID'] == eventTags[i])
                                   & (df_DeVries18['z'] == depth_vec[i])]
    grid_aftershock_count = np.double(df_eventTag['aftershocksyn'])

    #BASELINE: run model prediction for this slip distribution - we first define the 12 features
    df_eventTag_DeVries18_origFeatures = df_eventTag[['stresses_full_xx', 'stresses_full_xy',
                                                         'stresses_full_yy', 'stresses_full_xz', 'stresses_full_yz', 'stresses_full_zz',
                                                         'aftershocksyn', 'ID']]
    origFeatures = ['stresses_full_xx', 'stresses_full_xy', 'stresses_full_yy', 'stresses_full_xz', 'stresses_full_yz', 'stresses_full_zz',
                    'posabsxx', 'posabsxy', 'posabsyy', 'posabsxz', 'posabsyz', 'posabszz']
    negabs = ['negabsxx', 'negabsxy', 'negabsyy', 'negabsxz', 'negabsyz', 'negabszz']
    df_eventTag_DeVries18_engFeatures = pd.DataFrame()
    df_eventTag_DeVries18_engFeatures[['ID', 'aftershocksyn']] = df_eventTag_DeVries18_origFeatures[['ID', 'aftershocksyn']]
    df_eventTag_DeVries18_engFeatures[posabs] = abs(df_eventTag_DeVries18_origFeatures[origFeatures]) * 1e-6
    df_eventTag_DeVries18_engFeatures[negabs] = -abs(df_eventTag_DeVries18_origFeatures[origFeatures]) * 1e-6
    features = ['posabsxx', 'posabsxy', 'posabsyy', 'posabsxz', 'posabsyz', 'posabszz',
                'negabsxx', 'negabsxy', 'negabsyy', 'negabsxz', 'negabsyz', 'negabszz']
```



information on more than 118 major earthquakes from around the world.



A visual representation of the 1992 magnitude 7.3 southern California Landers earthquake where the multi-colored portion represents the initial quake and the red boxes represent aftershock locations.

6 TENTATIVE CHAPTER PLAN FOR THE PROPOSED WORK

CHAPTER 1: INTRODUCTION

This chapter will cover the overview of whole project

CHAPTER 2: LITERATURE REVIEW

This chapter include the literature available for crop disease classification The findings of the researchers will be highlighted which will become basis of current implementation.

CHAPTER 2: BACKGROUND OF PROPOSED METHOD

This chapter will provide introduction to the concepts which are necessary to understand the proposed system.

CHAPTER 4: METHODOLOGY

This chapter will cover the technical details of the proposed approach.

CHAPTER 5: EXPERIMENTAL SETUP

This chapter will provide information about the subject system and tools used for evaluation of proposed method.

CHAPTER 6: RESULTS AND DISCUSSION

The result of proposed technique will be discussed in this chapter.

CHAPTER 7: CONCLUSION AND FUTURE SCOPE

The major finding of the work will be presented in this chapter. Also directions for extending the current study will be discussed.

PUBLICATIONS (Optional)

REFERENCES

7 REFERENCES

- <https://www.blog.google/technology/ai/forecasting-earthquake-aftershock-locations-ai-assisted-science/>
- <https://www.nature.com/articles/s41586-018-0438-y#author-information>
- <https://www.nature.com/articles/d41586-018-06091-z>
- <https://phys.org/news/2018-08-ai-quake-aftershocks.html>
- Båth, M. Lateral inhomogeneities of the upper mantle. *Tectonophysics* **2**, 483–514 (1965).
- Utsu, T. A statistical study on the occurrence of aftershocks. *Geophys. Mag.* **30**, 521–605 (1961).
- Parsons, T., Stein, R. S., Simpson, R. W. & Reasenber, P. A. Stress sensitivity of fault seismicity: a comparison between limited-offset oblique and major strike-slip faults. *J. Geophys. Res.* **104**, 20183–20202 (1999).