ORIGINAL RESEARCH PAPER

Machine Learning

PREDICTING THE SIGNIFICANCE OF EARTHQUAKES USING MACHINE LEARNING ALGORITHMS

KEY WORDS:
Hyperparameters, Machine learning, Earthquakes, features

Ruchit Karnik Indian School Wadi Kabir Cambridge Student

ABSTRACT

Earthquakes impose a critical hazard to society. Their existence leads to a significant loss of life, habitat and infrastructure. What adds significantly to these grave adversities is poor disaster management and response. Governments and international bodies sometimes cannot appropriately allocate scarce resources due to clouded judgment and personal biases, among several other reasons. This research aims to completely mitigate this problem by predicting the significance of ongoing earthquakes using machine learning models, providing a reasoned basis for the distribution of resources in more than one region affected by earthquakes. The models used are Linear regression, K-nearest neighbour, Decision tree, XGBoost and Random Forests. Hyperparameters were used in order to improve accuracy. These models first learn essential data such as magnitude and exact location from 1000 earthquakes that occurred from 1995 to 2023 and then are tested for accuracy in predicting the significance of earthquakes from the testing data. In this case, the Random Forest model emerged as the best according to all three evaluation metrics. Therefore, it was used as the basis of the program to predict the significance of earthquakes. Data such as the MMI, CDI, latitude and longitude can be entered into the program to generate a well-informed prediction of the importance of an earthquake.

INTRODUCTION

Earthquakes pose a very serious threat to lives and infrastructure. Their occurrence slows down an economy, resulting in falling economic growth. In addition, when a country is severely affected, its average living standards fall, leading to a very poor quality of life. What drastically amplifies these impacts is poor disaster management. This happens when resources are not allocated correctly for the best possible outcome. This research aims to accurately predict the significance of earthquakes according to several metrics to help international authorities appropriately allocate their scarce resources between areas affected by earthquakes.

Dataset

The dataset for this project was obtained from Kaggle (Kaggle, n.d.). It included data for earthquakes from 1995 to 2023. The following features were present in the dataset:

- **Title:** title name given to the earthquake
- **Magnitude:** the magnitude of the earthquake
- **Date_time:** date and time
- **Cdi:** the maximum reported intensity for the event range
- **Mmi:** the maximum estimated instrumental intensity for the event
- **Alert:** the alert level - "green", "yellow", "orange", and "red"
- **Tsunami:** "1" for events in oceanic regions and "0" otherwise
- **Sig:** a number describing how significant the event is. Larger numbers indicate a more significant event. This value is determined on a number of factors, including: magnitude, maximum mmi, felt reports, and estimated impact
- **Net:** the id of a data contributor. Identifies the network considered to be the preferred source of information for this event.
- **Nst:** the total number of seismic stations used to determine earthquake location.
- **Dmin:** horizontal distance from the epicenter to the nearest station
- **Gap:** the largest azimuthal gap between azimuthally adjacent stations (in degrees). In general, the smaller this number, the more reliable is the calculated horizontal position of the earthquake. Earthquake locations in which the azimuthal gap exceeds 180 degrees typically have large location and depth uncertainties
- **Magtype:** the method or algorithm used to calculate the preferred magnitude for the event
- **Depth:** the depth where the earthquake begins to rupture
- **Latitude / longitude:** coordinate system by means of which the position or location of any place on earth's surface can be determined and described
- **Location:** location within the country
- **Continent:** continent of the earthquake hit country
- **Country:** affected country

The dataset was downloaded as a CSV file to use for further processing.

Methodology

Features

The features used were magnitude, cdi, mmi, depth, latitude and longitude. The reason for excluding other features is that some features do not at all have an impact on the significance of an earthquake. For example, 'Gap' was not included because there is no important relation between the distance between the earthquake and the station. The features used have a very strong impact on the significance of an earthquake and hence were used.

Machine Learning Models And Code

The machine learning models used are linear regression, random forest, decision tree, XGBoost and K-nearest Neighbour (KNN).

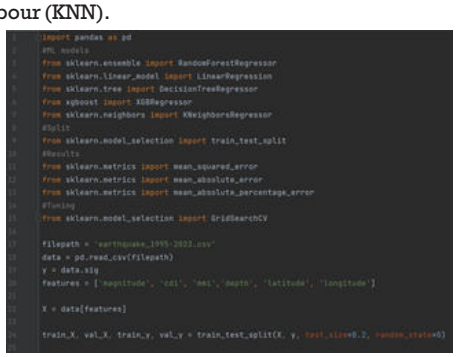

Figure 1: Common code

The above code first imports Pandas and, next, all the necessary machine learning models. It also imports the train test split feature (Geeksforgeeks, n.d.) in line 9. This feature splits the datasets into training and testing sets. This means that some part of the dataset will be used to train the models, and the remaining part will be used to test the accuracy of the model. Moving on, three types of evaluating metrics are imported: the mean squared error (scikit-learn, n.d.), the mean absolute error (scikit-learn, n.d.) and the mean absolute percentage error (scikit-learn, n.d.). These three metrics are used to obtain a comprehensive evaluation of

each model. Each metric captures different error characteristics: the mean squared error is sensitive to outliers and hence highlights more significant errors. The mean absolute error offers a straightforward measure of the average prediction error, regardless of outliers. Finally, the mean absolute percentage error allows the error to be interpretable as a percentage, allowing it to be easily understood. Moving on, GridSearchCV (Verma, 2023) was imported to select the best hyperparameters and so improve the performance of all models. Lines 17 and 18 tell the program which file to analyse, and line 19 sets the significance of an earthquake as a target for (Karnik, n.d.)prediction. Line 20 selects the features to use as the basis for prediction. Line 24 tells the program how much data should be training data and testing data. I have selected 20% of the data to be testing data and 80% as training data as generally majority of the data must be training data for the model to get an accurate understanding of the data. The random state feature is used here and throughout the code to make the results replicable.

The rest of the code is the standard code used for the machine learning models (Karnik, n.d.) along with code for hyperparameters. It can be seen in the following code that 'param_grid' is highlighted in yellow because not more than one machine learning model is to be used at a time in this code. This is done by putting a hash(#) to comment out the code of the other machine learning models, which is not done currently to show colour instead of grey code.

```
#KNN
knn_model = KNeighborsRegressor()
param_grid = {
    'n_neighbors': [1,3, 5, 7, 9, 11,15,20,50,100, 200, 300,500,],
    'weights': ['uniform', 'distance'],
    'leaf_size': [10, 20, 30, 40, 50, 60,70, 100],
    'p': [1, 2]}

#XGBoost
xgb_model = XGBRegressor()
xgb_model.fit(train_X,train_y)
param_grid = {
  'n_estimators': [50, 100, 200,500,1000],
 'max_depth': [ 5, 7, 10, None],
  'learning_rate': [0.001, 0.01, 0.05, 0.1, 0.3],
 'gamma': [0, 0.1, 0.2, 0.3],}

#Random forest
forest_model = RandomForestRegressor(random_state=1)
forest_model.fit(train_X, train_y)

param_grid = {
    'n_estimators': [100, 200, 300, 500],
   'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
   'min_samples_leaf': [1, 2, 4],}
```

Figure 2: KNN, XGBoost and Random Forest models

```
#Decision tree
tree_model = DecisionTreeRegressor(random_state=1)
tree_model.fit(train_X,train_y)
param_grid = {
    'max_depth': [3, 5, 7, 10, 15, None],
'min_samples_split': [2, 5, 10, 20, 50],
'min_samples_leaf': [1, 2, 5, 10,50],
'max_leaf_nodes': [None, 10, 20, 30],}

#Linear regression
linear_model = LinearRegression()
linear_model.fit(train_X,train_y)
pred = linear_model.predict(val_X)
```

Figure 3: Decision tree and linear regression models

```
grid_search = GridSearchCV(estimator=xgb_model,
                           param_grid=param_grid,
                           cv=5,
                           n_jobs=-1,
                           scoring='neg_mean_absolute_error')
grid_search.fit(train_X,train_y)
best_model = grid_search.best_estimator_
best_params = grid_search.best_params_
pred = best_model.predict(val_X)
```

Figure 4: GridSearchCV code

Hyperparameters

The tables below show the best hyperparameters for each model other than the linear regression model because this model does not use hyperparameters.

Table - 1 Decision Tree Hyperparameters

Max Depth	Max Leaf Nodes	Min samples leaf	Min samples split
10	None	5	20

Table-2 K-nearest Neighbour

Leaf	N neighbours	p	Weights
10	11	1	Distance

Table-3 Random Forest Hyperparameters

Random Forest			
Max Depth	Min samples leaf	Min samples split	N estimators
10	2	2	100

Table-4 Xgboost Hyperparameters

XGBoost			
Gamma	Learning Rate	Max Depth	N estimators
0.3	0.1	5	100

Evaluation Metrics Results

The table below highlights the best-performing machine learning model, which is the Random Forest model in all cases.

Table-5 Evaluation Metrics Results

	MAE	MSE	MAPE
Decision Tree	109.951	61285.789	9.825
XGBoost	103.653	63248.923	9.0453
K-nearest Neighbour	155.498	72206.399	16.248
Linear regression	136.165	56390.991	13.831
Random Forest	**100.333**	**55360.892**	**8.749**

Predicting Using Machine Learning Model

As the Random Forest model performed the best in all aspects in this case, it will be used to predict the significance of earthquakes. The following image displays the code for this.

```
model = RandomForestRegressor(n_estimators=100,
                              max_depth=10,
                              min_samples_leaf=2,
                              min_samples_split=2,
                              random_state=0)
model.fit(X, y)
input_data = {
    'magnitude': [6.5],
    'cdi': [7],
    'mmi': [5],
    'depth': [10],
    'latitude': [-15.8],
    'longitude': [167.0]
}
input_df = pd.DataFrame(input_data)
predicted_sig = model.predict(input_df)
print(f'Predicted sig value: {predicted_sig[0]}')
```

Figure 4: Prediction program code

The above code firstly sets the model to Random Forest regressor. Then, hyperparameters are set to those identified as the best ones by the evaluation metrics. The random state is set to zero to ensure that the results are replicable. Next, the standard code is used for predicting values based on the data. Here, the features (magnitude, cdi, mmi, depth, latitude, and longitude) are given random values to test the program. The last three lines of code are the standard code for predicting using data.

```
main
C:\Users\hp\PycharmProjects\Earthquake\venv\Sc
Predicted sig value: 661.1504309595634

Process finished with exit code 0
```

Figure 5: Result of inputted dummy values

The above image shows the predicted significance value for

the dummy values inputted.

CONCLUSION

To conclude, this research shows that machine learning models can be instrumental in predicting the significance of earthquakes. The ability to predict earthquake significance is crucial for disaster management and resource allocation. The outcome of this research can be used by international agencies such as the United Nations in order to compare the significance of earthquakes throughout the world and allocate resources appropriately between impacted countries utilizing the significance of the earthquakes as a significant basis, leading to an incredibly effective allocation of scarce resources using a reasoned basis rather an arbitrary one. This research can be extended by testing many other machine learning algorithms, including several other features, and testing more hyperparameters to make the result more accurate.

REFERENCES

1. Geeksforgeeks. (n.d.). How To Do Train Test Split Using Sklearn In Python. Retrieved June 15, 2024, from geeksforgeeks.org: https://www.geeksforgeeks.org/how-to-do-train-test-split-using-sklearn-in-python/
2. Kaggle. (n.d.). Earthquake Dataset. Retrieved from Kaggle: https://www.kaggle.com/datasets/warcoder/earthquake-dataset
3. Karnik, R. (n.d.). Retrieved from https://www.worldwidejournals.com/paripex/recent_issues_pdf/2024/September/investigating-the-effectiveness-of-different-machine-learning-algorithms-to-predict-the-prices-of-stocks_September_2024_0612784672_9305443.pdf
4. scikit-learn. (n.d.). mean_absolute_error. Retrieved June 16, 2024, from scikit-learn.org: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_absolute_error.html
5. scikit-learn. (n.d.). mean_absolute_percentage_error. Retrieved June 16, 2024, from scikit-learn.org: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_absolute_percentage_error.html
6. scikit-learn. (n.d.). mean_squared_error. Retrieved June 15, 2024, from scikit-learn.org: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_squared_error.html
7. Verma, A. (2023, February 10). GridSearchCV in scikit-learn: A Comprehensive Guide. Retrieved June 20, 2024, from dev.to: https://dev.to/anurag629/gridsearchcv-in-scikit-learn-a-comprehensive-guide-2a72