Imperial College London

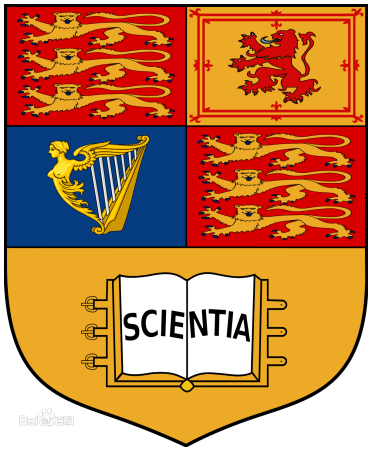
Department of Earth Science and Engineering

MSc in Applied Computational Science and Engineering

Forecasting induced seismicity in Oklahoma

Author:
Zhiyong Liu
zl1220@ic.ac.uk
Github: acse-liuzyon

Supervisor:
Dr. Stephen P. Hicks
s.hicks@imperial.ac.uk



August 2021

GitHub repository: https://github.com/acse-2020/acse2020-acse9-finalreport-liuzyon

Contents

Abstract

Human activities can cause minor earthquakes, which changes the stresses and strains on the earth's crust. These earthquakes are called induced earthquakes and most have a low magnitude. Multi-year research by the United States Geological Survey (USGS) published in 2015 showed that most of large earthquakes occured in Oklahoma may have been caused by deep injection of wastewater from the oil industry. In this project, a unique and rich dataset of industrial activities from regions in Oklahoma are used and correlations between these activities and seismicity are confirmed. With existing high-quality earthquake catalogues in these regions, possible signatures of human-induced seismicity are retrospectively forecasted through machine learning techniques. At the end of the project, a working forecasting model is generated and tested the permormance by the test dataset splitted from total dataset.

Key Words: induced earthquakes, machine learning method, forecasting model

1 Introduction

It is well known that humans can cause earthquakes through fluid injection and extraction. Ellsworth stated the understanding of the causes and mechanics of human-induced earthquakes. It includes wastewater injection, emerging oil and gas recovery technologies, and other indirect induced activities, including deep fluid injection (Ellsworth 2013). Norbeck and Rubinstein genreated a model based on fluid flow and seismic physics, which associate the past injection trends with the seismicity patterns (Norbeck & Rubinstein 2018). Such cases of induced seismicity have been recorded and proven in Oklahoma, where seismicity has been increased dramatically since 2010. In cases like Oklahoma, because the rate of earthquakes was very low before high-rate wastewater injection, it is relatively easy to determine the fundamental causes of induced seismicity. Therefore, due to the low background stress in intraplate regions, the human triggering signatures are clearly identifiable. However, in tectonically-active areas, it is more challenging to distinfuish between natural and triggered seismicity. In tectonically active regions in the US, such as California, although oil and gas extraction has taken place for many decades, only a handful of studies have been published with a focus on these areas (Hough et al. 2017). Big-data and statistical approaches will be crucial in separating natural causes from potential human triggering factors in these areas. In regions from Oklahoma, Hincks et al. developed a Bayesian network to for quantitative evaulation of correlations between well operational parameters, geological formation, and seismicity. The injection depth relative to crystalline basement was found to be the most significant parameter for seismic moment release (Hincks et al. 2018). Also, Wozniakowska and Eaton performed a machine learning estimation of the seismogenic activation potential for each well using logistic regression and also found that the injection depth influence greatly the seismogenic activation potential (Wozniakowska & Eaton 2020).

The correlations of induced earthquakes with fluid injection activities are well established, even in previous research some physics models are also developed for earthquake prediction. It is still not known why some areas are more susceptible to induced earthquakes. In this project, Several unique and rich datasets of industrial activities from regions in Oklahoma are used to statistically evaluate and retropectively forecast possible signatures of human-induced seismicity from existing high-quality earthquake in these regions. These datasets contains geological formation, injections, hydraulic fracturing activities and wells, which all have millions of items. They were selected subset from 2011 to 2018 and displayed on the local map of Oklahoma. Rough visual correlations between the earthquakes and these potential induced factors were discovered. Then, stepwise regression method was used to choose statistically significant features from all features. A logistic model was fitted before and after stepwise regression, which proved that the work of features selection indeed benefit to induced forecasting.

In previous studies for induced earthquakes in Oklahoma, forecasting works were almost always implemented using regression models. Compared to them, after selecting statistically significant features, this project originally designed and implemented a neural network for induced seismicity forecasting, which effectively improves the performance of model fitting and prediction.

2 Software Description

2.1 Visualization: Matplotlib & GeoPandas

GeoPandas can help researcher make working with geospatial data in Python easier. It extends the datatypes in Pandas, which allows users perform spatial operations on geometric types. Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Combining these two libraries, after setting the appropriate coordinate system, the data items can be mapped in spatial clearly by short code. Figure 1 shows all earthquakes recorded in the dataset.

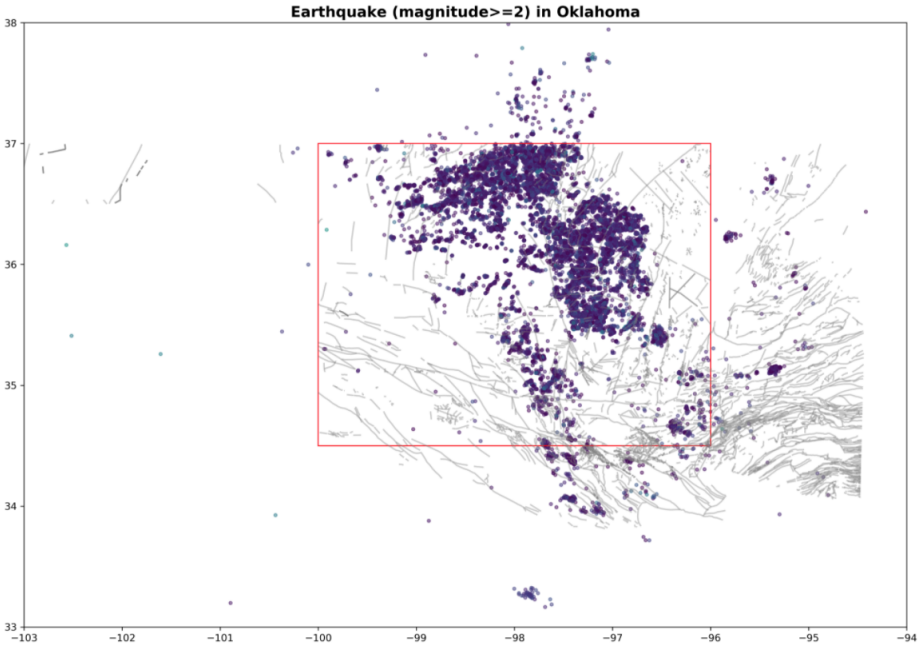


Figure 1: Earthquake Occurrences in Oklahoma.

The red rectangle is the interest area to study for this project. Here we only concentrate on the earthquakes which have a magnitude greater than 2.0.

2.2 Stepwise Regression

In any phenomenon, some factors will play a more important role in determining the outcome. Stepwise Regression is method which is widely used to find which factors are important and which are not. The factors that have a rather high p-value do not make a meaningful contribution to the accuracy of our predictions. Therefore, only the important factors are retained to ensure that the model make predictions based on the factors that will help it produce the most accurate results.

P-values are often used to see if the patterns they measured were statistically significant. If the p-value of a statistical test is small enough, we are able to reject the null hpytothesis of the test and the pattern measured is statistically significant. The most common threshold for p is 0.05. For the variable whose p-value is greater than 0.05, we can assume that it is not statistically significant as shown in Fig **??**. Here is step in stepwise logistic regression process. The 'injection_vol', 'HF_Base_Water_Volume', 'HF_Base_NoWater_Volume' both have a p-value than threshold 0.05 in logistic regression results, which are not statistically significant and should be eliminated in later process.

In the process of regression analysis, there usually exists the problem of high collinearity, which can cause the following influence:

- The model parameters are estimated inaccurately, and sometimes even the sign of the regression coefficients is exactly opposite to the actual situation, for example, the characteristic coefficients that should logically have positive coefficients are calculated to be negative.
- The independent variables that should be significant are not significant, and the independent variables that are not significant show the significance.

VIF.(required continue)

In this project, we performed a stepwise logistic regression that used a forward selection approach on

Algorithm 1 An algorithm with caption

Require: $n \geq 0$
Ensure: $y = x^n$
  $y \leftarrow 1$
  $X \leftarrow x$
  $N \leftarrow n$
  while $N \neq 0$ **do**
    if N is even **then**
      $X \leftarrow X \times X$
      $N \leftarrow \frac{N}{2}$ ▷ This is a comment
    else if N is odd **then**
      $y \leftarrow y \times X$
      $N \leftarrow N - 1$
    end if
  end while

all interest potential factors (Ryan 2020). All features were initially excluded and a feature is added in each step whilst keping current features satisfy p-value <0.05. This process continued untill all features kept are statistically significant. Meanwhile, in each step, we also remains the VIF <10 for current kept features even some new added feature required to be eliminated.

2.3 Neural Network Construction: Pytorch

Pytorch is an open source machine learning library for Python and is completely based on Torch library, used for applications such as computer vision and natural language processing. It is mostly developed by Facebook's AI Research lab (FAIR) (Patel 2018). PyTorch provides two features:

- Tensor computing (like Numpy) with strong acceleration via graphics processing units (GPU)
- Deep neural networks built on a type-based automatic differentiation system.

Pytorch provides a class called Tensor (torch.Tensor) to store and operate on homogeneous multidimensional rectangular arrays of numbers. They are tailored for datasets in machine learning. Tensors in Pytorch are similar to Numpy Arrays, but it supports to be operated on a CUDA-capable Nvidia GPU (Paszke et al. 2019).

To define a neural network in Pytorch, it is pretty convenient to create a class that inherits from nn.Module as below:

```
# Get cpu or gpu device for training.
device = "cuda" if torch.cuda.is_available() else "cpu"
print("Using {} device".format(device))

# Define model
class NeuralNetwork(nn.Module):
    def __init__(self):
        super(NeuralNetwork, self).__init__()
        self.flatten = nn.Flatten()
        self.linear_relu_stack = nn.Sequential(
            nn.Linear(4, 8),
            nn.ReLU(),
            nn.Linear(8, 8),
            nn.ReLU(),
            nn.Linear(8, 2),
            nn.ReLU()
        )
```

```
18
19          def forward(self, x):
20              x = self.flatten(x)
21              logits = self.linear_relu_stack(x)
22              return logits
23
24      model = NeuralNetwork().to(device)
```

User can define the layers of the network in the *__init__* function and indicates how data will pass through the network in the *forward* function. With Pytorch, user can move it to the GPU to accelate operations in the neural netwok if available. Figure 13 is the model created above trained on the playground online through Tensorflow. The neural network prediction model used in this project is implemented by this way.

2.4 Feature Attribution: Captum

Feature attribution is the technique to investgiate how much each feature in the model contributes to the prediction for given dataset fitting. With the increase in model complexity and the resulting lack of transparency, model interpretability methods have become increasingly important in machine learning field. In this project, we implemented a neural network model with the PyTorch library (PyTorch 2021) for improving the accuracy of prediction. Therefore, we use the Captum for feature attribution in neural network model, which is a model interpretability and understanding library for Pytorch. It includes general purpose implementations of integrated gradients, saliency maps, smoothgrad, vargrad and others for PyTorch models.

Captum provides advanced algorithms, including Intergarated gradientes used in this project, to provides users an simple way to understand which features are contributing to a models' output. Integrated gradients represents the integral of gradients with respect to inputs along the path from a given baseline to input. The integral can be approximated using a Riemann Sum or Gauss Legendre quadrature rule. Formally, it can be described as follows in Equation (1):

$$IntegratedGrads_i(x) ::= (x_i - x_i^{'}) \times \int_{\alpha=0}^{1} \frac{\partial F(x^{'} + \alpha \times (x - x^{'}))}{\partial x_i} d\alpha \tag{1}$$

Intergarated Gradients along the i-th dimension of input X. Alpha is the scaling coffecient (Sundararajan et al. 2017).

As shown in Fig 14, this is the result of feature importance analysis for the model trained in official tutorial of Captum. The official tutorial used an example model trained on the titanic survival data. It first trained a deep neural network on the data using Pytorch and used Captum to understand which of the features were most important and how the network reached its prediction. The features involved Age, Sibsp, Parch, Fare, Gender, Embark, Class and the target is a binary indicating whether passager is survived. After normalization and one-hot encodings, the neural network architecture was defined and trained. The neural network had a simple architecture using 2 hidden layers, the first with 12 hidden units and the second with 8 hidden units, each with Sigmoid non-linearity. The final layer performed a softmax operation and had 2 units, corresponding to the outputs of either survived or not survived. The accuracy of this model on dataset achieved to 81.6%. To perform feature attribution on this model, this example applied integrated Gradients, which is one of the Feature Attribution methods included in Captum. The feature attribution method code is shown in below.

```
1   ig = IntegratedGradients(model)
2   test_input_tensor.requires_grad_()
3   attr, delta = ig.attribute(test_input_tensor,target=1,
    return_convergence_delta=True)
```

Table 1: Average Feature Importances for each feature

age	sibsp	parch	fare	male	embark_C	embark_Q	embark_S	class_1	class_2	class_3
-0.454	-0.119	-0.056	0.175	-0.359	0.086	-0.001	-0.082	0.062	0.021	-0.159

```
    attr = attr.detach().numpy()

    # Helper method to print importances and visualize distribution
    def visualize_importances(feature_names, importances, title="Average Feature
    Importances", plot=True, axis_title="Features"):
        print(title)
        for i in range(len(feature_names)):
            print(feature_names[i], ": ", '%.3f'%(importances[i]))
        x_pos = (np.arange(len(feature_names)))
        if plot:
            plt.figure(figsize=(12,6))
            plt.bar(x_pos, importances, align='center')
            plt.xticks(x_pos, feature_names, wrap=True)
            plt.xlabel(axis_title)
            plt.title(title)
    visualize_importances(feature_names, np.mean(attr, axis=0))
```

The result of feature attribution was listed in Table 1 and shown in Fig 14. From the feature attribution information, we obtain some interesting insights regarding the importance of various features. We see that the strongest features appear to be age and being male, which are negatively correlated with survival. Embarking at Queenstown and the number of parents / children appear to be less important features generally.

3 Code Metadata

This project was built under macOS Catalina environment with Jupyter Notebook 6.0.3 and Python 3.7.6. The information of libraries and their usages are listed in Table 2.

In the Github repository, source code files (*.ipynb*) are stored in the src directory, including two main parts, which are feature extraction and prediction model. The data files (*.csv*) generated from feature extraction part are located at *data* directory. The models (*.pt*) generated from prediction part are located at the *model* directory.

4 Implementation

4.1 Data Processing and Spatial visualization

In the initial phase of project, we imported all the available datasets (earthquake data, geological data, hydraulic fracturing data, injection data, well data) and preprocessed on various datasets. The metadata of earthquake dataset was exported from *Oklahoma Geological Survey Earthquake Catalog Download Tool* (Walter et al. 2020). The injection dataset was exported from *Oklahoma Corporation Commission* (OklahomaCorporationCommission 2021). The preprocessing operations include missing values processing and data selecting by date. The date of injection dataset we imported was in (2011, 2020). However, we only used the data between 2011 and 2018. Because the data files >2019 supported by the *Oklahoma Corporation Commission* are still under review and corrections are being made as warranted, those files are currently incomplete and subject to change. Therefore, due to date limitation of injection data, We set the interest time in the period of 2011 to 2018 for all

Table 2: The dependencies of this project

Module	Version	Purpose
os	3.7.11	Search the data files (csv or xlsx) in the directory
numpy	1.18.1	Dataset process
pandas	1.0.1	
Shapely	1.7.1	Map the activities on map in spatial
geopandas	0.9.0	
matplotlib	3.1.3	Plot the figure
seaborn	0.10.0	
scipy	1.6.2	Load and process the geological formation data and plot the figure
xarray	0.18.2	
netCDF4	1.5.7	
statsmodels	0.12.2	Used in stepwise regression
torch	1.9.0	Neural network model implementation and training, and visualization of intermediate information
scikit-learn	0.22.1	
livelossplot	0.5.4	
captum	0.4.0	Perform the feature attribution

datasets. For the interest area, we set it to the area whose longitude ranges in (-99.5W, -96W) and latitiude ranges in (35.0N, 37.0N), which is a available spatial scope provied by all datasets. Then, we visualized all the selection data by *matplotlib* and *geopandas* module. The geological formation (depth to basement) is shown in Fig 6, and the wells along with various activities (seismicity, injection, hydraulic fracturing) are shown in Fig 5. The core of visualization code for all dataset is similar and is shown in the below:

```
earthquake_geometry = [Point(xy) for xy in zip(earthquake_df['longitude'], earthquake_df['latitude'])]
earthquake_geo_df = gpd.GeoDataFrame(earthquake_df, crs = "EPSG:4326", geometry = earthquake_geometry)

fig, ax = plt.subplots(figsize=(20, 20), dpi=300)
oklahoma_map.plot(ax=ax, alpha=0.4,color='grey')
earthquake_geo_df.plot(column='magnitude',ax=ax, alpha=0.5, legend=True, markersize=10)
plt.title('Earthquake (magnitude>=2) in Oklahoma', fontsize=15,fontweight='bold')
plt.xlim(-99.5, -96.0)
plt.ylim(35.0, 37.0)
plt.show()
```

The geodetic coordinate system we used for mapping in this project is 'EPSG:4326' (Howard et al. 2007), which is the most popular coordinate system.

4.2 Spatial Griding and Features Extraction

For the interest area (longitude ranges in (-99.5W, -96W) and latitiude ranges in (35.0N, 37.0N)), we divided it into 40×40 grids. Then we counted the values of all features and target for each grid by the various data from Section 4.1:

- *injection_vol_sum [BPD]*: The sum of injection volume in this grid.
- *injection_psi_sum [PSI]*: The sum of injection pressure in this grid.
- *injection_depth_avg [m]*: The mean of injection depth in this grid.

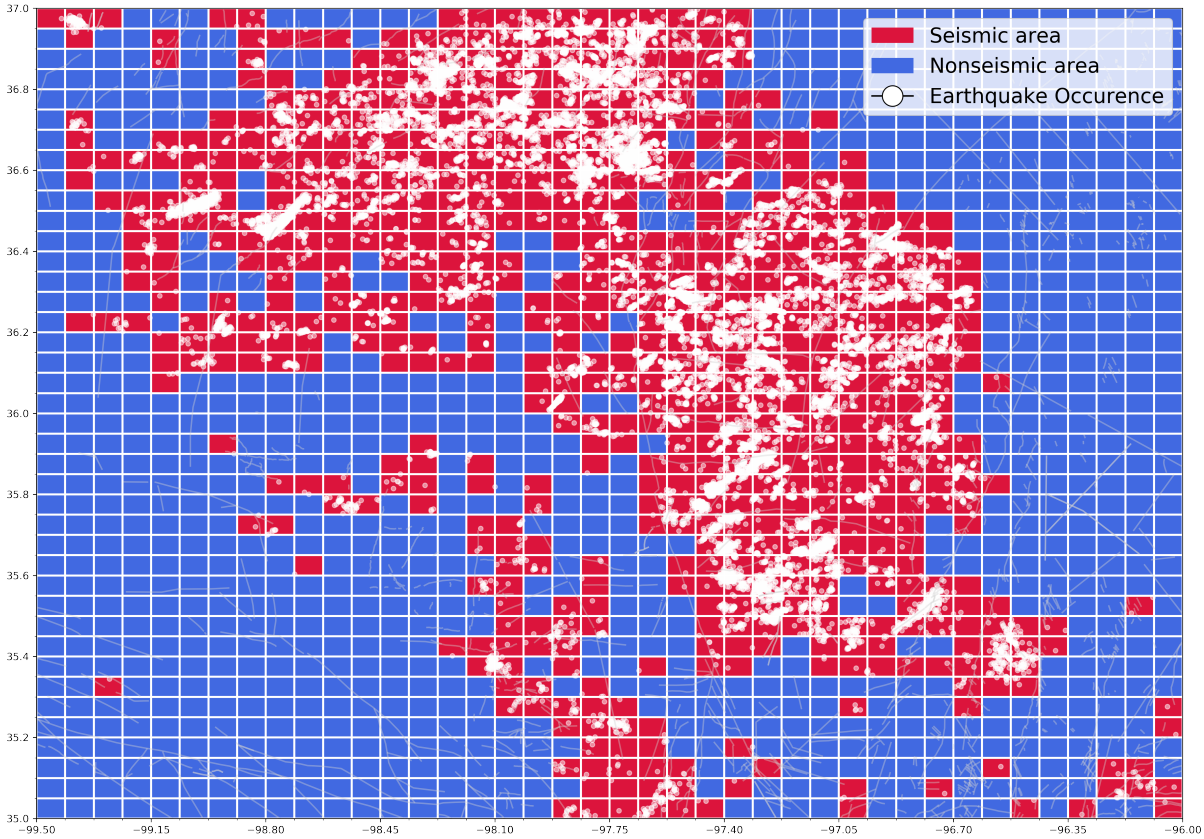


Figure 2: Target variable of earthquake occurrence within the interest area in Oklahoma.

- *injection_under_basement_number [entry]*: The number of injections whose depth are deeper than the basement in this gird.
- *active_well_number [entry]*: The number of working wells in this grid.
- *well_depth_avg [m]*: The mean of well depth in this grid.
- *well_under_basement_number [entry]*: The number of working wells whose depth are deeper than the basement in this grid.
- *depth_to_basement_avg [m]*: The mean of the depth to the basement in this grid.
- *hf_number [entry]*: The number of hydraulic fracturing activities in this grid.
- *hf_base_water_volume_sum [BPD]*: The sum of volume by hydraulic fracturing with water
- *hf_base_nowater_volume_sum [BPD]*: The sum of volume by hydraulic fracturing with nowater
- *earthquake_occurrence [0 or 1]*: If earthquake occurred in this grid. 0: not occured; 1: occurred.

The data files were generated in *.csv* and saved in *data* directory.

After this statistical process, we obtained a dataset in terms of spatial distribution, each of which represents a subarea of our interest area and contains the values of features. The target variable was based on seismicity, which represents the earthquake occurrence in spatial distribution, we assigned each grid a value of *one* if there was at least one (magnitude>2) earthquake occurred during 2011 to 2018. Otherwise, if no earthquake occurred in this grid, we assigned it a value of *zero*. The result grid of the earthquake occurrence (target) from 2011 to 2018 is shown in Figure 2. The white lines show the grid, with a spacing in the x direction of 8 km and the y direction of 5.5 km. The white points denotes the earthquake occurrences. The red region denotes it has seismicity in this region

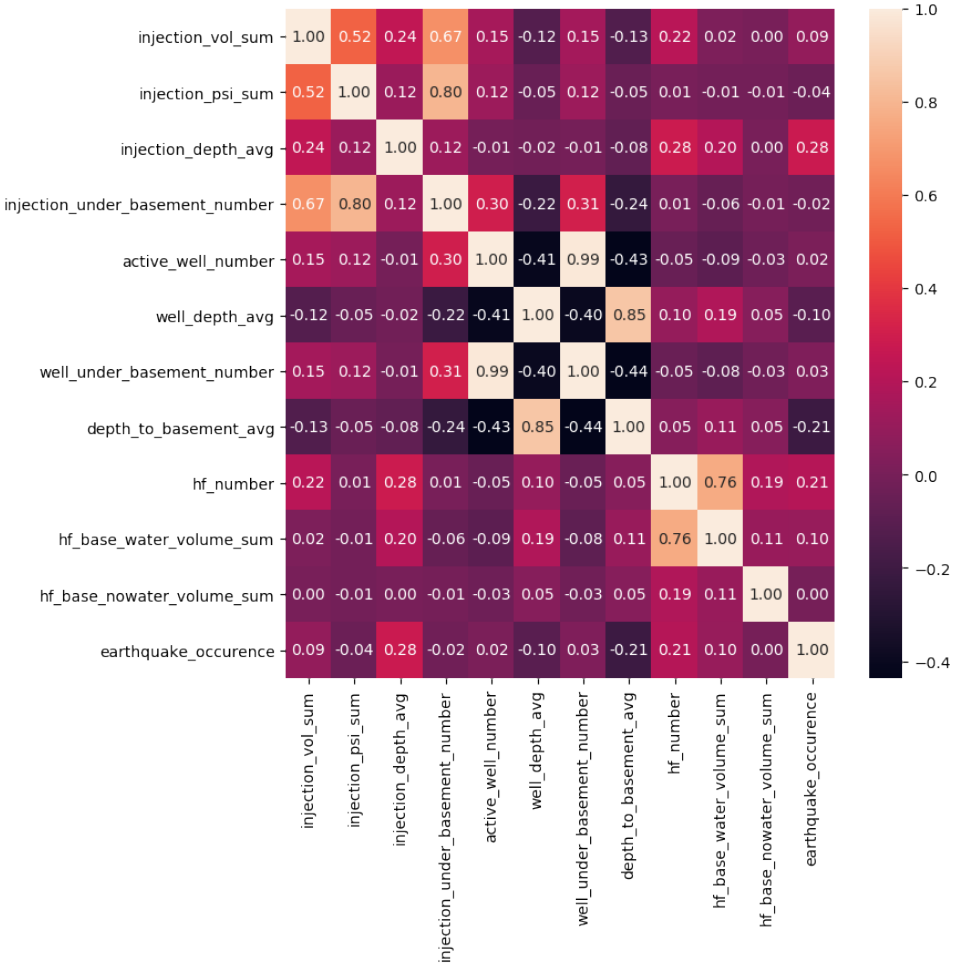


Figure 3: Correlation Heatmap.

and blue region denotes it does not have seismicity in this region. In our target grid, 42.5% of the 1600 gird points are assigned the value '1'.

4.3 Stepwise Feature Selection Approach

As some activities are strongly overlapped in spatial distribution, they may have a high degree of multicollinearity. We calculated the feature similarity on all fearures based on Peason correlation coffecients and generated a heatmap as Figure 3. In general, We consider a high degree of collinearity between two features with a coefficient greater than 0.9, like 'active_well_number' and 'well_under_basement_number'. Therefore, in the process of selecting statistically significant features, we should also account for the high multicollinearity to prevent the influence in Section 2.2.

To select 'best' features and deal with high multicollinearity problem, we use a forward stepwise approach introduced in Section 2.2. We iteratively add a feature whilst ensuring the p-value of all current added feature remains below 0.05. This means that each feature we kept is statistically significant at the 95% confidence level. Meanwhile, in each iteration step, we also deal with the high multicollinearity by checking the Variance Inflation Factor (VIF) for current kept features. We iteratively eliminated the feature whose VIf is greater than 10 after adding a new feature (generally we think think there exists a high multicollinearity when VIF of some feature $>$10).

After this process, we eliminated insignificant features and also solved the problem of high multi-

collinearity between variables. The result of features selecting with stepwise approach is shown in Figure 7.

4.4 Logistic Regression Model

With the selected features in Section 4.3, we constructed a multiple logistic regression model in this part. which used the features selected as independent variables (hf_base_water_volume_sum, hf_number, depth_to_basement_avg, well_depth_avd, injection_under_basement_number, injection_depth_avg, injection_vol_sum) and earthquake occurrence as the target variable. We standardise and normalise all input features by z-score normalization (Patro & Sahu 2015) to reduce data skewness and bring all features on the same scale. The normalization of features allows us to interpret the relative difference in the model coffecients.

We splitted the dataset into train dataset and test dataset with the ratio of 9:1 , which have 1440 and 160 samples (Each sample is associated with one gird point in the interest area). In this build part, we use *sklearn.linear_model.LogisticRegression* to construct and fit our logistic model, which achieved the accuracy of 72.5% on the test dataset. The core part of code is shown in below:

```
1    X_train, X_test, y_train, y_test = train_test_split(X,Y, train_size=0.9,
     random_state=42)
2    model = LogisticRegression()
3    model.fit(X_train, y_train)
4    model.score(X_test, y_test)
```

After fitting the model and testing, we check the coffecients for all input features by *model.coef_[0]* and plot them in Figure 10. With the trained logistic model, we used the whole dataset to predict and compared output labels with true labels, which is shown in Figure 9.

4.5 Neural Network Model

Due to the relatived low accuracy of prediction by logistic regression model in Section 4.3, we used the neural network model to improve the accuracy of prediction, which can be used in practical. Because logistic regression is so simple that it is difficult to model for nonlinear data or data with polunomial correlation, it is hard to deal with the complicated problem. After all, in practice we don't know the actual relationship between earthquakes and these potential factors. We don't even know if the relationship between input features and the target variable (earthquake occurrence) is linear or non-linear. Therefore, neural network algorithm is suitable for the prediction of this kind of complicated problem. Neural network model have the ability to capture information contained in large amounts of data and build incredibly complex models. With the abilities of self-learning and high-speed for optimal solution, it easily achieves a higher performance compared with regression method.

4.5.1 Architecture

In this project, we use a neural network model which has its input layer with 7 neurons, two hidden layers of 8 neurons per layer and an outout layer with 2 neurons as shown in Figure 4. In general, 1-5 hidden layers will serve for most problems and using the same number of neurons for all hidden layers will suffice. Therefore in this project, we choose to use two hidden layers and each layer has 8 neurons. The input features used are also based on the feature selecting result in Section 2.2, so we had seven input features for our training model. It's taken for granted that we set the number of neurons to 7, which presents the each input feature. For the earthquake prediction problem, which is a binary

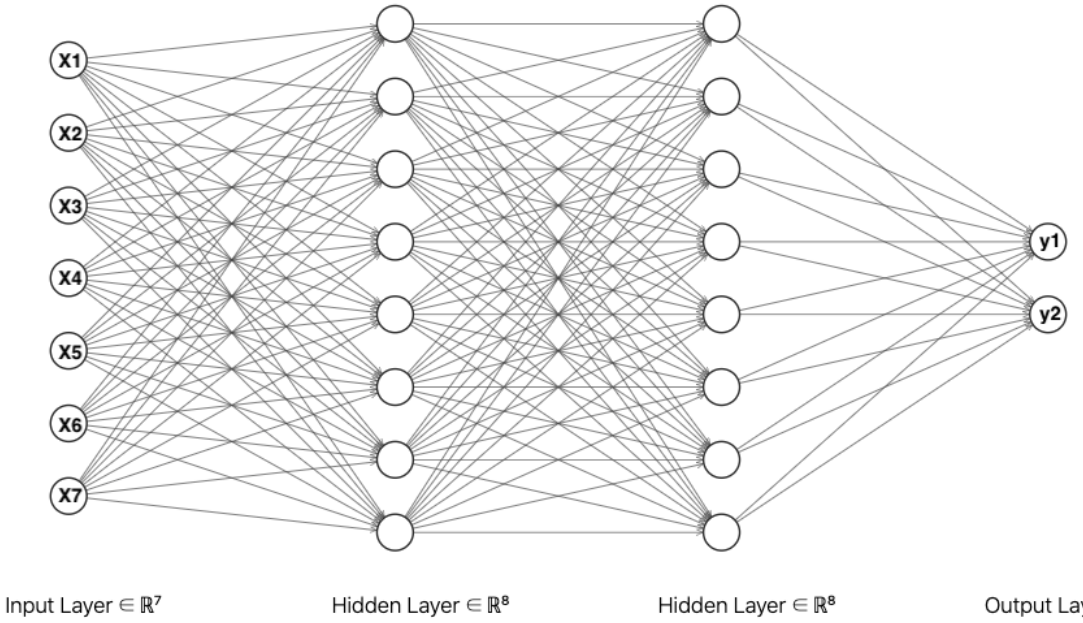


Figure 4: Neural Network Model Architecture.

classification, so we use two neurons in output layer which one output neuron per positive class. The output represents the probability of the positive class (earthquake or no earthquake). We use the softmax activation function on the output layer to ensure the final probabilites sum to 1.

4.5.2 Configuration

For this prediction problem is classification problem, we use the cross-entropy (De Boer et al. 2005) as the loss function which serves well. By lots of training attempts, we final used a set of hyperparameters which make the model performance in a well level. Generally the best preformance is obtained by mini-batch sizes between 2 and 32, in training attempts, we found that the model performed best when the batch size was set to 16. For the epoch size, we started from a large number of epochs and use early stopping to halt training when performance stops improving. The selection of the learning rate is very important, we start with a very low value (10^{-6}) and slowly multiply it by a constant unitl it reaches a pretty high value (here the max value is 10). We measured the performance (vs the log of learning rate) in *Weights and Biases* to determine which rate served well for our earthquake prediction problem. We found 10^{-2} is best and we then retrrin our model using this optimal learning rate. For the optimizer used in our model, we decide use Adam optimizer, which tends to be quite forgiving to a bad learning late and other non-optimal hyperparameters compared with most common used SGD optimizer.

The configuration which is called hyperparameters in machine learning is summarized below:

- *learning rate*: 10^{-2}
- *batch size*: 16
- *epoch*: 200
- *optimizer*: Adam
- *loss function*: cross entropy

4.5.3 Training

We initially divided the dataset into training dataset and testing dataset and normalised them as we done before fitting logistic regression model in 4.4. We use the Pytorch to conveniently generate the neural network model corresponding to architecture in Figure 4 by the code below:

```
    class Net(nn.Module):
    def __init__(self,n_input, n_output):
        super(Net,self).__init__()
        self.linear1 = nn.Linear(n_input, 8)
        self.sigmoid1 = nn.Sigmoid()
        self.linear2 = nn.Linear(8, 8)
        self.sigmoid2 = nn.Sigmoid()
        self.linear3 = nn.Linear(8, n_output)

    def forward(self, x):
        lin1_out = self.linear1(x)
        sigmoid_out1 = self.sigmoid1(lin1_out)
        sigmoid_out2 = self.sigmoid2(self.linear2(sigmoid_out1))
        return self.linear3(sigmoid_out2)
```

As same with the Figure 4 shows, we implemented three linear transformations, which represents input layer to first hidden layer, first hidden layer to second hidden layer and second hidden layer to output layer. Each transformation applies a linear transformation to the incoming data: $y = xA^T + b$. x represents the vector of neuron values before transformation and y represents a vector of neuron values after transformation. A represents the weights multiplied in linear transformation (similar to coefficients in regression) and b represents the vector of bias. Our input layer corresponding to input features so we set the *n_input* variable in code to 7. Our output layer corresponding to output target class so we set the *n_output* variable in code to 2. The *__init__* function defines the architecture of neural network. The *forward* function in the code represents the process of propagation.

we trained the model by 200 epochs in total. For each epoch, we trained by the whole training dataset and test the accuracy of current model in testing dataset. We used *torch.utils.data.DataLoader* to load the training dataset and testing dataset. Then, everytime we load a batch (16 samples) of training data and input into model for prapagation, then the outputs were compared with the labels. According to loss of current batch, we updated the weights in model by *optimizer.step()*. After training by one epoch, we calculate the accuracy of model in whole training dataset and whole testing dataset and updated the loss value in log plot. The training performance figure of our model is shown in Figure 11. By use the network model trained, we achieved a much better accuracy of 80.6% on testing dataset compared with logistic regression model in Section 4.4. Similarly, we also used the whole dataset to predict by our trained neural network model and compared output labels with true labels, which is shown in Figure 12.

5 Results and Discussion

This section will analogize the results from Section 4.

5.1 Visualiztion

Through these activities distribution in Fig 5 and Fig 6, we can found a rough spatial correlation between some of them. We firstly compared the depth to basement with earthquake activities, we found the lower left region of interest area has a thicker softer sedimentary rocks than upper right region (In the Figure 6 it showes lower left region has larger depth value and upper right region

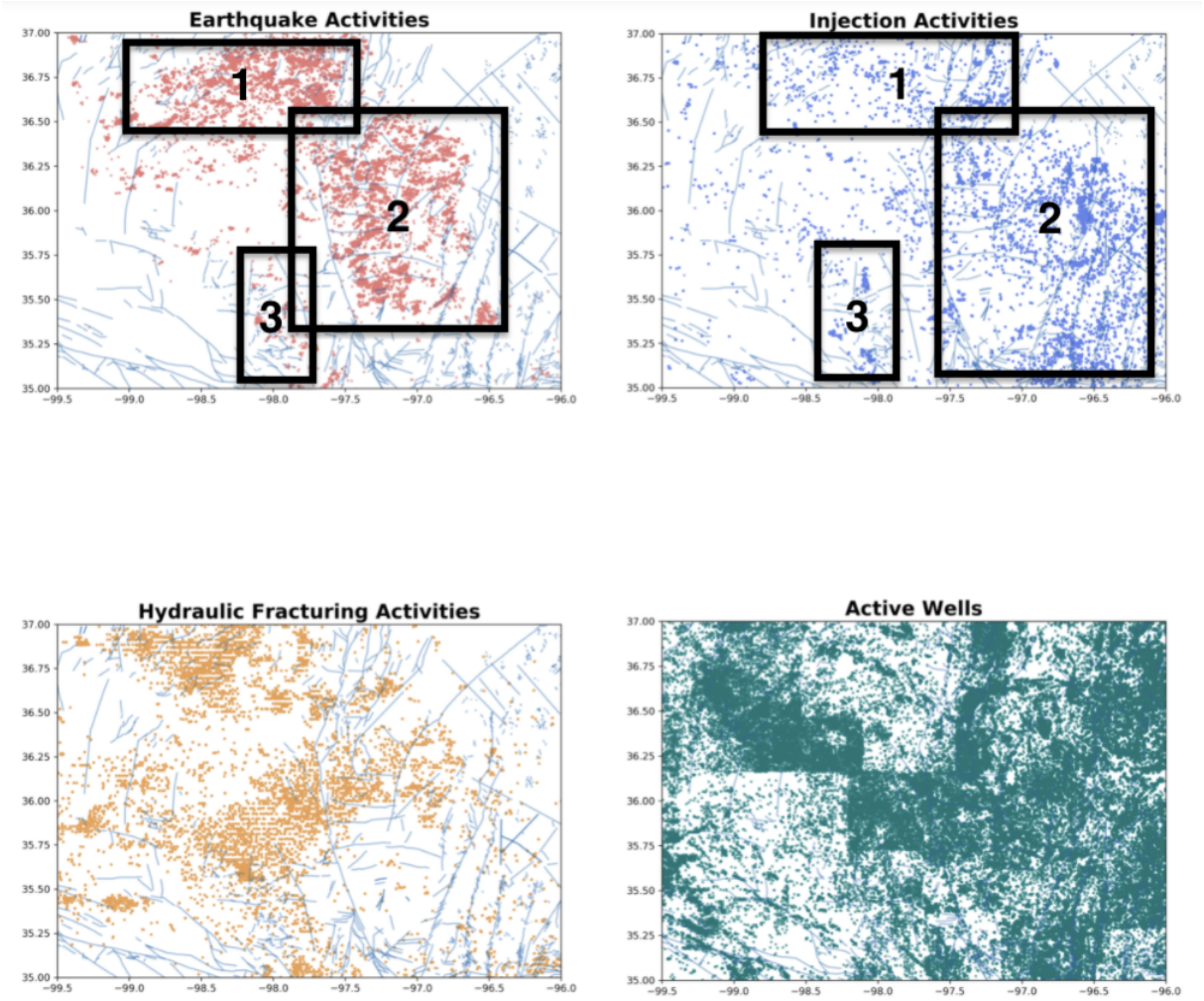


Figure 5: Activities during 2011 to 2018 in interest area from Oklahoma.

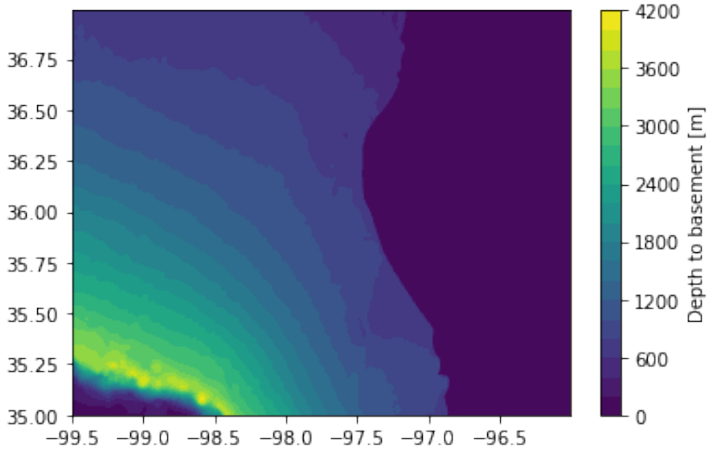


Figure 6: Depth to basement in Oklahoma.

```
Feature selected:  injection_vol_sum
Optimization terminated successfully.
         Current function value: 0.590683
         Iterations 6
                           Logit Regression Results
==============================================================================
Dep. Variable:                  class   No. Observations:                 1600
Model:                          Logit   Df Residuals:                     1593
Method:                           MLE   Df Model:                            6
Date:                Sat, 07 Aug 2021   Pseudo R-squ.:                  0.1337
Time:                        16:10:17   Log-Likelihood:                -945.09
converged:                       True   LL-Null:                       -1091.0
Covariance Type:            nonrobust   LLR p-value:                 4.787e-60
=================================================================================================
                                    coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------------------
hf_base_water_volume_sum         -0.5612      0.114     -4.908      0.000      -0.785      -0.337
hf_number                         0.9908      0.152      6.517      0.000       0.693       1.289
depth_to_basement_avg            -1.4814      0.177     -8.372      0.000      -1.828      -1.135
well_depth_avg                    0.9690      0.166      5.827      0.000       0.643       1.295
injection_under_basement_number  -0.6736      0.119     -5.660      0.000      -0.907      -0.440
injection_depth_avg               0.4854      0.060      8.037      0.000       0.367       0.604
injection_vol_sum                 0.2731      0.105      2.612      0.009       0.068       0.478
=================================================================================================
                           feature       VIF
0         hf_base_water_volume_sum  2.631406
1                        hf_number  2.814120
2            depth_to_basement_avg  3.822348
3                    well_depth_avg  3.851949
4  injection_under_basement_number  1.999341
5               injection_depth_avg  1.143023
6                 injection_vol_sum  2.161789
```

Figure 7: Feature Selection Result with Stepwise Approach.

almost has 0 value). It is can be inferred that the induced seismicity may be related to depth to basement. Besides, the injection activities are similar to earthquakes activities in spatial distribution. As the Figure 5 shows, the three black rectangular regions (marked by 1 to 3) represents the activities dense area, which are highly similar in spatial distribution, even in small area as rectangular area 3. By the high similarity, we can concluded that the induced seismicity have high correlation with injection activities. For the hydraulic fracturing activities and well spatial data, we can not found some relation beteen induced earthquake and them and we need to explore and interpret through the model generated.

5.2 Stepwise Approach

The Process of features selection with stepwise approach is shown in below block and the feature result along with their p-value and VIF are shown in Figure 7.

- Iteration 1
  Feature selected: hf_base_nowater_volume_sum
  P-value >0.05 check: hf_base_nowater_volume_sum
  Feature removed: hf_base_nowater_volume_sum
  VIF>10 check: None
  Feature removed: None
- Iteration 2
  Feature selected: hf_base_water_volume_sum
  P-value >0.05 check: None
  VIF>10 check: None
  Feature removed: None

- Iteration 3
  Feature selected: hf_number
  P-value >0.05 check: None
  VIF>10 check: None
  Feature removed: None

- Iteration 4
  Feature selected: depth_to_basement_avg
  P-value >0.05 check: None
  VIF>10 check: None
  Feature removed: None

- Iteration 5
  Feature selected: well_under_basement_number
  P-value >0.05 check: None
  VIF>10 check: None
  Feature removed: None

- Iteration 5
  Feature selected: well_depth_avg
  P-value >0.05 check: None
  VIF>10 check: None
  Feature removed: None

- Iteration 6
  Feature selected: active_well_number
  P-value >0.05 check: well_under_basement_number(0.477), active_well_number(0.772)
  Feature removed: active_well_number
  VIF>10 check: None
  Feature removed: None

- Iteration 7
  Feature selected: injection_under_basement_number
  P-value >0.05 check: well_under_basement_number(0.102)
  Feature removed: well_under_basement_number
  VIF>10 check: None
  Feature removed: None

- Iteration 8
  Feature selected: injection_depth_avg
  P-value >0.05 check: None
  Feature removed: None
  VIF>10 check: None
  Feature removed: None

- Iteration 9
  Feature selected: injection_psi_sum
  P-value >0.05 check: injection_psi_sum(0.969)

Table 3: Indicators of two modules

Model\Indicator	ACC	Precision	Recall	F1-score
Logistic Regression Model	0.705	0.705	0.530	0.604
Neural Network Model	0.79	0.734	0.792	0.762

Feature removed: injection_psi_sum
VIF>10 check: None
Feature removed: None

- Iteration 10
  Feature selected: injection_vol_sum
  P-value >0.05 check: None
  Feature removed: None
  VIF>10 check: None
  Feature removed: None

We can see all the selected features by stepwise approach meet the requirement of p-value <0.05 and VIF <10. This means we successfully eliminated the features which are not statistically significant and there is no high multicollinearity in all features selected.

5.3 Model Comparison

5.3.1 Logistic Regression Model

By the Section 4.4 and Section 4.5, we fitting the logistic regression model and training the configured neural network model on the training dataset, the prediction accuracy on testing dataset for these two models achieves at 72.18% and 77.5% respectively. This denotes that neural netwok model performs better than logistic regression model. Besides, we extraly use these models to test on the whole dataset, which contains all the grids point of interpret area. The Indicators which we usually used to assess for models in machine learning field are calculated and listed in Table 3.

The *ACC* refers to accuracy, which denotes the proportion of all correct predictions in the total predictions. We can see for all regions in interpret area, the prediction accuracy of NN model is 10 percent higher than LR model. The *ACC* is a simple indictor in model evaluation, it is an inituitive representation of whether the model is capable of making accurate predictions. However, it is easily affected by the imbalance of the samples. For example, there are a total of 100 samples and 90 of them are positive samples. If the model predicts all samples to positive sample, the accuracy can reaches up to 90%, but such a model is meaningless obviously. Fortunately, the samples in our dataset are relatively balanced, which have 680 positive samples and 920 negative samples in the ratio of 2:3 as shown in Figure 8. Therefore, accuracy can be a import indictor for model comparison and evaluation in this project. Through the *ACC* comparison for LR model and NN model in Table 3, we can concluded that our NN model have a better ability pf prediction than LR model.

In a classification problem, precision attempts to answer the question that what proportion of positive identifications was actually correct. The Recall attempts to answer the question that what proportion of actual positives was identified correctly.

The prediction accuracy achieved 70.75% by the comparison between seismicity in truth and prediction. Generally three dense regions distribution are correctlypredicted, but the precision of regional outline is pretty poor.

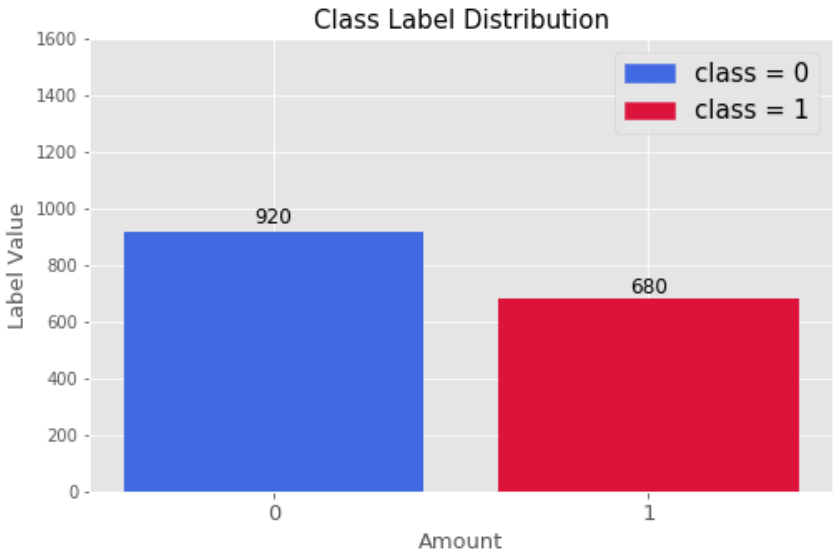


Figure 8: Class Label Distribution in whole dataset.

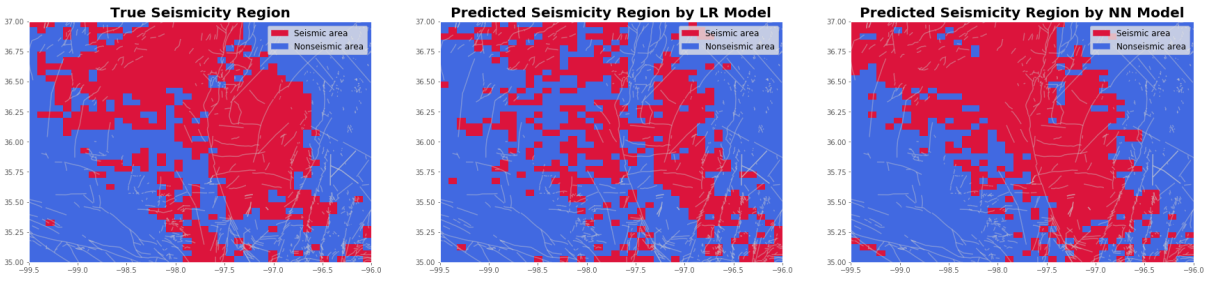


Figure 9: Seismicity Prediction by Logistic Regression Model and Neural Network Model.

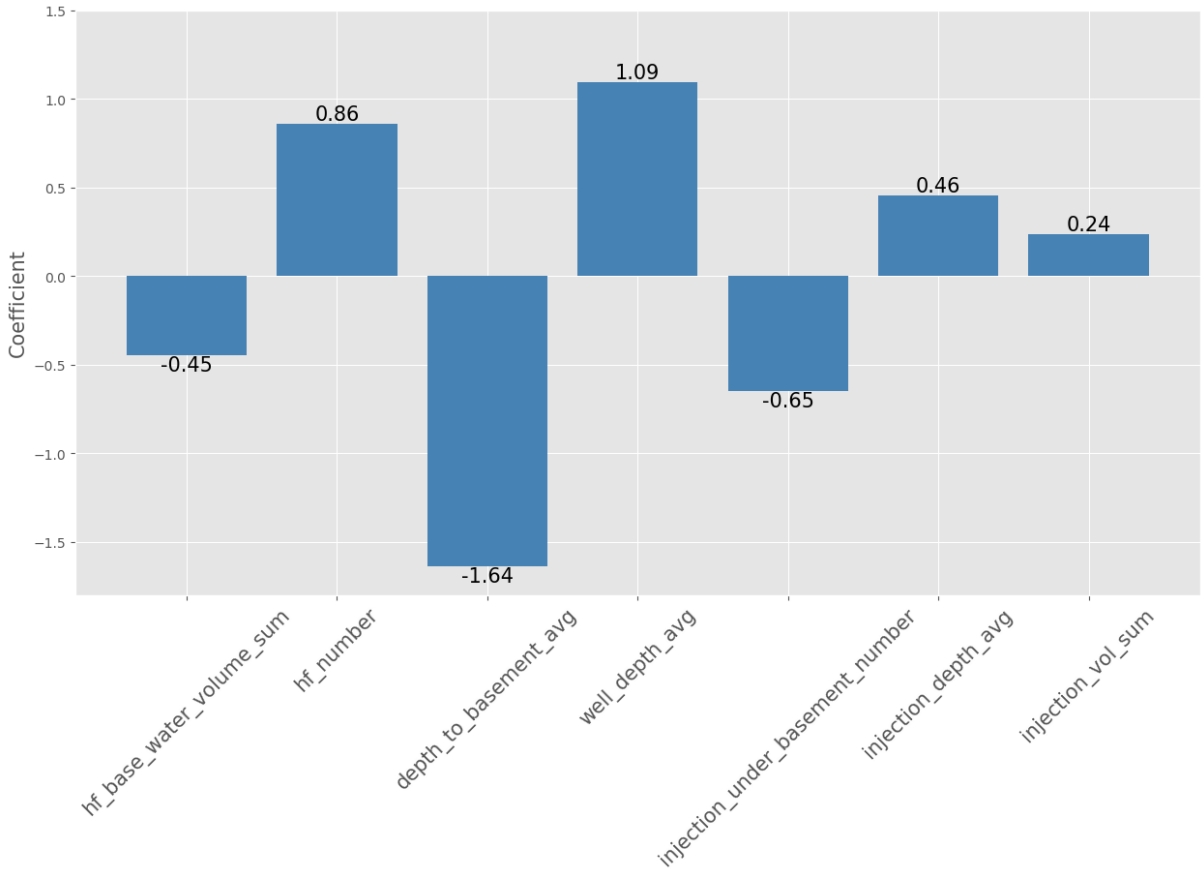
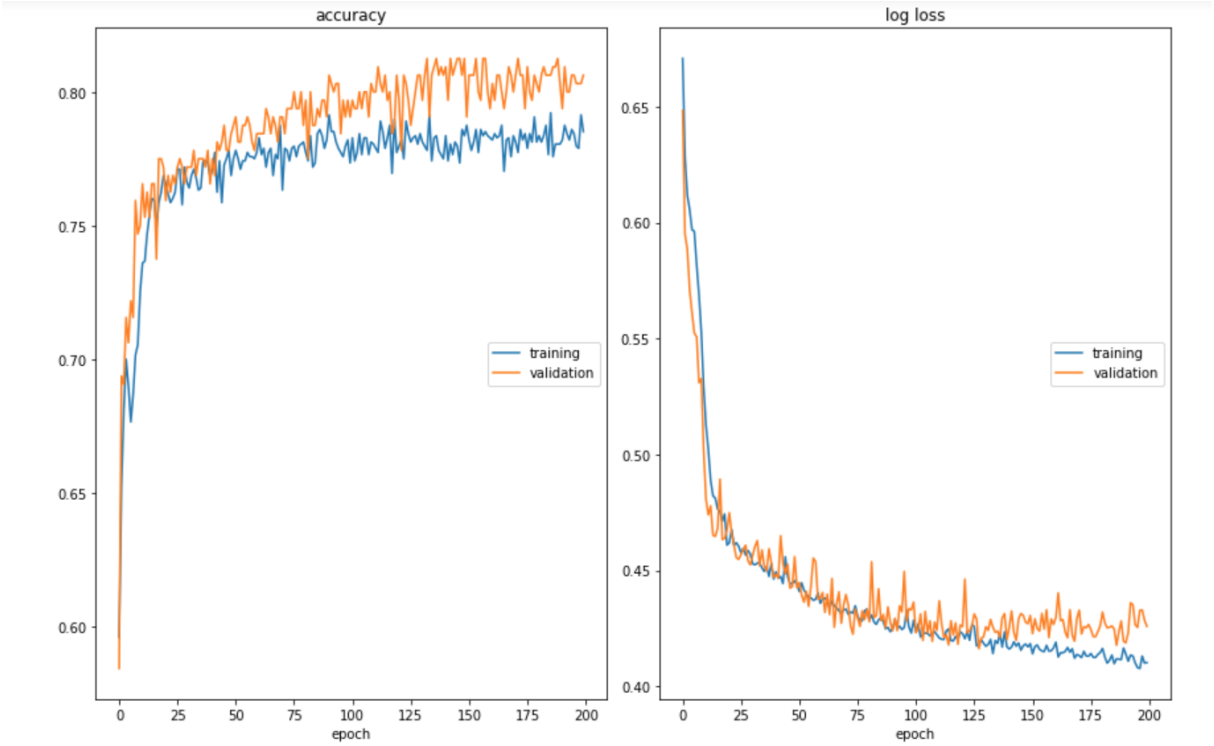


Figure 10: Coffecients for Features in Fitted Logistic Regression Model.




```
accuracy
        training                (min:    0.596, max:    0.792, cur:    0.785)
        validation              (min:    0.584, max:    0.812, cur:    0.806)
log loss
        training                (min:    0.408, max:    0.671, cur:    0.410)
        validation              (min:    0.416, max:    0.648, cur:    0.426)

Avg. Test Loss: 0.426  Avg. Test Accuracy: 0.806
```

Figure 11: Neural Network Model Training Process.

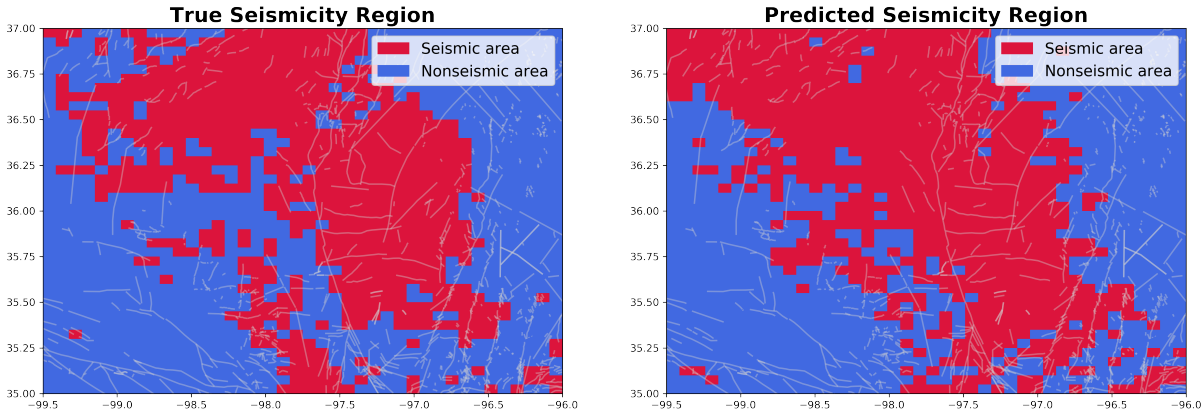


Figure 12: Seismicity Prediction with Trained Neural Network Model.

6 Conclusion and Future Work

References

De Boer, P.-T., Kroese, D. P., Mannor, S. & Rubinstein, R. Y. (2005), 'A tutorial on the cross-entropy method', *Annals of operations research* **134**(1), 19–67.

Ellsworth, W. L. (2013), 'Injection-induced earthquakes', *Science* **341**(6142).

Hincks, T., Aspinall, W., Cooke, R. & Gernon, T. (2018), 'Oklahoma's induced seismicity strongly linked to wastewater injection depth', *Science* **359**(6381), 1251–1255.

Hough, S., Tsai, V., Walker, R. & Aminzadeh, F. (2017), 'Was the mw 7.5 1952 kern county, california, earthquake induced (or triggered)?', *Journal of Seismology* **21**, 1613 – 1621.

Howard, B., Christopher, S., Dane, S. & Josh, L. (2007), 'epsg projection 4326 - wgs 84'.
URL: *https://spatialreference.org/ref/epsg/wgs-84/*

Norbeck, J. & Rubinstein, J. L. (2018), 'Hydromechanical earthquake nucleation model forecasts onset, peak, and falling rates of induced seismicity in oklahoma and kansas', *Geophysical Research Letters* **45**(7), 2963–2975.

OklahomaCorporationCommission (2021), 'Oil and gas data files'.
URL: *https://spatialreference.org/ref/epsg/wgs-84/*

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L. et al. (2019), 'Pytorch: An imperative style, high-performance deep learning library', *Advances in neural information processing systems* **32**, 8026–8037.

Patel, M. (2018), 'When two trends fuse: Pytorch and recommender systems'.

Patro, S. & Sahu, K. K. (2015), 'Normalization: A preprocessing stage', *arXiv preprint arXiv:1503.06462* .

PyTorch (2021), 'Stepwise regression tutorial in python'.
URL: *https://pytorch.org/docs/stable/index.html*

Ryan, K. (2020), 'Stepwise regression tutorial in python'.
URL: *https://towardsdatascience.com/stepwise-regression-tutorial-in-python-ebf7c782c922*

Sundararajan, M., Taly, A. & Yan, Q. (2017), Axiomatic attribution for deep networks, *in* 'International Conference on Machine Learning', PMLR, pp. 3319–3328.

Walter, J. I., Ogwari, P., Thiel, A., Ferrer, F., Woelfel, I., Chang, J. C., Darold, A. P. & Holland, A. A. (2020), 'The oklahoma geological survey statewide seismic network', *Seismological Research Letters* **91**(2A), 611–621.

Wozniakowska, P. & Eaton, D. W. (2020), 'Machine learning-based analysis of geological susceptibility to induced seismicity in the montney formation, canada', *Geophysical Research Letters* **47**(22), e2020GL089651.

7 Appendix

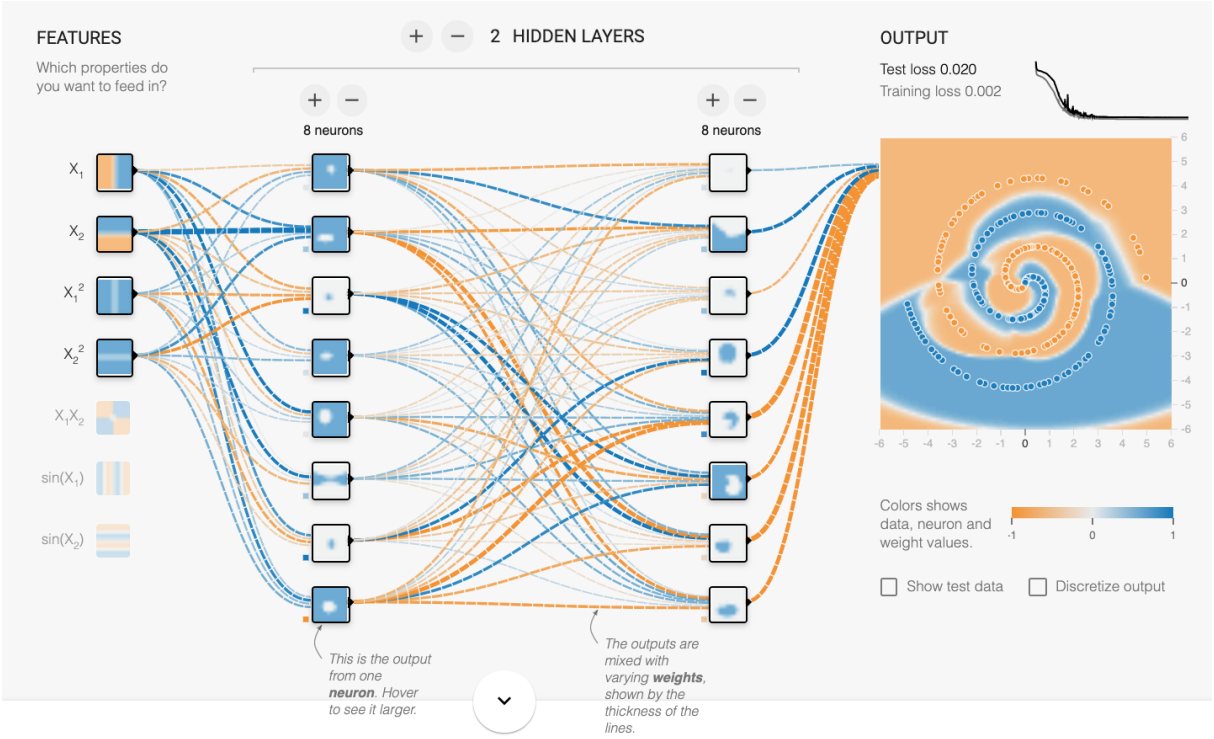


Figure 13: The model created from above code simulated through playground.tensorflow.org

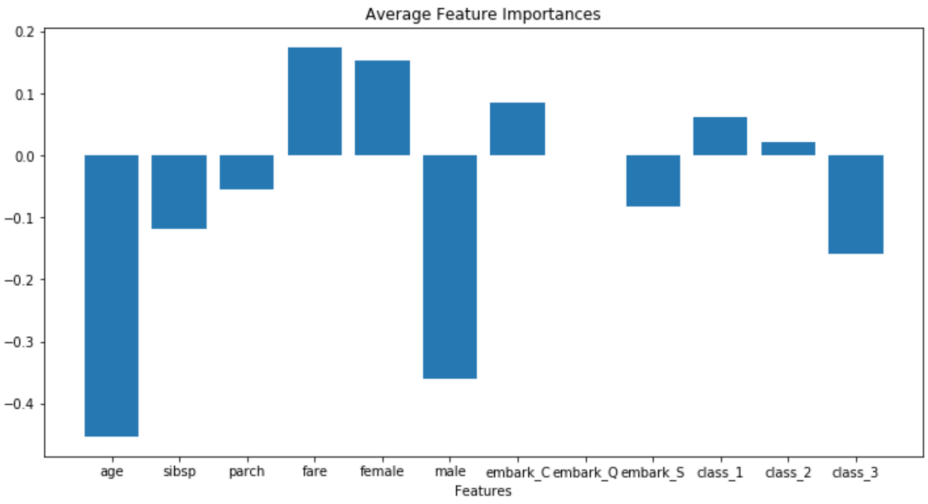


Figure 14: Feature Importance Example in Captum Tutorial.