



Earthquake Pattern Recognition

Bachelor, Computing

Department of Computing, Mathematics and Physics

Faculty of Engineering and Science

Submission date:
04.06.2018

Number of words:
6189

Joakim Vindenes Trond Fossdal Århus
Sigbjørn Myhre Torstein Leversund

Report title:

EN: Earthquake Pattern Recognition
NO: Mønstergjenkjenning i Seismiske Data

Date:

04.06.2018

Authors:

Joakim Vindenes, Trond Fossdal Århus
Sigbjørn Myhre, Torstein Leversund

Number of pages:

17

Study:

Bachelor, Computing

Number of appendix pages:

5

Supervisor:

Richard Kjepso

Classification:

Open

Remarks:

None

Assigning company:

SeisAnalysis AS

Company contact:

Mohammad Raeesi

Keywords:

Machine Learning

Seismology

Predictions

Contents

1	Introduction	1
1.1	Aims	1
1.2	Context	1
1.3	Project Owner	1
2	Machine Learning	2
2.1	Neural Networks	2
2.2	Backpropagation	2
2.3	Recurrent Neural Networks	4
2.3.1	Long Short-Term Memory	4
2.4	Overfitting	5
3	Earthquakes	7
3.1	Richter Scale and Moment Magnitude	7
3.2	Focal Mechanism and the Moment Tensor	7
4	Previous work	8
4.1	Neural Networks to predict Earthquakes in Chile	8
4.2	Probabilistic Neural Network for Earthquake Magnitude Prediction	8
4.3	Medium-large Earthquake Magnitude Prediction in Tokyo with Artificial Neural Networks	8
5	Project Design	9
5.1	Data Source	9
5.2	Input Data Model	9
5.3	Choice of Neural Network architecture	10
5.4	Tools	10
6	Detailed Design	11
6.1	Setting up the environment	11
6.2	Experimenting with data	11
6.3	Testing the models	11
7	Evaluation	12
8	Results	13
9	Discussion	15
10	Conclusions	16
10.1	Tools	16
10.2	Model	16
	References	17

1 Introduction

On average, earthquakes have claimed 50 000 lives¹ each year for the last 15 years, and caused considerable economic loss. It has been estimated that in the US alone, the cost caused by earthquakes each year is in excess of 4.4 Billion USD [1]. Early detection systems could potentially mitigate some of the consequences of earthquakes by providing ample warning to local population centers. Such endeavors has not yet borne fruits; finding patterns in earthquake measurements has proven difficult.²

The availability of cheap computing power and further theoretical development in the field has triggered renewed interest in Machine Learning. Neural Network algorithms have in the recent years showed a remarkable propensity to extract patterns from complex data sets. These algorithms have even surpassed humans in some narrow disciplines, such as medical imaging analysis and medical diagnostics [2]. This raises some interesting questions. In the near future, will there be systems capable of detecting and extracting relevant patterns from earthquake data? And subsequently can these systems be building blocks for detection systems that will give people information that is accurate enough to be useful? This report documents an attempt to apply machine learning tools in earthquake detection.

1.1 Aims

- Create a successful machine learning model for earthquake prediction
 - Using the free and open source tools Python, Keras and TensorFlow.
 - Based on information about time, magnitude and mechanism of previous earthquakes.

1.2 Context

These days we have large amounts of seismic data openly available and the computers are powerful enough to automatically classify these data fairly quickly. A lot of free and open sources machine learning tools are available to work with these data.

This work is done for SeisAnalysis as a basis for further work towards recognizing patterns in earthquakes.

1.3 Project Owner



SeisAnalysis AS was established in 2017 by seismologist Mohammad Raeesi and Torstein Leversund. SeisAnalysis focuses on analysis of subjects related to seismology, with special emphasis on earthquake seismology. The stated goal of the company is to develop early warning systems for earthquakes in order to mitigate the potentially devastating effects of large seismic events. SeisAnalysis sees modern machine learning algorithms as a possible path towards this goal.

¹<https://earthquake.usgs.gov/earthquakes/browse/stats.php>

²https://www.usgs.gov/faqs/can-you-predict-earthquakes?qt-news_science_products=0#qt-news_science_products

2 Machine Learning

2.1 Neural Networks

The term «machine learning» lumps together a wide variety of techniques and algorithms which have in common that they are strategies for learning to perform pattern recognition and classification based upon empirical data. «Neural networks» is a sub discipline of machine learning, where the core architecture is loosely based on how it is believed that neurons in animal brains interact with each other. Artificial neurons are mathematical models which mimics the biological neurons. A neuron has an output which is based on a sum of each of its inputs. Each input has a corresponding weight which is multiplied with the value from the input. Following, the output value is calculated with an activation function, where the weighted input sum is the input.

Typically, networks are trained to classify and/or make predictions by receiving an input and producing an output. The output is then compared to the desired output, often referred to as the «ground truth». The network is rewarded or punished based upon the difference between actual output and desired output. A common learning strategy for neural networks is to modify the connection strength between neurons in a process referred to as backpropagation, often described as the workhorse of modern artificial neural networks.

The first neural networks consisted of neurons which had a step-function as its output, yielding either 0 or 1 based upon whether or not the sum of its inputs exceeded a threshold value. The step-function will however only be able to approximate linear regression. In order to introduce non-linearity, the step-function was swapped for non-linear functions such as sigmoid and tanh (and later ReLU).

2.2 Backpropagation

The first step of the backpropagation algorithm is to determine the difference between the network output and the ground truth. Let a_o be the output of the output layer neuron, and let y be the desired output (ground truth). The error for this output neuron is the square of the difference between the desired and actual output.

$$E = (a_o - y)^2 \quad (1)$$

The output of the neuron, a_o , is determined by adding all the neuron inputs together, denoted by z , and passing that sum as an argument into the activation function of the neuron, henceforth denoted by

$$a_o = \sigma(z) \quad (2)$$

The subscript of a indicates the current layer, o referring to the output layer and $o - 1$ referring to the hidden layer connected to the output layer. z is the sum of the output of each neuron in the previous layer multiplied with the weight of the synapse connecting the output neuron to the neuron in the previous layer.

$$z = \sum_{i=1}^j w_i a_{i, o-1} \quad (3)$$

The objective of the backpropagation algorithm is to modify the weights of the synapses as to minimise the error E . Changing the weights directly changes the input z , which in turn cause a change in a_o which is used to calculate the error.

$$\frac{\partial E}{\partial w} = \frac{\partial z}{\partial w} \frac{\partial a_o}{\partial z} \frac{\partial E}{\partial a_o} \quad (4)$$

Each of the terms in equation (4) can be evaluated separately:

$$\frac{\partial E}{\partial a} = 2(a_o - y) \quad (5)$$

$$\frac{\partial a_o}{\partial z} = \sigma'(z) \quad (6)$$

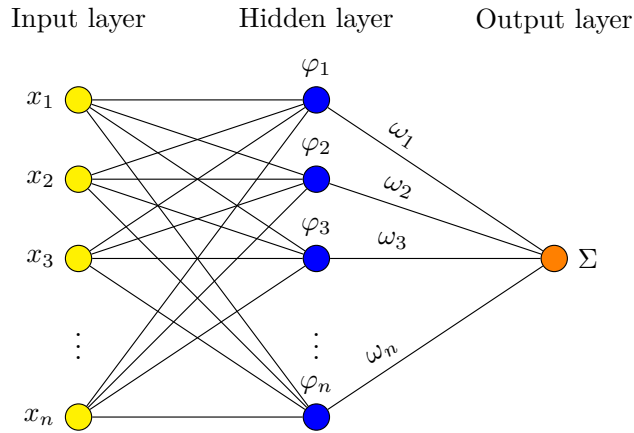
$$\frac{\partial z}{\partial w} = \sum_{i=1}^j a_{i,o-1} \quad (7)$$

By inserting equation (5), (6) and (7) into equation (4) the following expression for the partial derivative of the error with respect to the weights of the network is obtained:

$$\frac{\partial E}{\partial w} = 2a_o(a_o - y)\sigma'(z) \quad (8)$$

After computing the change to output layer weights, the process is repeated for each layer in the network, working backwards until the input layer is reached. For all layers before the output layer, the error is the weighted sum of each of the neurons error, in the next forward layer.

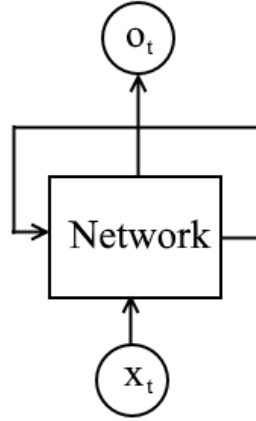
Figure 1: Neural network with 3 layers; an input layer, a hidden layer and an output layer.
 \LaTeX code for producing figure inspired by Stack Exchange user «Tom Bombadil»



2.3 Recurrent Neural Networks

A recurrent neural network (RNN) is a kind of neural network where the network takes into account not only the current input, but also the *context* of the input, i.e. previous inputs. The need for this becomes apparent when considering predicting the next word in a sentence, given the previous word as input. The best the network can do without a proper context is to predict the next word to be the most commonly appearing word after the word given as input. Recurrent neural networks attempt to solve this problem by considering the current input in concert with the previous inputs. In order for the network to remember previous inputs, it needs to have an inner state that is dependent upon what it has previously been given as input, i.e. *memory*. This concept is illustrated in figure 2.

Figure 2: Recurrent neural network receiving its previous state, as well as a new input, to produce a new output.



2.3.1 Long Short-Term Memory

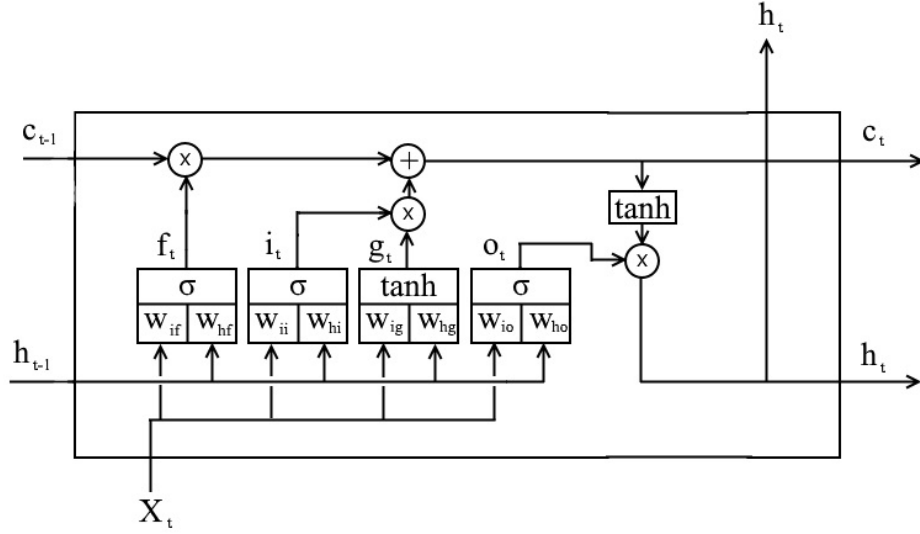
A long short-term memory (LSTM) network is a special type of recurrent neural network which has proven especially capable of remembering and thus taking into consideration inputs given arbitrarily far back in the sequence of network inputs [3]. Like any other recurrent neural network, LSTM networks pass its internal state to the next instance of the network. However, LSTM networks use an elaborate architecture for deciding what information from the previous state to «forget» and what information from the new input to include in its state.

As shown in figure 3, a LSTM network receives both the cell state and the output of from the previous instance of the network. First the network decides what, if any, of the information from the previous cell state to forget. This decision is made by a neural network which considers both the previous output h_{t-1} and the current input x_t . The output from the «forget network», f_t , containing values ranging from 0 to 1, is multiplied with the previous cell state thus potentially removing information from the previous cell state.

$$f_t = \sigma(W_{if} \cdot x_t, W_{hf} \cdot h_{t-1}) \quad (9)$$

In order to decide which new information to include in the cell state, the input gate, whose output is labelled i_t , decides which elements in the cell state to update by producing a new vector containing values ranging from 0 to 1. Each element in i_t is multiplied with the corresponding element in g_t , which can be thought of as new potential values, and the resulting vector is added to the modified cell state, yielding the current cell state c_t .

Figure 3: LSTM cell.



$$i_t = \sigma(W_{ii} \cdot x_t, W_{hi} \cdot h_{t-1}) \quad (10)$$

$$g_t = \tanh(W_{ig} \cdot x_t, W_{hg} \cdot h_{t-1}) \quad (11)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \quad (12)$$

Finally, the output h_t is computed by point-wise multiplication of the output from the last neural network layer o_t and the cell state, after each element is compressed to a value ranging from 0 to 1 by the \tanh function.

$$o_t = \sigma(W_{io} \cdot x_t, W_{ho} \cdot h_{t-1}) \quad (13)$$

$$h_t = o_t \odot \tanh(c_t) \quad (14)$$

2.4 Overfitting

Overfitting refers to instances where a machine learning model essentially memorise the training data as opposed to learning the common features shared among objects of the same type. This often expresses itself as high accuracy for the training data set and lower accuracy for the validation and test set, ie. the model is *not generalising well* from the training set to other data, essentially rendering the model useless.

As an example, imagine training a network to recognise and classify handwritten letters. For the character A, the underlying signal that should be captured has to do with the shape of the input. The model should not conclude that the input is the character A before all the telltale signs are identified, such as two lines forming a bottomless cone with a third line horizontally crossing the cone at about mid height. This is referred to as the signal, the true underlying abstract pattern of the capital letter A. Randomness or non-essential information that is irrelevant for predicting the input correctly can be essentially noise obfuscating the signal. The network may find some particular noise that, for the training data set, is strongly correlated with the letter A, or, given enough free parameters and/or small enough data set, the network can essentially memorise what output a particular input corresponds to.

One popular technique to avoid overfitting is to randomly deactivate neurons in the network, thus forcing the network to develop alternative signal paths and detect other patterns in the data, avoiding co-adaptation of neurons. This has been shown to effectively mitigate overfitting and increasing network performance on validation and test data in a wide variety of networks.[4]

3 Earthquakes

Earthquakes are seismic events that cause a measurable shaking of the planet surface. These events are caused by fracture propagation along a fault plane, releasing stored up elastic energy. Tectonic plates on the surface of the earth is continuously drifting, and in the interface between two plates the friction can cause potential energy to be stored. Eventually the potential energy becomes large enough to overcome the friction, and an earthquake occurs.

3.1 Richter Scale and Moment Magnitude













The magnitude of an earthquake is often presented in the Richter scale. This number refers to the logarithm of the wave caused by the earthquake and recorded by a seismograph. However, this is not a method suited well for measurements done over larger distances. The moment magnitude scale is now the preferred parameter when quantifying earthquake magnitude³, and is calculated from the total amount of energy released in the earthquake⁴

3.2 Focal Mechanism and the Moment Tensor

The direction of slip and resulting deformation of the epicenter of an earthquake is referred to as the focal mechanism. This can be described by the moment tensor, a mathematical way to describe the initial movement in the fault. The tensor consists of a matrix with 9 values, however only 6 are required to calculate the movement. The matrix consists of 3 x 3 values representing movement in axes.

To visualise a moment tensor, so called «Beachballs» are used. Beachballs delivers a graphical representation of the tensor. Beachballs consists of a circle with varied colored fields, where a color of white represents direction of compression and black or red presents area of expansion. See figure 4.

Figure 4: Moment tensor visualized as beachballs.

Moment tensor	Beachball	Moment tensor	Beachball
$\frac{1}{\sqrt{3}} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$		$-\frac{1}{\sqrt{3}} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	
$-\frac{1}{\sqrt{2}} \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$		$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$	
$\frac{1}{\sqrt{2}} \begin{pmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix}$		$\frac{1}{\sqrt{2}} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{pmatrix}$	
$\frac{1}{\sqrt{2}} \begin{pmatrix} -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$		$\frac{1}{\sqrt{2}} \begin{pmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	
$\frac{1}{\sqrt{6}} \begin{pmatrix} 1 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$		$\frac{1}{\sqrt{6}} \begin{pmatrix} -2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	
$\frac{1}{\sqrt{6}} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -2 \end{pmatrix}$		$-\frac{1}{\sqrt{6}} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -2 \end{pmatrix}$	

³<http://www.geo.mtu.edu/UPSeis/intensity.html>

⁴<http://www.usgs.gov/faqs/>

4 Previous work

Artificial neural networks (ANN) with nonlinear layers of neurons has been with us for at least half a century, but did not prove superior to other machine learning architectures until recently. In 2011, an ANN demonstrated superhuman performance in a traffic sign recognition test [5]. The ANN had an error rate of 0.56% compared to an average error rate of 1.16% for the human group. In the last 5 years ANNs have made their way into a wide variety of fields including, but not limited to medical diagnosis [2], financial forecasting [6], weather forecasting [7] and even energy consumption forecasting [8]. In light of the recent advances in the field of ANNs, some researchers has attempted to utilise these techniques to predict seismic activity.

4.1 Neural Networks to predict Earthquakes in Chile

J. Reyes *et al.* attempted to predict whether an earthquake above a certain threshold magnitude would occur within a time interval of 5 days [9]. The authors used a feed forward neural network and trained it by giving it temporal information about the temporal changes in the b-value from Gutenberg Richter Magnitude law. The authors estimated the b-value for six different periods in temporal sequence, and gave the resulting b-value vector as input, and whether or not an earthquake above a certain threshold occurred within a certain period of time as the target output. Reyes *et al.* claimed to be able to predict seismic events with «great reliability», and suggest that further work should investigate the use of different measurable seismic activity indicators such as Hurst’s exponent as input for the model.

4.2 Probabilistic Neural Network for Earthquake Magnitude Prediction

Adeli *et. al.* presents a probabilistic neural network model (PNN) for predicting the largest magnitude earthquake within a given time period [10]. The network input vector contains a combination of historical data for seismic events that occurred in the region in question. The paper concludes that the neural network yielded good predictions for earthquakes of magnitude between 4.5 and 6.0, correctly predicting 2 out of 4 quakes. As input data the authors used what is commonly referred to as the eight seismic indicators. The same authors had previously developed a recurrent neural network model for predicting earthquakes which, by their own accord, achieved good results for events with magnitude greater than 6.0.

4.3 Medium-large Earthquake Magnitude Prediction in Tokyo with Artificial Neural Networks

According to Asencio-Cortès *et. al.*, most researchers in the field today are using hybrid methods which attempt to combine the strengths of the different techniques [11]. Much effort is being put into the study of which seismic indicators should be given as input to the models. As Reyes *et. al.* and others demonstrated, the temporal fluctuations in the b-value of the Gutenberg-Richter law is one indicator which seems to have a strong correlation with future seismic events.

5 Project Design

5.1 Data Source

All training- and validation data was obtained from the Global Centroid Moment Tensor (CMT) catalogue.⁵ Each earthquake entry in the global CMT database contains many features, as shown in the example below.

Figure 5: Global CMT sample .ndk file

```
PDEW 2017/01/01 14:12:08.1 3.66 128.60 25.9 0.0 5.6 NORTH OF HALMAHERA, INDO
S201701011412A B: 0 0 0 S:128 222 50 M: 0 0 0 CMT: 1 TRIHD: 1.4
CENTROID: 6.5 0.2 3.64 0.02 128.64 0.01 31.1 0.4 FREE Q-20170102103712
24 2.460 0.083 -0.492 0.042 -1.970 0.048 -0.027 0.041 0.435 0.042 -0.033 0.021
V10 2.503 84 264 -0.492 1 1 -2.013 6 91 2.258 182 39 91 1 51 89
```

The projects main focus is the information about magnitude, location, time of occurrence as well as the moment tensor. See section 3.2 for details about the moment tensor. The earthquake data used was limited to the Kurile Islands and the area around (see figure 6). The Kurile Islands were chosen due to the frequent seismic activity in the area. This will make the project more manageable and also narrow down the location of the earthquake predictions.

From this catalogue of over 90 000 seismic events, about 2155 events taking place at or near Kurile Islands from 1970 until 2018 were used. See also the Global CMT websites in depth explanation to the Global CMT catalog entries.^{6, 7}

Figure 6: The data used in this project was limited to approximately this area around the Kurile Islands



5.2 Input Data Model

An important aspect of every machine learning venture is a thoughtfully designed input data model. Ideally, it should only include information needed to predict future events, but the challenge is ascertaining which

⁵<http://www.globalcmt.org/CMTsearch.html>

⁶http://www.ldeo.columbia.edu/~gcmt/projects/CMT/catalog/allorder.ndk_explained

⁷See Ekstrøm et al., 2012 [12] and Dziewonski et al., 1981 [13] for papers describing the Global CMT catalog.

parameters this actually pertains to. One approach when the best input is not known, is to include as much data as possible in the input element, but this runs the risk of drowning the target signal in noise. Due to this uncertainty several different input models containing different earthquake parameters were implemented and tested.

5.3 Choice of Neural Network architecture

A core assumption of this project is that seismic events measured over time contain information about future seismic events, i.e. the sequence is not random. By carefully examining past events, it should therefore be possible to gain information about future events, and hopefully use these insights to obtain better predictions about future occurrences.

For simpler neural network (NN) architectures, like «multilayered perceptron», previous inputs does not affect the state of the network after training is complete; the sole predictor of the next output is the input itself. Recurrent neural networks (RNN) is a class of NN architectures where the network has an internal state which is determined by previous inputs, i.e. memory. This enables the network to take previous inputs as well as the current input into consideration when making a prediction. After careful investigation into different flavors of NN and especially RNNs, it was decided to focus on a variant of RNN which is called long short-term memory (LSTM). These networks set them self apart from other RNNs by being able to remember and consider input given arbitrarily far back in the sequence of inputs.

5.4 Tools

Python was chosen as the programming language due to its integration with common machine learning libraries and its high level approach to programming, allowing for rapid prototyping.

Keras is a high-level neural networks API that allows for easy and fast prototyping. It can run on top of several of the most popular neural networks toolkit. [14]

TensorFlow is an open-source machine learning library developed by the Google Brain team, initially intended for internal use. In 2015 it was released under the Apache 2.0 open source license, and has been a popular choice among developers ever since.

Jupyter notebook is a web-browser based integrated development environment (IDE) for Python. Due to its flexible setup and how the notebook-feature advocates a natural flow for the visualisation and documentation of data and results it was deemed good fit for this project.

MariaDB was used to store the data. This enabled easy access to data from Jupyter, as well as enabling querying and sorting with SQL commands. In addition, several group members have extensive prior experience using MariaDB.

Docker is a container through which the development environment was installed. This was done in order to ensure equal development environment and behaviour independent of OS.

6 Detailed Design

6.1 Setting up the environment

In order to get consistent results, the lab environments were set up in containers with Docker. Containerized environments ensured that each of the project members were running the same versions of all the software to be used, regardless of which underlying hardware and software they are using.

Each of the project members also needed to have access to the exact same data. This was accomplished with help of the Docker container with an SQL-database, populated with data from the Global CMT project.

After researching different machine learning frameworks for Python, Keras proved itself promising. In order to know whether Keras was suitable for this project, the project members tested Keras in simpler scenarios. This also allowed the group to get familiar with the Keras API. These experiments were successful in showing that Keras was sufficient for this use case.

6.2 Experimenting with data

The group determined that having a common framework for managing data would become convenient, as data management is a critical element in any machine learning project. This led to the construction of `Utils.py`, a file consisting of utilization classes and functions relevant for managing the data.

Having these utilization tools was advantageous, and made the code less complex during the experimenting. The file consisted of code which abstracted the database handling, made formatting and translation of the data easier, and simplified measurement of accuracy. A fair amount of time were spent on creating, testing and experimenting with various functions included in this file.

The `Utils.py` file contained multiple ground truth methods, each of which formatted or represented the data in various ways. Formatting the data was an important step to make good input data for the neural networks. After the construction of multiple ground truth methods, they were tested rigorously with different parameters and settings in order to ensure consistent results.

6.3 Testing the models

Each of the project members made their own experiments in Jupyter notebooks to find out which settings, parameters, and data formats were most promising. This experiments were done in iterations, where experiences were shared between the members in weekly and daily meetings. With an iterative plan the members could then learn from each other and apply new experiences to their own experiments.

Through both trial and error, and previous experiences, the team was able to elect some model candidates to be compared in further experiments. It was decided to use a model with an LSTM layer of 100 nodes following a «dense layer» as the output. A dense layer is fully connected layer which performs a linear operation on the layers input vector. See figure 7 for example of magnitude prediction model.

The activation function were to be evaluated during the testing. This would be a comparison between the ReLU activation function and the sigmoid activation function. The last parameters to the model was the optimizer and the loss function. Based on previous research the «adam» optimizer seemed the right way to go [15]. Binary crossentropy, a multi-label classification function, was chosen as the loss function.

7 Evaluation

The project divides earthquake prediction into two main sub-tasks. Predict the **magnitude** of the next earthquake.⁸ And predict the **time** of the next earthquake, both for magnitude above 5.5 and 6.5

A jupyter notebook was created for predicting magnitude, and one for predicting time. These notebooks were used as templates for evaluating the models. The notebooks import the data, format the data to the desired format, and generate the model. The final step is to compare the results predicted by the model with the results from a statistical baseline. See bar charts in appendix.

When testing the models, the two activation functions sigmoid and ReLu were chosen. The parameter «days» was introduced in order to test the impact of different accuracy levels. A low number of days will give a more fine grained prediction. Different numbers of epochs were also tested. In the epochs column, the «es» in «200es» refers to a method called early stopping, which is a method used to avoid overfitting. See table 5.

Table 5: Configurations used when training the models.

Test	Activation	Days	Epochs
0	relu	30	2
1	relu	30	200
2	relu	30	200es
3	relu	180	2
4	relu	180	200
5	relu	180	200es
6	sigmoid	30	2
7	sigmoid	30	200
8	sigmoid	30	200es
9	sigmoid	180	2
10	sigmoid	180	200
11	sigmoid	180	200es

A goal in this project is to create a model for earthquake prediction based on information about time, magnitude and mechanism of previous earthquakes. These are just some of the features that is contained in the Global CMT database. To be able to measure if these features have a stronger relevance than other data, three different data sets were tested. See table 6. Data set 2 includes Data sets 1 and 0, and data set 1 includes data set 0.

Table 6: Data sets used when training the models.

Data set 0 (9 features)	Data set 1 (9 + 4 = 13 features)	Data set 2 (9 + 4 + 9 = 22 features)
- Time (since previous event)	- Latitude	- Eigenvalue (3 values)
- Magnitude	- Longitude	- Plunge (3 values)
- Moment Tensor Exponent	- Depth	- Azimuth (3 values)
- Moment Tensor (6 values)	- Scalar moment	

3 data sets (see table 6) were tested with 2 different input formats. One where each input value is between -1 and 1, and one where each input value assigned to one of 10 categories. To keep track of the different combinations of data sets, and input formats, different input types were named. See table 7.

⁸In the data used for this project, the lowest recorded magnitude in the area of the Kurile Islands is 4.3. The «next earthquake» refers to earthquakes above this magnitude.

8 Results

After analysing all the model outputs, the results were divided into these categories:

Category A Model consistently predicted the most common output category.

Category B Model makes predictions that coincide with the statistical distribution of events.

Category C Model consistently predicted one of the most dominant output categories, but not the most dominant.

See table 7 for details about the different input types. See table 8 for the prediction results when predicting the magnitude of the next earthquake. See table 9 and 10 for the prediction results when predicting the time of the next earthquake.

Table 7: Types identifying the combination of the data set used, and the format of the input.

Input type	Data set	Format
00	9 features	between -1 and 1
01	9 features	discrete (10 categories)
10	13 features	between -1 and 1
11	13 features	discrete (10 categories)
20	22 features	between -1 and 1
21	22 features	discrete (10 categories)

Table 8: Model tests performed when predicting the **magnitude** of the next earthquake

Test	Activation	Days	Epochs	Input type	00	01	10	11	20	21
0	relu	30	2		A	B	C	B	A	B
1	relu	30	200		B	B	B	B	B	B
2	relu	30	200es		A	B	A	B	B	B
3	relu	180	2		A	B	A	B	A	B
4	relu	180	200		B	B	A	B	B	B
5	relu	180	200es		B	B	A	B	A	B
6	sigmoid	30	2		A	B	A	A	A	A
7	sigmoid	30	200		B	B	B	B	B	B
8	sigmoid	30	200es		B	B	A	B	A	B
9	sigmoid	180	2		A	A	A	A	B	A
10	sigmoid	180	200		B	B	B	B	B	B
11	sigmoid	180	200es		B	B	C	C	B	B

Table 9: Model tests performed when predicting the **time** of the next earthquake with magnitude above 5.5.

Test	Activation	Days	Epochs	Input type	00	01	10	11	20	21
0	relu	30	2		B	B	A	B	A	B
1	relu	30	200		B	B	B	B	B	B
2	relu	30	200es		A	B	A	A	A	B
3	relu	180	2		A	A	A	A	A	A
4	relu	180	200		A	B	A	B	A	B
5	relu	180	200es		A	A	A	A	A	A
6	sigmoid	30	2		A	A	A	A	A	A
7	sigmoid	30	200		B	B	B	B	B	B
8	sigmoid	30	200es		B	B	B	B	B	B
9	sigmoid	180	2		A	A	A	A	A	A
10	sigmoid	180	200		A	B	A	B	A	B
11	sigmoid	180	200es		A	A	A	A	A	A

Table 10: Model tests performed when predicting the **time** of the next earthquake with magnitude above 6.5.

Test	Activation	Days	Epochs	Input type	00	01	10	11	20	21
0	relu	30	2		A	A	A	A	A	A
1	relu	30	200		A	A	A	A	A	A
2	relu	30	200es		A	A	A	A	A	A
3	relu	180	2		A	A	A	A	A	A
4	relu	180	200		A	B	A	B	A	B
5	relu	180	200es		A	A	A	A	A	A
6	sigmoid	30	2		A	A	A	A	A	A
7	sigmoid	30	200		A	A	A	A	A	A
8	sigmoid	30	200es		A	A	A	A	A	A
9	sigmoid	180	2		A	A	A	A	A	A
10	sigmoid	180	200		A	B	A	B	A	B
11	sigmoid	180	200es		A	A	A	A	A	A

9 Discussion

The predictions obtained for earthquake magnitude and time span until next event did not outperform the baseline prediction models. The results seems to indicate that the different models either made predictions based on the statistical distribution of events or consistently predicted the most common category.

When the networks was made to train for a set number of epochs regardless of improvement in validation set predictions, the network often identified the distribution of the ground truth classification. Correct predictions for each category correlated closely with the percentage of events in the training set ground truth corresponding with the classifications multiplied by the number of events of that category in the test set. This phenomenon can be observed in figure 9, 12 and 14.

The models which utilized early stopping exhibited 3 different behaviours: either the predictions followed the same pattern as described above, or the network appeared to get stuck in a local minimum, predicting the most likely or second most likely category regardless of the input. This phenomenon can be observed in figure 8, 10, 11 and 13.

There are in the authors opinions several possible explanations for this observed network behaviour, the most likely one being that the information needed to predict parameters of future seismic event was not contained in the network input model.

10 Conclusions

10.1 Tools

Modern open source machine learning tools have high quality, and active and helpful communities. With some programming knowledge, the threshold for setting up a machine learning environment with Python, Keras and TensorFlow is low. Keras is particularly user friendly, and its high level abstractions makes Keras a good interface for TensorFlow, as well as other machine learning frameworks.

10.2 Model

The long short-term memory networks described in this paper were not able to predict parameters of future seismic events with any greater accuracy than what is obtained when always predicting the most common outcome. The networks mostly ended up either always predicting the most common category in the training sets ground truth, or approximated the statistical distribution of the classifications found in the training data. There is no observable difference in the results based on what data set is used as input to the models. This could be due to sparseness of the input model, not enough training data or the use of non-ideal network architectures.

References

- [1] Claire Drury. Estimated annualized earthquake losses for the united states. 2000.
- [2] Arpit Bhardwaj and Aruna Tiwari. Breast cancer diagnosis using genetically optimized neural network model. *Expert Systems with Applications*, 42(10):4611–4620, 2015.
- [3] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [4] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
- [5] Dan CireşAn, Ueli Meier, Jonathan Masci, and Jürgen Schmidhuber. Multi-column deep neural network for traffic sign classification. *Neural networks*, 32:333–338, 2012.
- [6] Paolo Tenti. Forecasting foreign exchange rates using recurrent neural networks. In *Artificial Intelligence Applications on Wall Street*, pages 567–580. Routledge, 2017.
- [7] Mohammad Valipour. Optimization of neural networks for precipitation analysis in a humid region to detect drought and wet year alarms. *Meteorological Applications*, 23(1):91–100, 2016.
- [8] Gamze Oğcu, Omer F Demirel, and Selim Zaim. Forecasting electricity consumption with neural networks and support vector regression. *Procedia-Social and Behavioral Sciences*, 58:1576–1585, 2012.
- [9] Jorge Reyes, A Morales-Esteban, and Francisco Martínez-Álvarez. Neural networks to predict earthquakes in chile. *Applied Soft Computing*, 13(2):1314–1328, 2013.
- [10] Hojjat Adeli and Ashif Panakkat. A probabilistic neural network for earthquake magnitude prediction. *Neural networks*, 22(7):1018–1024, 2009.
- [11] G Asencio-Cortés, Francisco Martínez-Álvarez, A Troncoso, and A Morales-Esteban. Medium-large earthquake magnitude prediction in tokyo with artificial neural networks. *Neural Computing and Applications*, 28(5):1043–1055, 2017.
- [12] G. Ekström, M. Nettles, and A.M. Dziewoński. The global cmt project 2004-2010: Centroid-moment tensors for 13,017 earthquakes. *Physics of the Earth and Planetary Interiors*, 200-201(nil):1–9, 2012.
- [13] A. M. Dziewonski, T.-A. Chou, and J. H. Woodhouse. Determination of earthquake source parameters from waveform data for studies of global and regional seismicity. *Journal of Geophysical Research: Solid Earth*, 86(B4):2825–2852, 1981.
- [14] François Chollet et al. Keras. <https://keras.io>, 2015.
- [15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Appendix

Figure 7: **Example of network configuration of input type 00.** (Magnitude prediction: Data set 0 - 9 features, and input format 0 - values between -1 and 1).

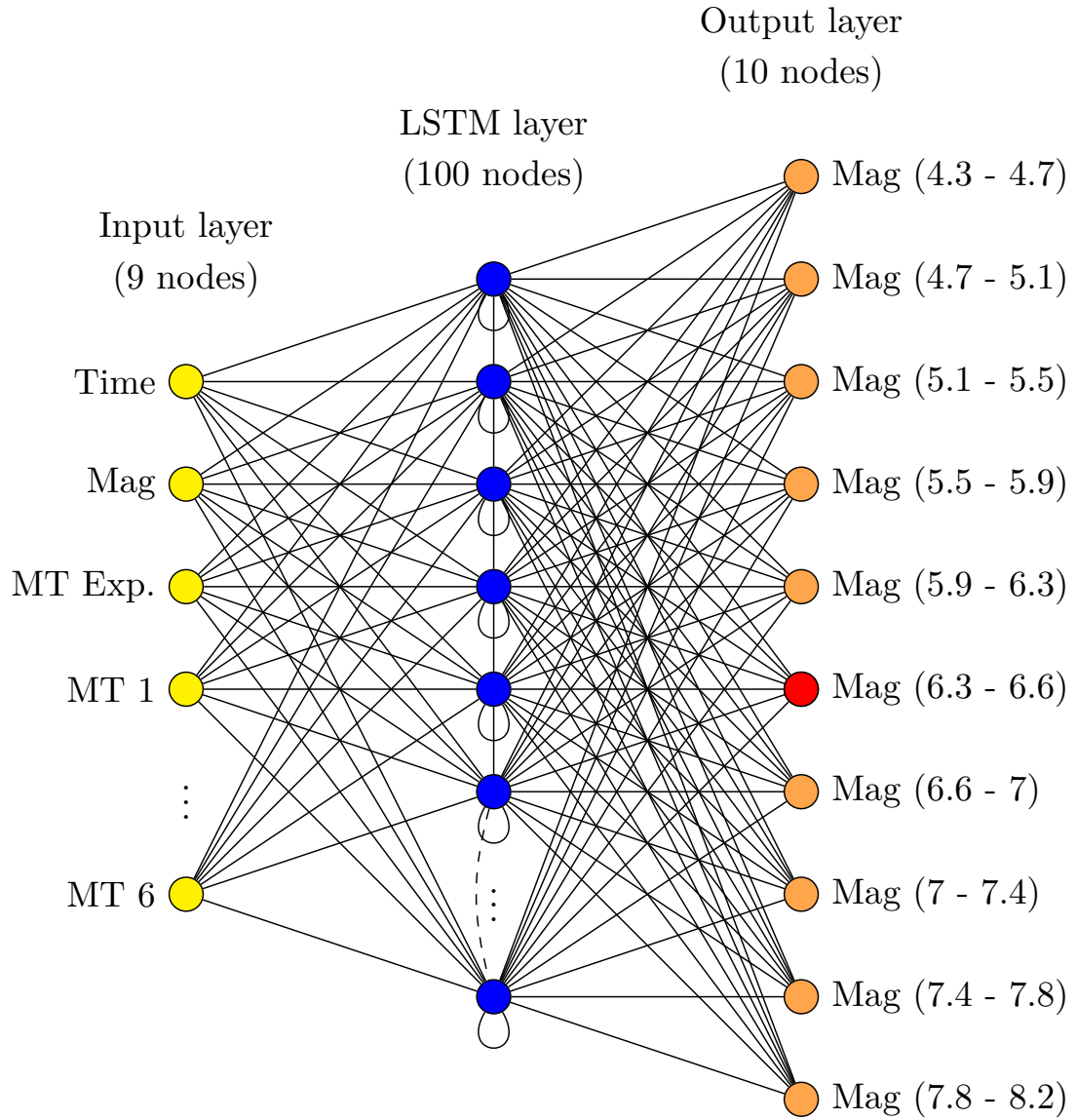


Figure 8: Predicting **magnitude** of upcoming earthquakes. Example of **category A**.

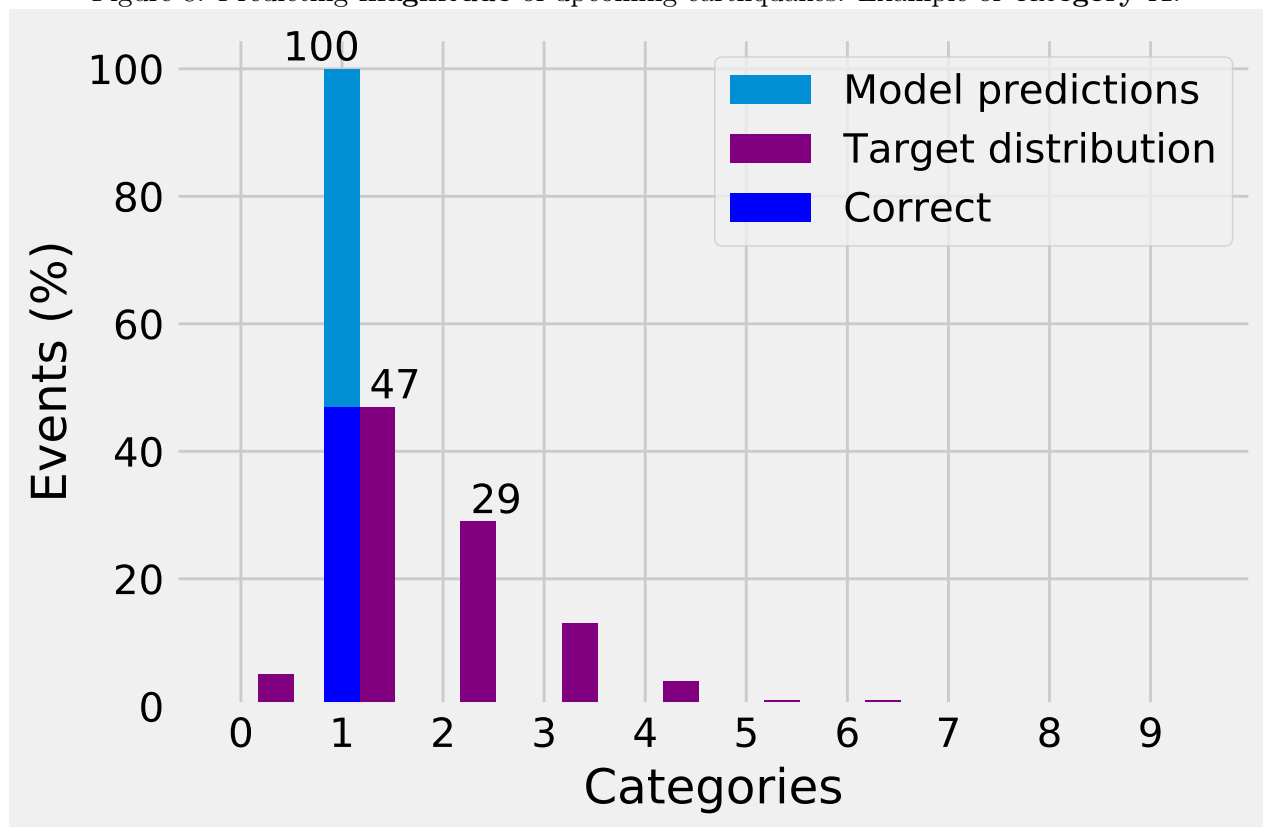


Figure 9: Predicting **magnitude** of upcoming earthquakes. Example of **category B**.

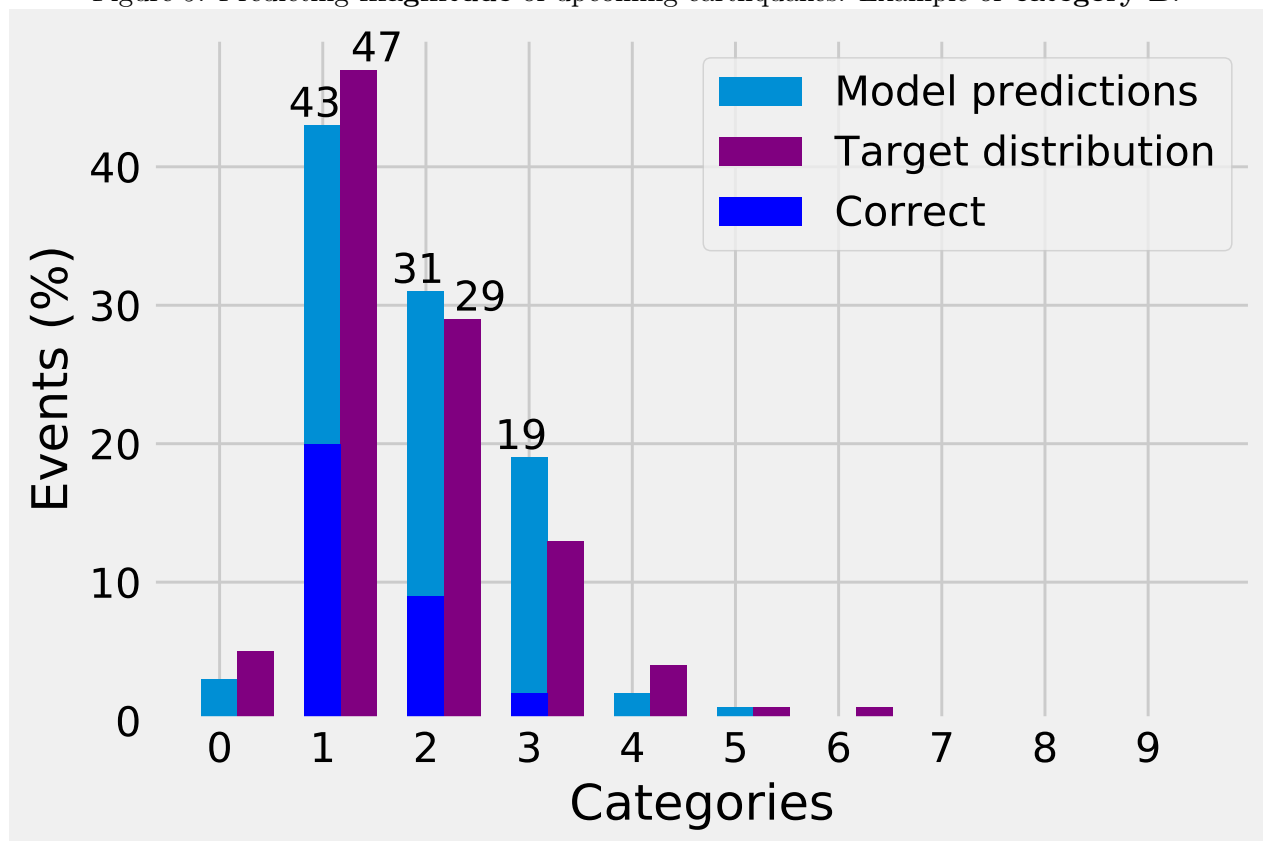


Figure 10: Predicting **magnitude** of upcoming earthquakes. Example of **category C**.

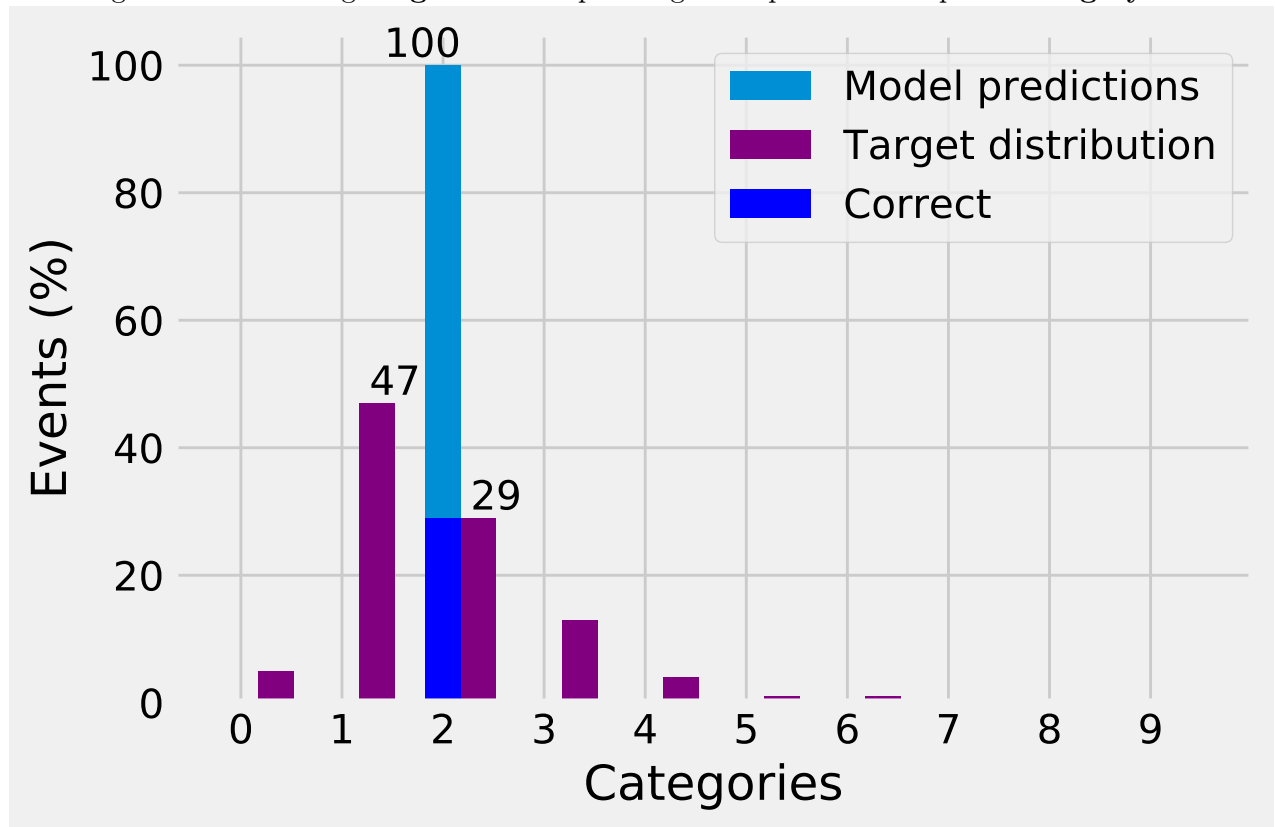


Figure 11: Predicting **time** of upcoming earthquakes (with magnitude above 5.5). Example of **category A**.

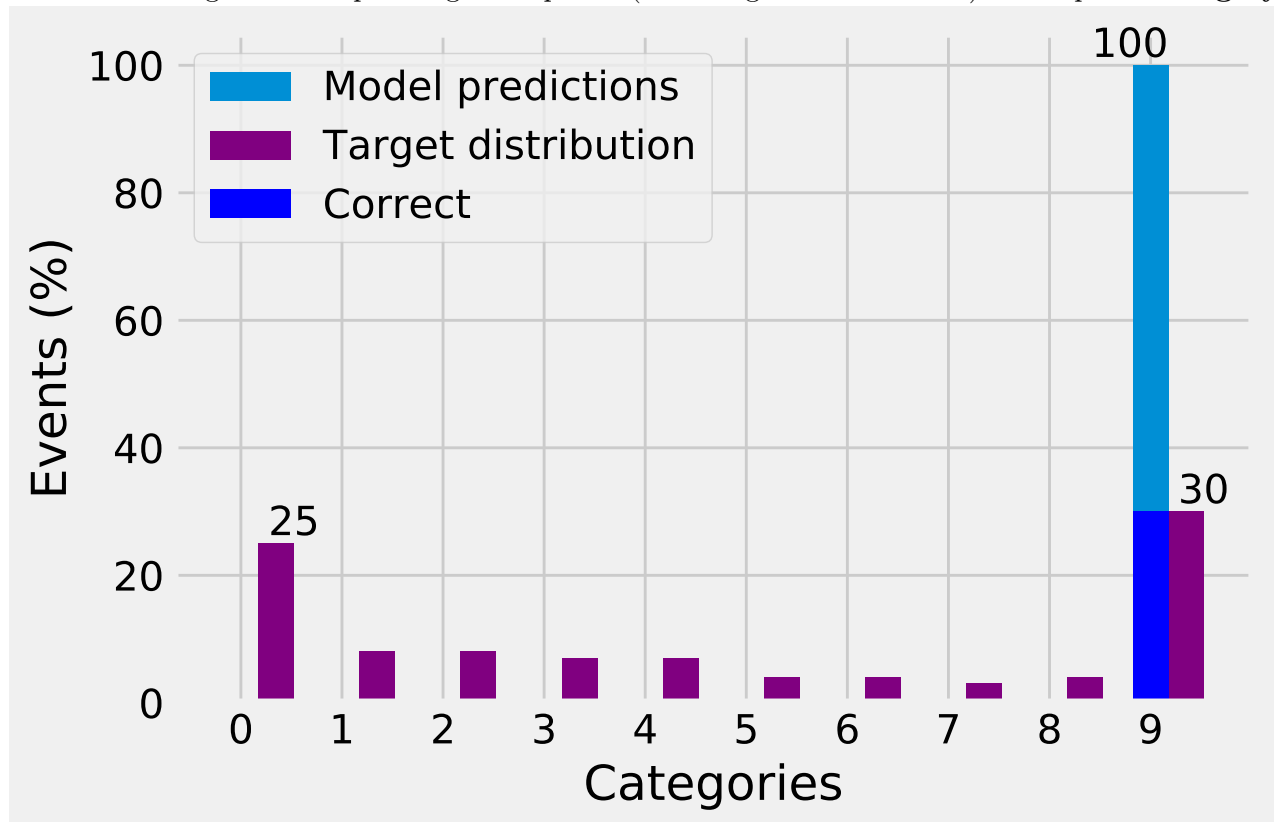


Figure 12: Predicting **time** of upcoming earthquakes (with magnitude above 5.5). Example of **category B**.

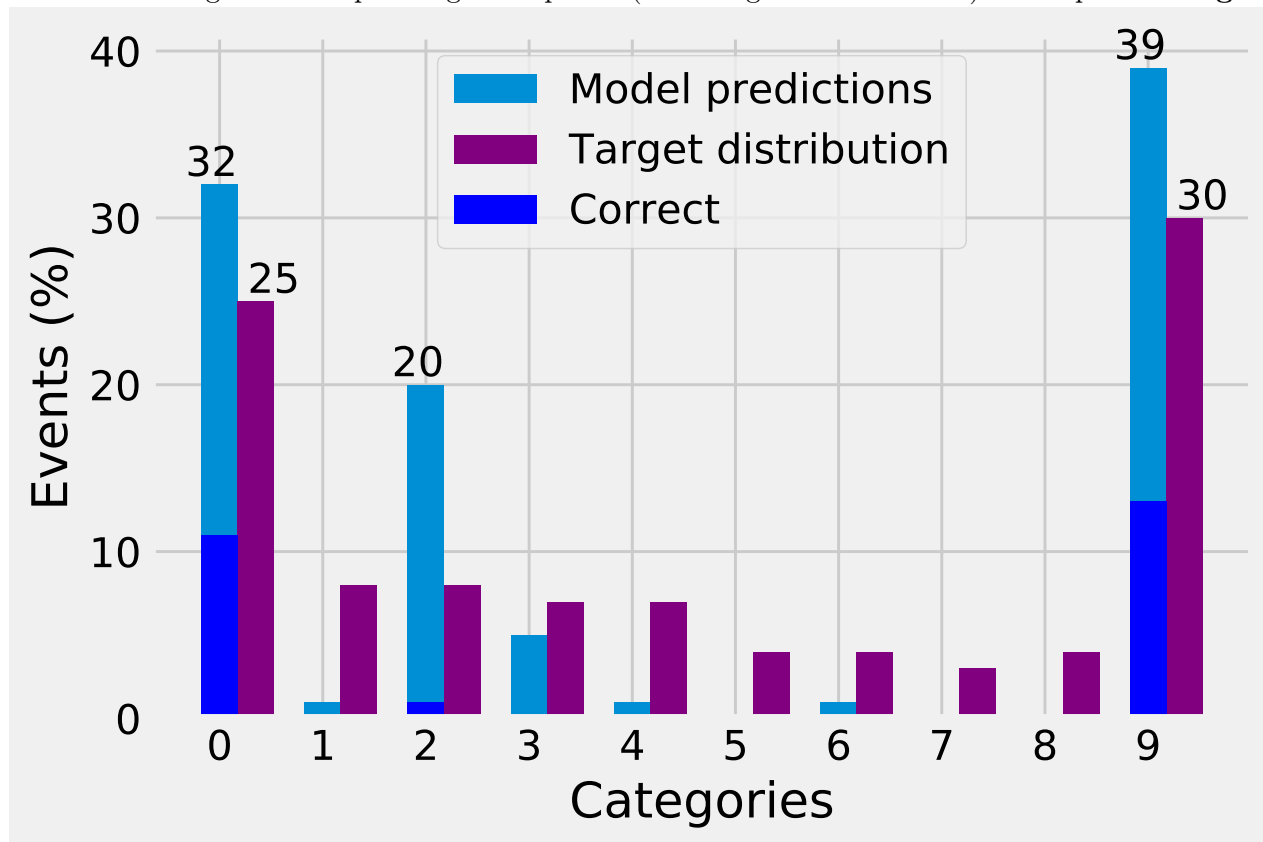


Figure 13: Predicting **time** of upcoming earthquakes (with magnitude above 6.5). Example of **category A**.

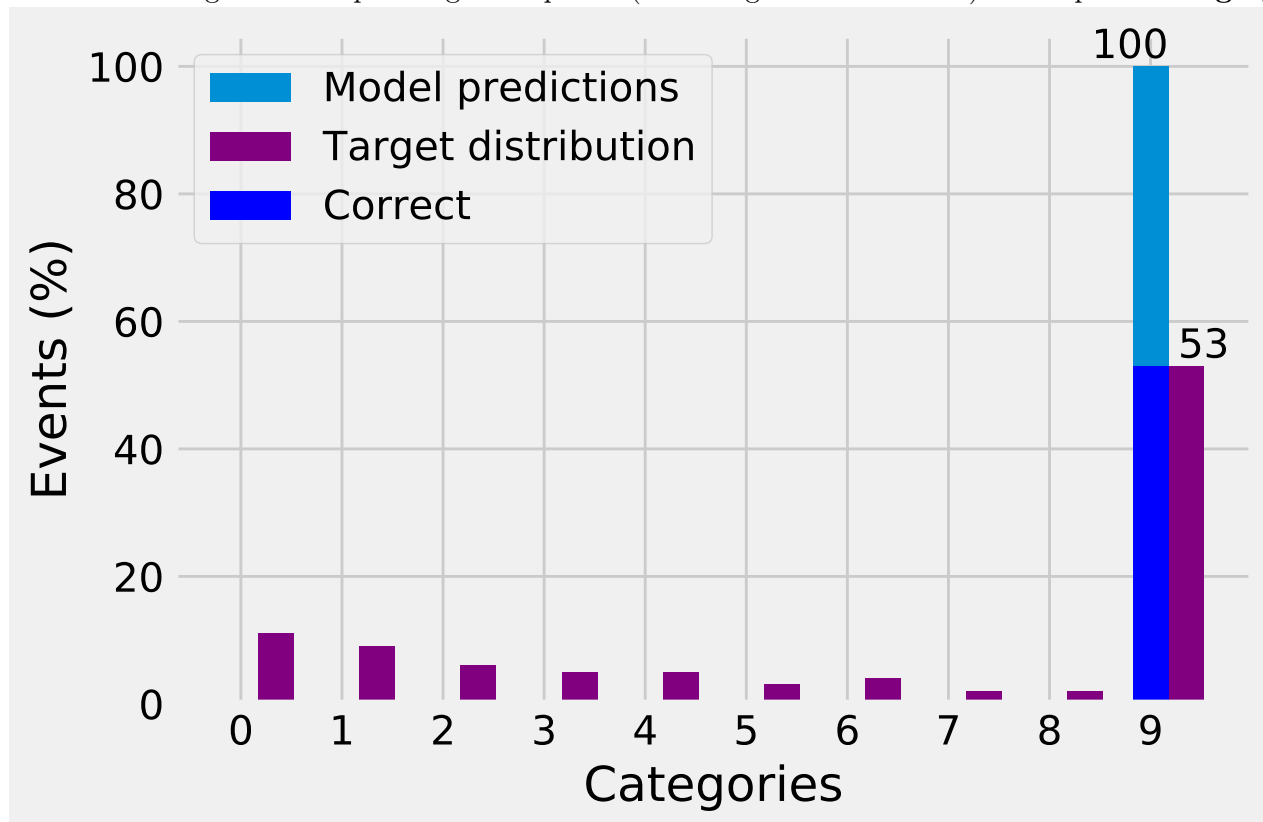


Figure 14: Predicting **time** of upcoming earthquakes (with magnitude above 6.5). Example of **category B**.

