Faculty of Information Technology

Information Systems

Final Project Report

Earthquake prediction

Done by: Tatlymuratov Rustembek,

Samarkhan Yeldan,

Yerkhassym Altynay,

Kissymova Gulnur

Checked by: Jongtae Park

Almaty, 2021

Motivation and Objective

Earthquakes are one of the most dangerous natural disasters, primarily due to the fact that they often occur without an explicit warning, leaving no time to react. This fact makes the problem of earthquake prediction extremely important for the safety of humankind. Despite the continuing interest in this topic from the scientific community, there is no consensus as to whether it is possible to find the solution with sufficient accuracy. Earthquake prediction is a very important problem in seismology, the success of which can potentially save many human lives. Various kinds of technologies have been proposed to address this problem, such as mathematical analysis, machine learning algorithms like decision trees and support vector machines, deep learning models and precursors signal study. Unfortunately, they usually do not have very good results due to the seemingly dynamic and unpredictable nature of earthquakes.

Related work and Originality

Despite the great effort made and the multiple models developed by different authors, no successful method has been found yet. Due to the random behavior of earthquake generation, it may never be possible to ascertain the exact time, magnitude and location of the next damaging earthquake.

This section explores the different approaches recently published in the literature related to the earthquake prediction. Most methods proposed to address the prediction of earthquakes can be divided into probabilistic approaches, which intend to discover the seismicity distribution, and the artificial intelligence techniques, which obtain a learning model from time series data.

Asim and et al researched earthquake prediction in Hindukush region using tree-based ensemble learning (Asim et al., 2017). They obtained an earthquake catalog from January 1980 to January 2016. In the paper, authors used seismic parameters such as time of n events, Gutenberg-Richter law, standard deviation of b value, seismic rate changes and probabilistic recurrence time. There were compared four earthquake prediction models naming Decision tree, Random forest, Rotation forest and RotBoost. The results were evaluated in terms of sensitivity, specificity, precision, F measure and area under curve (AUC). Authors find out that Rotation forest is able to attain higher prediction performance with regards to seismic activity in Hindukush region, compared to other classification methodologies and concluded that machine learning methods can play considerable role in prediction of seismic activity.

Neural network model was presented to predict earthquakes in Chile by J.Reyes and other authors (Reyes et al., 2013). They constructed three-layer feedforward artificial neural networks to predict earthquakes in four seismogenic areas in Chile. These networks included as input parameters information related to the b−value, the Bath's law and the Omori/Utsu's law. As output, the networks were able to predict the occurrence of earthquakes for a five-day horizon. The results were evaluated using specificity and sensitivity measures. With reference to the specificity, all the zones obtained values especially high − 87.2% on average. The sensitivity achieved is, in general, not so high − 40.9% on average.

G. Asencio-Cortes et al also researched ANN for earthquake forecasting task (Asencio-Cortés et al., 2017). They described a medium-large earthquake magnitude prediction in Tokyo in their paper. Authors used different seismic indicators as an input value. Two groups of features were defined to separate those dependent on b-value and those independent of such b value. The prediction algorithm is based on three layer ANN, implemented using a well-known multilayer perceptron that trains using back propagation technique. And it was compared with other machine learning algorithms such as KNN, SVM, NB and decision trees. Authors concluded that the proposed model outperforms other methods in terms of the quality parameters. The results achieved were considered as satisfactory since it has reached values greater than 70 % for all the five consecutive datasets considered.

Another research was conducted to predict a large earthquake magnitude in Taiwan by JP. Huang et al (Asencio-Cortés et al., 2017). They proposed deep learning based neural networks to predict an area, span of time and magnitude range of the earthquake. Neural network was designed consisting of convolutional and subsampling (also called pooling) layers, dropout was added to reduce overfitting. For better normalization, they

used ReLU activation function. Authors received good results when using the past 120 days of seismic events to predict the largest earthquake magnitude in Taiwan in the upcoming 30 days.

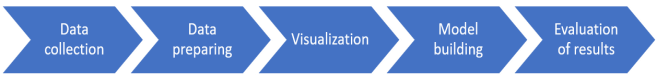
Speaking about the performance of proposed models, it is worth noting that it is hard to compare approaches proposed in different papers, because the researchers use different performance measures for assessing the quality of predictors. That is why one cannot objectively state that one model is better than the other is. However, Md. Hasan Al Banna et al studied different papers and compared the performance of AI techniques to predict earthquakes (Al Banna et al., 2020). According to this paper, SVR based and DT based-model have higher accuracy among classical machine learning based approaches. They also noted the IABC-MLP method as a high accuracy model for neural network-based approach and LSTM for deep learning-based approach.

To conclude, it is noticeable that most research was conducted on the repeatedly studied regions of Southern California, Chile and Hindukush. Also, most of the papers studied ANN as a proposed model for earthquake prediction. Another thing to note is that the models proposed in most of the papers reviewed were tested on data for different regions obtained from different earthquake catalogs. As shown in a number of papers, an approach may perform differently on zones with different seismic properties, and that is another reason why it is near to impossible to compare the methods proposed in different studies (Galkina & Grafeeva, 2019).

In this paper, we aimed to propose a model to predict earthquakes in the Almaty region. However, the dataset of history of Almaty earthquakes is incomplete and has unsuitable structure. Therefore we will use the dataset containing earthquake data happening in Turkey since 1900. In this paper, we will research the application of machine learning methods to predict earthquakes in Turkey.

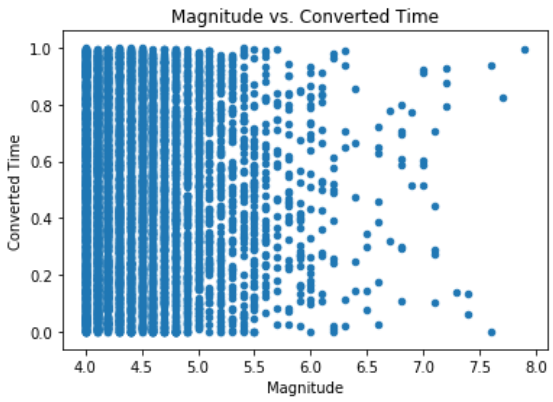
Design Architecture

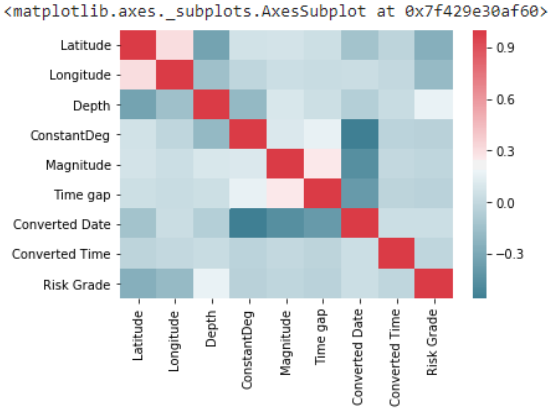
For our Final ML project we decided to build standard design architecture format procedure that is illustrated below:



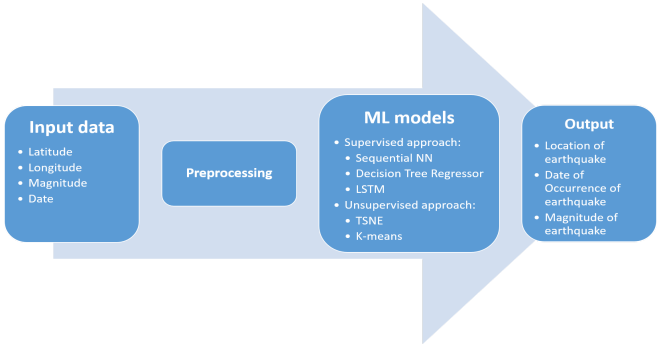
Firstly we obtained a dataset for kaggle into our project then in order to make it easy to work with rough data we made a data preprocessing part in which we changed some column added, new one and dropped some useless objects and columns from the dataset.

After that we builded scatter plot models and a heatmap(correlation matrix) in order to visualize and determine the feature dependencies and their correlations. Two picture below show example of our code:





In order to find better solution of predicting earthquakes we made different ML models for this problem:



The picture above shows the exact model building architecture.And in the last part of our project we evaluated our models separately in order to understand the efficiency of different model approaches.

Detailed Algorithm and functions

In the data preprocessing part we have some interesting self made algorithms that helped us analyze and process our data for later use.

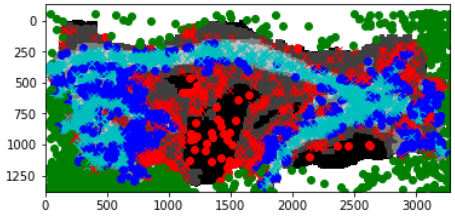
For example we used self-written methods below in order to work with Turkey risk map

```
def risk(latitude, longitude):

  west=25.67
  east=44.81
  south=35.81
  north=42.10

  real_width = east - west
  real_height = north - south

  map_width = np.shape(risk_map)[1]
  map_height = np.shape(risk_map)[0]

  width_ratio = map_width/(real_width*100)
  height_ratio = map_height/(real_height*100)

  # Calculating pixels to look up for the grade
  easting = longitude-west
  northing = latitude-south

  pixel_x = int(round(easting*100*width_ratio))
  pixel_y = map_height-int(round(northing*100*height_ratio))

  # Correction of the error caused by floating points
  if pixel_x >= map_width:
    pixel_x = map_width-1

  if pixel_y >= map_height:
    pixel_y = map_height-1

  grade=risk_map[pixel_y, pixel_x]

  return grade
```

```
def position(latitude, longitude):

  west=25.67
  east=44.81
  south=35.81
  north=42.10

  real_width = east - west
  real_height = north - south

  map_width = np.shape(risk_map)[1]
  map_height = np.shape(risk_map)[0]

  width_ratio = map_width/(real_width*100)
  height_ratio = map_height/(real_height*100)

  # Calculating pixels to look up for the grade
  easting = longitude-west
  northing = latitude-south

  pixel_x = int(round(easting*100*width_ratio))
  pixel_y = map_height-int(round(northing*100*height_ratio))

  # Correction of the error caused by floating points
  if pixel_x >= map_width:
    pixel_x = map_width-1

  if pixel_y >= map_height:
    pixel_y = map_height-1

  return [pixel_y, pixel_x]
```

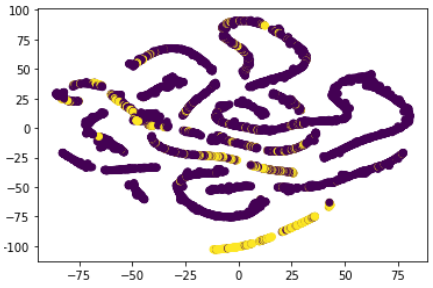
The risk() method helps us create and divide our risk_map into risk_states and add an additional column ’risk grade’ to our dataset. The second method helps illustrate illustrates every earthquake in dataset on the risk map:

```
plt.imshow(risk_map, cmap='gray', vmin=1, vmax=5)
colours = ['ro', 'rx', 'bo', 'cx', 'go']
grades = []
for i in range(len(df)):
  pos = position(df['Latitude'][i], df['Longitude'][i])
  # print(pos[0])
  plt.plot(pos[1], pos[0], colours[int(df['Risk Grade'][i] - 1)])
  # grades.append(position(df.loc[i].iloc[0], df.loc[i].iloc[1]))
# plt.plot(grades[:,0], grades[:,1], colo)
```



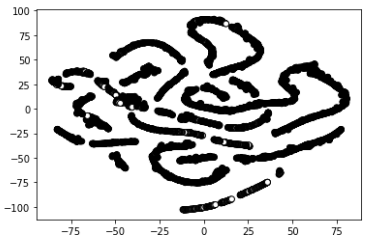
For our Final project of this ML course we decided to implement a set of different prediction models for this task.We divided our models into two groups: supervised learning models and unsupervised learning models and in this sub-groups we have tested different models and tried to predict different outputs for this problem such as “coordinates of the earthquake” , “the date of the earthquake in particular area”,”the times of day when will be earthquake ” and others in order to find a better solution model for this particular problem.

For the unsupervised learning models we decided to test TSNE approach for data visualization and K-means model in order to find clusters in dataset:




```
from sklearn.cluster import KMeans, AgglomerativeClustering
plt.scatter(x_tsne_clf[:,0], x_tsne_clf[:,1], edgecolor='black', c=y_clf, cmap='gray')
```

```
<matplotlib.collections.PathCollection at 0x7fdfd97cdb38>
```



According to the pictures we concluded that unsupervised learning is unsuitable for this dataset.In addition to 3 supervised sequential models that are described below, we tried to implement a decision tree regression model and LSTM model for predictions.

Coding

First, the model is built.

```
x = df.drop(columns = ['Latitude', 'Longitude', 'Magnitude', 'Converted Date']).to_numpy()
y = df[['Latitude', 'Longitude', 'Magnitude', 'Converted Date']].to_numpy()
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=1)
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.2, random_state=1)
x_train
```

```
scaler_x=MinMaxScaler(feature_range=(0,1))
scaler_y=MinMaxScaler(feature_range=(0,1))

x_train = scaler_x.fit_transform(x_train)
x_val = scaler_x.fit_transform(x_val)
x_test = scaler_x.fit_transform(x_test)

y_train = scaler_y.fit_transform(y_train)
y_val = scaler_y.fit_transform(y_val)
y_test = scaler_y.fit_transform(y_test)
y_train
```

The model has 5 inputs and uses a final Dense layer with 4 outputs.

```
model = Sequential()
# model.add(Dense(1000, input_shape = (5, ), activation='relu', kernel_regularizer=l2(0.001)))
# model.add(Dropout(0.5))
# model.add(BatchNormalization())
# model.add(Dense(500, activation='relu', kernel_regularizer=l2(0.001)))
# model.add(Dropout(0.5))
# model.add(BatchNormalization())
# model.add(Dense(250, activation='relu', kernel_regularizer=l2(0.001)))
# model.add(Dropout(0.5))
model.add(BatchNormalization())
model.add(Dense(128, input_shape = (5, ), activation='relu'))
model.add(BatchNormalization())
model.add(Dense(4))
# optimizer = RMSprop(lr=0.001, rho=0.9)
model.compile(optimizer = 'rmsprop', loss='mse', metrics=['mae'])
```

```
step_size = 800
clr = CyclicLR(base_lr=0.0001,
               max_lr=0.001,
               step_size=step_size,
               mode='exp_range',
               gamma=0.99994)
```

Next, we train the model and set hyperparameter settings: epochs=700, batch_size=32.

```
model.fit(x_train, y_train, epochs=700, batch_size=32, shuffle=True,
  validation_data=(x_val, y_val), verbose=1,
  callbacks=[
    clr,
    ModelCheckpoint('./models/model.h5', monitor='val_loss', verbose=1, save_best_only=True, mode='auto'),
    ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=5, verbose=1, mode='auto')
  ]
)
```

The following two other models were created and trained the same way.

Applying unsupervised learning or trying to clusterize was a failure due to the dataset structure. Therefore, some tinkering was done and additional models, namely Decision Tree and LSTM, were tested.

```
y_clf=df['ConstantDeg'].to_numpy()
from sklearn.manifold import TSNE
tsne=TSNE(n_components=2)
x_tsne_clf=tsne.fit_transform(df.drop(columns = 'ConstantDeg'))
X_train_c2D2, X_test_c2D2, y_train_c2D2, y_test_c2D2 = train_test_split(x_tsne_clf, y_clf, test_size=0.33, random_state=42)
plt.scatter(x_tsne_clf[:,0], x_tsne_clf[:,1], c=y_clf)
```

The creation and training of the Decision Tree model.

```
x = df.drop(columns=['Latitude', 'Longitude', 'Magnitude',
                     'Converted Date']).to_numpy()
y = df[['Latitude', 'Longitude', 'Magnitude', 'Converted Date']].to_numpy()

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,
                                                    random_state=1)
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train,
                                                  test_size=0.2,
                                                  random_state=1)

scaler_x = MinMaxScaler(feature_range=(0, 1))
scaler_y = MinMaxScaler(feature_range=(0, 1))

x_train = scaler_x.fit_transform(x_train)
x_val = scaler_x.fit_transform(x_val)
x_test = scaler_x.fit_transform(x_test)

y_train = scaler_y.fit_transform(y_train)
y_val = scaler_y.fit_transform(y_val)
y_test = scaler_y.fit_transform(y_test)
```

```
model = tree.DecisionTreeRegressor(max_depth=20, splitter="random")

model = model.fit(x_train, y_train)
prediction = model.predict(x_test)
```

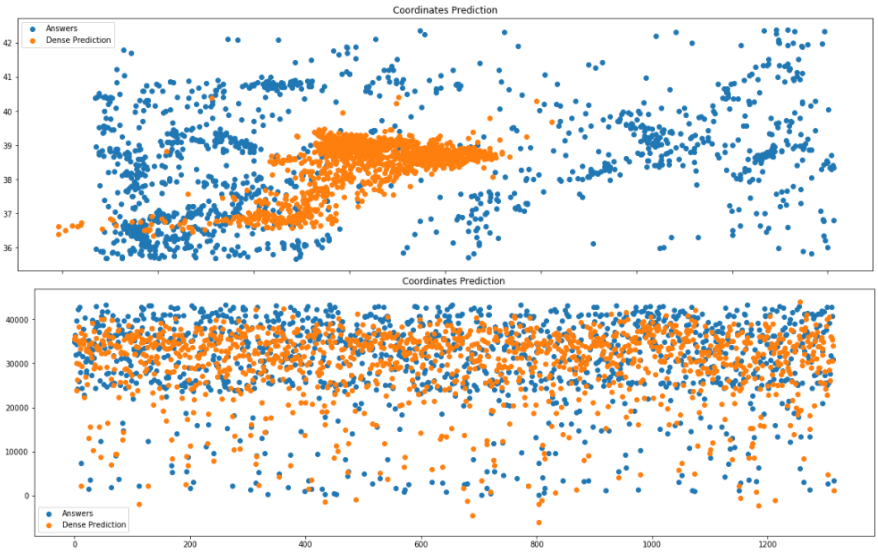
The creation, training, and compiling of the LSTM model.

```
model = Sequential()
model.add(LSTM(30, activation='relu', input_shape=(20, 9)))
model.add(Dropout(0.5))
model.add(BatchNormalization())
model.add(Dense(4))
```

```
model.compile(optimizer='adam', loss='mse', metrics=['accuracy', 'mae'])
```

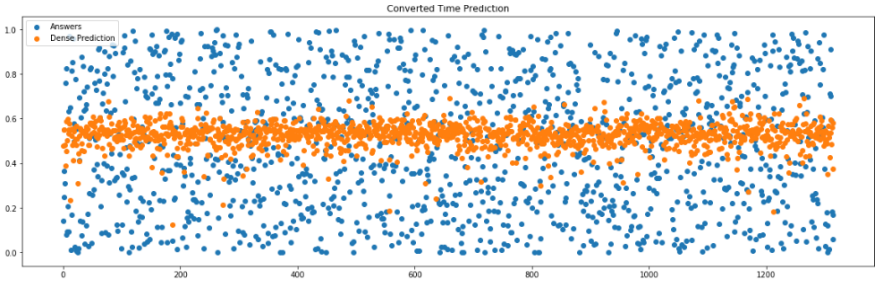
Results and Performance Evaluation

All three initial models were evaluated and the results were displayed in the graph based on coordinates, and converted time. The risk map of Turkey was used as a basis. Here are the visualized predictions:



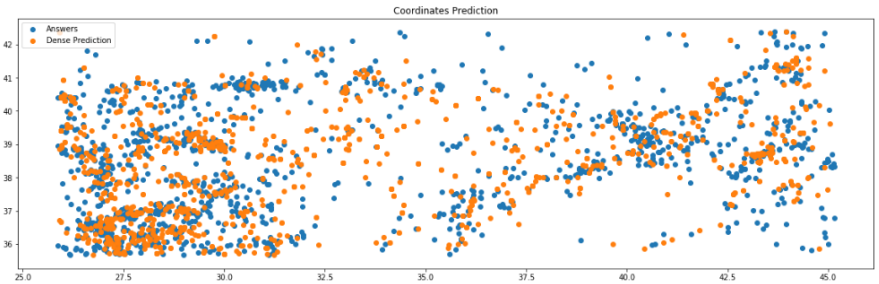
As we can observe from this graph of the first model, predictions are awful. These were predictions based on coordinates.

From the second graph it can be seen that predictions are better than previous one, however not precise. These were predictions based on coordinates.

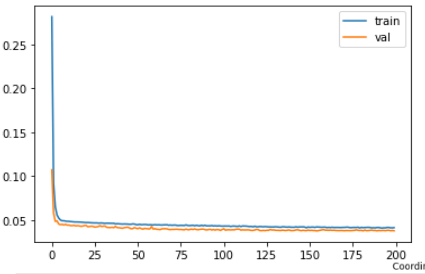


The last graph was based on converted time (expected time of earthquake happening). It can be clearly noticed that predictions were unacceptable. As they were indicating midday earthquakes.

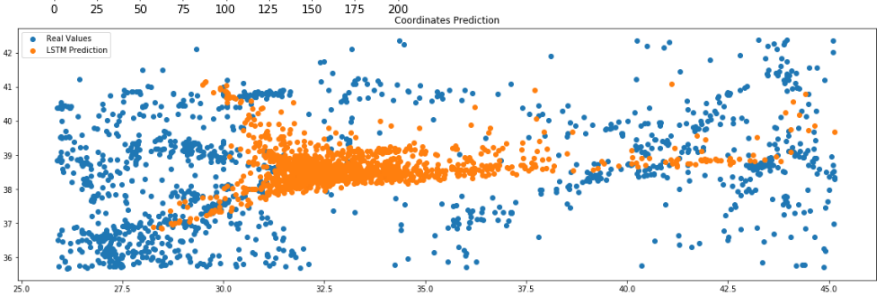
Next, the results for Decision Tree and LSTM models are shown below.



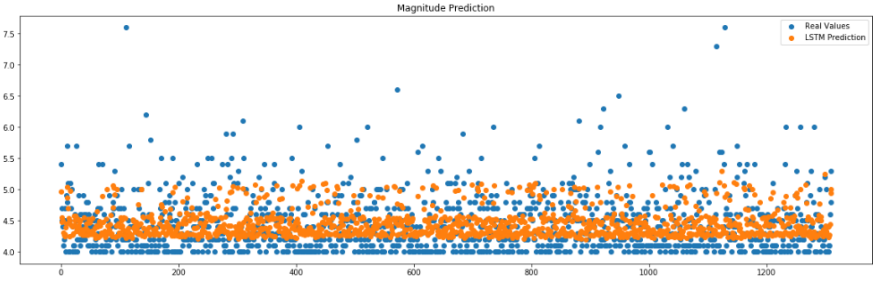
The Decision Tree model predictions were made based on coordinates. Accuracy of this method looks average.



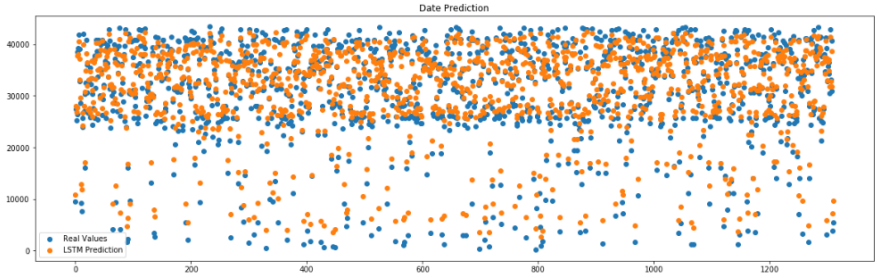
In the LSTM model, the comparison of trained and validated loss was shown in the graph. The predictions were made based on following features: coordinates, magnitude, and date, and were plotted on the three charts below.



It can be viewed that based on coordinates the predictions were a complete miss.



The second chart that was based on magnitude shows ok results with magnitude ranging from 4.25 to 5. The results are better in comparison with the first chart, still they are not accurate.



The last chart, which is based on date, demonstrates the best results among all the models used in this project with an accuracy of 80% approximately.

Conclusion

After all the work done, we can conclude that there is not enough data to solve this problem, and it is almost impossible to predict the next earthquake with great accuracy. The graphs show that the parameters in the available data set do not have a strong relationship with each other. Since earthquakes are the random natural phenomenons that can occur anywhere in the world and at any time, we can not be absolutely sure that the results of the developed models will be accurate.

To sum up, as it was described in previous parts different models were used to solve our problem. Both supervised and unsupervised models were implemented and evaluated.

Prediction efforts in this study relies on checking out whether there are any correlations with former earthquakes. According to graphics above:

- For placement prediction: both models are not well enough to alarm anybody.
- For days prediction: Dense model is pretty better than LSTM model.
- For magnitude prediction:LSTM model seems better, but it can not be well as desired.

In total, Dense model is better for days prediction and LSTM model is better for others.

In any case, our attempt to develop the most effective model for solving such problems related to natural phenomena has shown that we are not able to predict with high accuracy the place, time and magnitude of the next earthquake.

References

1. Al Banna, M. H., Taher, K. A., Kaiser, M. S., Mahmud, M., Rahman, M. S., Hosen, A. S. M. S., & Cho, G. H. (2020). Application of Artificial Intelligence in Predicting Earthquakes: State-of-the-Art and Future Challenges. *IEEE Access*, *8*, 192880–192923. https://doi.org/10.1109/ACCESS.2020.3029859
2. Asencio-Cortés, G., Martínez-Álvarez, F., Troncoso, A., & Morales-Esteban, A. (2017). Medium–large earthquake magnitude prediction in Tokyo with artificial neural networks. *Neural Computing and Applications*, *28*(5), 1043–1055. https://doi.org/10.1007/s00521-015-2121-7
3. Asim, K. M., Idris, A., Martinez-Alvarez, F., & Iqbal, T. (2017). Short Term Earthquake Prediction in Hindukush Region Using Tree Based Ensemble Learning. *Proceedings - 14th International Conference on Frontiers of Information Technology, FIT 2016*, 365–370. https://doi.org/10.1109/FIT.2016.073
4. Galkina, A., & Grafeeva, N. (2019). Machine learning methods for earthquake prediction: A survey. *CEUR Workshop Proceedings*, *2372*(June), 25–32.
5. Huang, J. P., Wang, X. A., Zhao, Y., Xin, C., & Xiang, H. (2018). Large earthquake magnitude prediction in Taiwan based on deep learning neural network. *Neural Network World*, *28*(2), 149–160. https://doi.org/10.14311/NNW.2018.28.009
6. Reyes, J., Morales-Esteban, A., & Martínez-Álvarez, F. (2013). Neural networks to predict earthquakes in Chile. *Applied Soft Computing Journal*, *13*(2), 1314–1328. https://doi.org/10.1016/j.asoc.2012.10.0
7. Tensorflow, https://www.tensorflow.org/
8. Keras, https://keras.io/

Roles of Members

Tatlymuratov Rustembek – construction of models
Samarkhan Yeldan – construction of models
Yerkhassym Altynay – data visualization and analysis
Kissymova Gulnur – data preparation