EARTHQUAKE PREDICTION MODEL USING PYTHON

Abstract:

It is common knowledge that disasters tend to repeat themselves after they strike one area. We can anticipate earthquakes using machine learning and other data-driven techniques like linear regression since there are more seismic monitoring stations because to the rising usage of technology. In this project, we'll use Python programming and machine learning to build a model that predicts earthquakes based on variables like longitude, latitude, duration, nation, depth, etc.

Key words: earthquake, machine learning, regression, seismic.

INTRODUCTION:

Many things happen in the modern world that humans cannot control. Natural catastrophes like tsunamis, tornadoes, floods, volcanic eruptions, etc. are examples of such events. Humans are powerless to avert an oncoming threat; all we can do is act quickly to minimize both human and financial damages. However, not all natural disasters are equally extensively researched and foreseen.

One of the most harmful and catastrophic natural disasters is an earthquake. First of all, they frequently take place with little prior notice, leaving no opportunity for preparation. The problem is further complicated by the fact that earthquakes frequently trigger other natural disasters like tsunamis and landslides.

However, the quick development of machine learning techniques and their successful application to a variety of issues suggest that these technologies could aid in producing precise forecasts.

LITERATURE SURVEY:

1. **Authors: Wenrui Li , Nakshatra, Nishita Narvekar, Nitisha Raut, Birsen Sirkeci, Jerry Gao**

   Title: Strong earthquake followed by aftershocks.

   We can detect location of these aftershocks by analysis of arrival time of P-waves and S-waves. Data collection from 16 earthquake stations in SAC file format, which contains time series data and is a waveform, used by authors to study trends in Pwaveand S-wave. Data is clipped followed by noise removal to only obtain needed waveform by means of triggering algorithm and filters. AR picker algorithm used to determine values of P-wave and S-wave arrival time which are treated as extracted feature. Waveform is then converted into ASCII format. Data is then fed to different machine learning models-SVM, Decision trees Random forest and linear regression for comparison purpose. Random Forest distinguishes between earthquake leading and nonearthquake leading data the best, with an accuracy of 90. Use of triangulation technique to calculate epicentre, predict arrival time of P-wave and S-wave and the difference between the two arrivals.[2]

2. **Authors:Khawaja Muhammad Asim, Adnan Idris, Francisco Martinez-Alvarez, Talat Iqbal.**

   Title:Earthquake prediction using random forest Regression model.

   carried out prediction of earthquake for Hindu-Kush region where small to medium earthquakes hit regularly, in accordance with tree based ensemble classifiers like rotboost, random forest and rotation forest. They employ earthquake data-set, and convert magnitude into binary classes, hence adapting concept of binary classification. A new combination of features based on 3 factors- Gutenberg-Richter relationship, seismic rate changes and distribution of fore-shock frequency. Highlighting factor is calculation of 51 seismic feature using suitable procedures and techniques. Since all the models performed exceptionally well, we can conclude the strategy of calculating 51 features was very effective. Rotation forest gives an accuracy of 95.9% and titles itself the best among rest models.[3] The useful insights for us come in the fact that for every region on this earth, a prediction model needs to be deployed however there is no prediction of when and of what magnitude will an earthquake occur of.

PROBLEM DEFINITION:

The problem is to develop an earthquake prediction model using a Kaggle dataset. The objective is to explore and understand the key features of earthquake data, visualize the data on a world map for a global overview, split the data for training and testing, and build a neural network model to predict earthquake magnitudes based on the given features.

DATASETS:

When a specific field is researched in terms of machine learning, the first question is where to find data. As for earthquake datasets, various organizations and research institutions are constantly monitoring seismic activity of all over the world. The structure of earthquake datasets are usually presented in the form of a table, each record of which corresponds to a certain seismic event. The sets of attributes are different for data published in different datasets.

Dataset Link: **https://www.kaggle.com/datasets/usgs/earthquake-database**

DATASET 1: earthquake1.csv:

Contains all the recorded earthquakes in Turkey between 1910-2017 larger than 3.5 intensity. The parameters are:

• Id: order number of the earthquake

• Date: earthquake occurrence date

• Time: time of the earthquake

• Lat: latitude of the earthquake epicenter

• Long: longitude of the earthquake epicenter

• Country: country of the earthquake epicenter

• City: province of the occurred earthquake

• Area: region of the occurred earthquake

• Direction: direction of the earthquake signal

• Dist: distance of direction in km

• Depth: depth of the occurred earthquake (distance from the surface)

• Xm: the largest of the given magnitude values

• Md: magnitude depending on time

• Richter: Richter magnitude or the local magnitude (ML)

• Mw: moment magnitude

• Ms: surface wave magnitude

• Mb: body wave magnitude

DATASET 2: earthquake2.csv:

Contains all Significant Earthquakes that occurred in the period of 1965-2016. The parameters are:

• Time: Time of the earthquake

• Latitude: latitude of the earthquake epicenter

• Longitude: longitude of the earthquake epicenter

• Depth: depth of the occurred earthquake (distance from the surface)

• mag: size of the earthquake

• magType: type of magnitude occurred (ml or ms)

• Nst: Number of seismic stations

• Gap: Region along an active fault where stress is accumulating because no earthquakes have occurred there recently

• D-min: Horizontal distance from the epicenter to the nearest station

• RMS: In general, the smaller this number, the more reliable is the calculated depth of the earthquake

• Id: order number of the earthquake

• Updated: updation in sequence of earthquake

• Place: place where earthquake occurred

• Type: type of earthquake occurred

• Horizontal error: error in range

• Depth error: error occurred in calculation of depth of earthquake

• magError: error occurred in calculation of magnitude of earthquake

• magNst: error occurred in calculation of number of seismic stations

• Status: current status of earthquake

• Location source: source of location where earthquake occurred • magSource: source where magnitude is occurred

METHODS:

To begin with, the dataset is pre-processed to make it suitable for modelling. Second, the dataset is partitioned into training and testing data-set and third, different models are constructed and fitted to the training data-set. The accuracy of the system is obtained by testing the system using the testing data. In this study, we explore, evaluate and analyze dataset characteristics using different machine learning algorithms like linear regression, decision tree and KNN.

A. Linear Regression:

Linear Regression is a supervised Machine Learning model which assumes a linear relationship between the input variables (x) and the single output variable (y). For linear regression, we will use only numerical data (float). This form of analysis estimates the coefficients of the linear equation, involving one or more independent variables that best predict the value of the dependent variable. For Earthquake prediction, we use various input variables (X) such as earthquake occurrence date, surface wave magnitude, body wave magnitude, Richter magnitude or the local magnitude (ML) for the first dataset to predict the output variable (Y) - Xm, the largest of the given magnitude values. For the 2nd dataset, we predict the value of magnitude (Y) using variables such as the number of seismic stations, type of magnitude that occurred (ml or ms), gap etc.

B. Decision Tree:

Decision-tree algorithm falls under the category of supervised learning algorithms. It works for both continuous as well as categorical output variables. Decision tree regression observes the features of an object and trains a model in the structure of a tree to predict data in the future to produce meaningful continuous output. The goal of a decision tree is to learn a model that predicts the value of the magnitude of the earthquake by learning simple decision rules inferred from the data features of the given dataset. For example, if the Richter value > 0.49, then XM value will be approx. 0.07. For our decision tree, the first node is the column with the highest information gain i.e., X[9] which is md - magnitude depending on time.

C. K-Nearest Neighbour

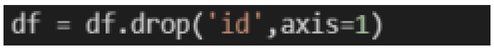
K nearest neighbours is a simple algorithm that uses 'feature similarity' to predict the values of any new data points. This means that the new point is assigned a value based on how closely it resembles the points in the training set. the KNN algorithm would find the k-nearest data points to a new data point and predict the continuous output based on the average of the output values of the nearest neighbours. For earthquake prediction, we use k-Nearest Neighbours Regressor. Even though we had multiple features, the importance of each feature in the Euclidean distance calculation can vary. To handle this, we performed normalization before calculating the distance, so that all features are on the same scale and have an equal contribution to the distance.

PREDICTIONS:

Algorithm:

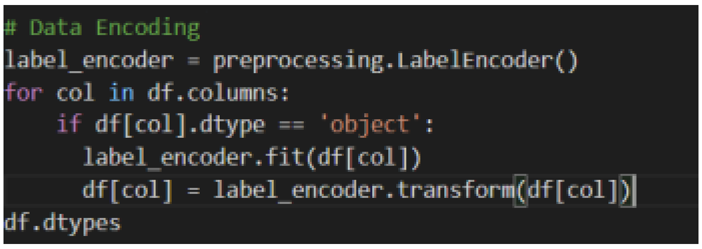
- **Input data-set and load libraries.**
- **Data Pre-processing**.
  For both the datasets the following pre-processing steps were taken:

• First, we dropped the 'id' column since it is not useful for prediction.



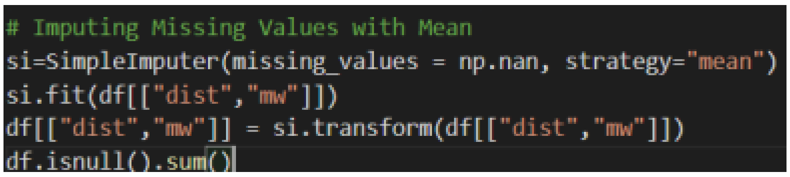
```
df = df.drop('id',axis=1)
```

• Next we converted categorical variables into numerical values so that it could be easily fitted to a machine learning model using a label encoder.



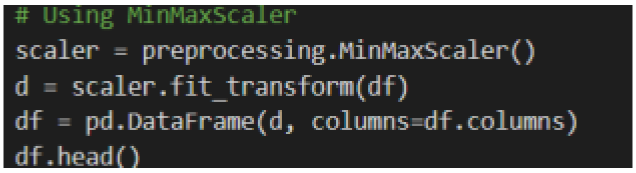
```
# Data Encoding
label_encoder = preprocessing.LabelEncoder()
for col in df.columns:
    if df[col].dtype == 'object':
      label_encoder.fit(df[col])
      df[col] = label_encoder.transform(df[col])
df.dtypes
```

• Imputation with SimpleImputer which replaced the missing data with mean was done to handle null values.



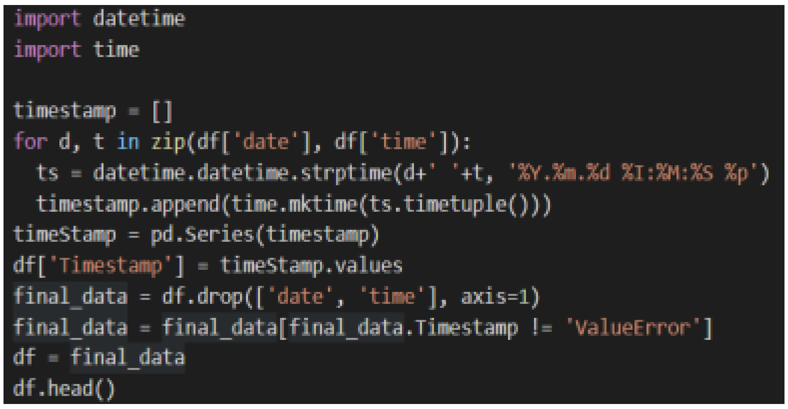
```
# Imputing Missing Values with Mean
si=SimpleImputer(missing_values = np.nan, strategy="mean")
si.fit(df[["dist","mw"]])
df[["dist","mw"]] = si.transform(df[["dist","mw"]])
df.isnull().sum()
```

• Normalization of the data was done using MinMax Scaler. The Min-Max scaling method helps the dataset to shift and rescale the values of their attributes, so they end up ranging between 0 and 1.



```
# Using MinMaxScaler
scaler = preprocessing.MinMaxScaler()
d = scaler.fit_transform(df)
df = pd.DataFrame(d, columns=df.columns)
df.head()
```

• Dataset1 had two columns ‘date’ and ‘time’. It was converted to another column ‘timestamp’ using packages such as ‘datetime’ and ‘time’.

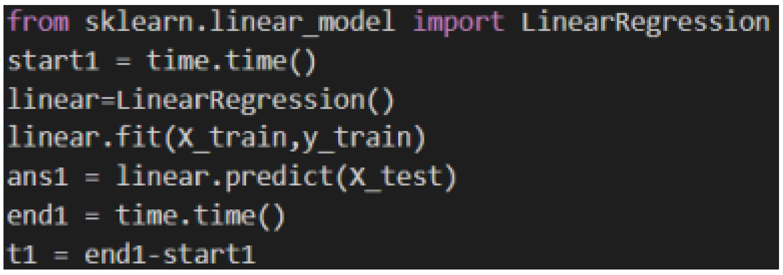


```
import datetime
import time

timestamp = []
for d, t in zip(df['date'], df['time']):
  ts = datetime.datetime.strptime(d+' '+t, '%Y.%m.%d %I:%M:%S %p')
  timestamp.append(time.mktime(ts.timetuple()))
timeStamp = pd.Series(timestamp)
df['Timestamp'] = timeStamp.values
final_data = df.drop(['date', 'time'], axis=1)
final_data = final_data[final_data.Timestamp != 'ValueError']
df = final_data
df.head()
```

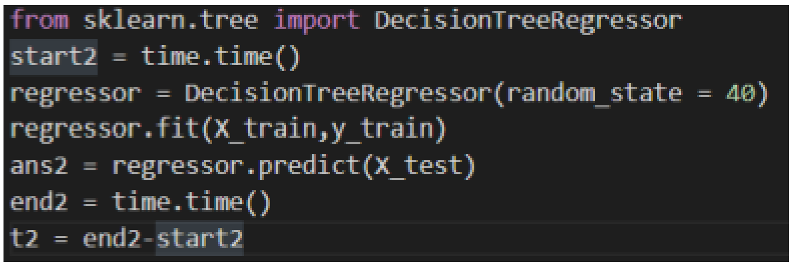
- **Model Building.**

  1.Linear regression:



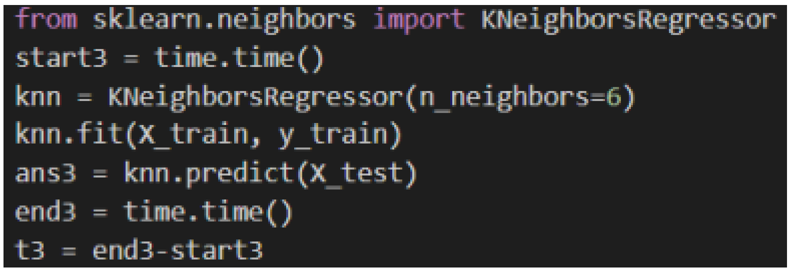
```
from sklearn.linear_model import LinearRegression
start1 = time.time()
linear=LinearRegression()
linear.fit(X_train,y_train)
ans1 = linear.predict(X_test)
end1 = time.time()
t1 = end1-start1
```

  2.Decision Tree:



```
from sklearn.tree import DecisionTreeRegressor
start2 = time.time()
regressor = DecisionTreeRegressor(random_state = 40)
regressor.fit(X_train,y_train)
ans2 = regressor.predict(X_test)
end2 = time.time()
t2 = end2-start2
```

  3.K-Nearest-Neighbour:



```
from sklearn.neighbors import KNeighborsRegressor
start3 = time.time()
knn = KNeighborsRegressor(n_neighbors=6)
knn.fit(X_train, y_train)
ans3 = knn.predict(X_test)
end3 = time.time()
t3 = end3-start3
```

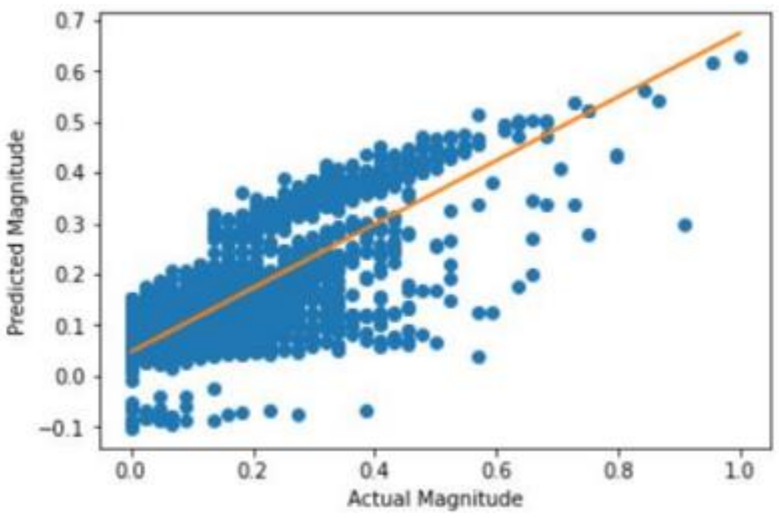
- Making Predictions.

RESULTS:

DATASET 1: earthquake1.csv

1.Linear Regression Model

- Accuracy of Linear Regression model is: 0.63134131503029
- Mean Absolute Error: 0.05878246463205686
- Mean Squared Error: 0.00625827169726636
- Root Mean Squared Error: 0.07910923901331854

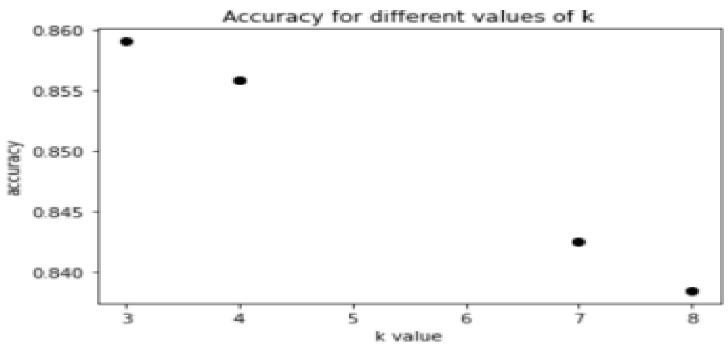


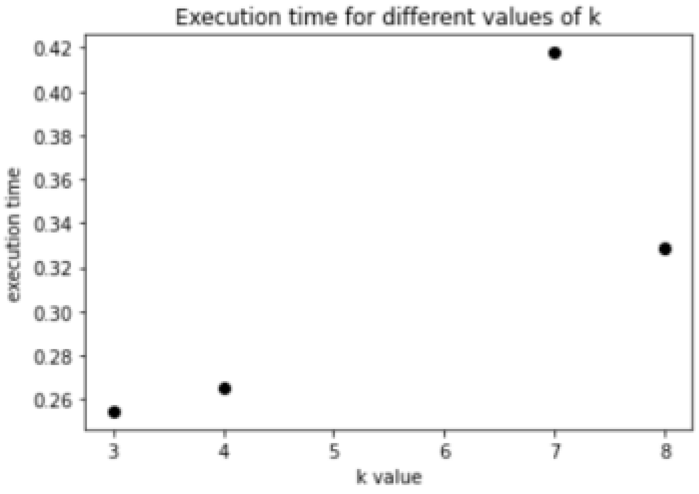
2.Decision Tree Model

- Accuracy of Decision Tree model is: 0.9932960893884235.
- Mean Absolute Error: 0.0006909999621372331
- Mean Squared Error: 0.00011380416561969702
- Root Mean Squared Error: 0.010667903525046383

3. KNN Model

- Accuracy of KNN model is: 0.8457466919393031
- Mean Absolute Error: 0.03305598677318794
- Mean Squared Error: 0.002618571462992348
- Root Mean Squared Error: 0.051171979275696854

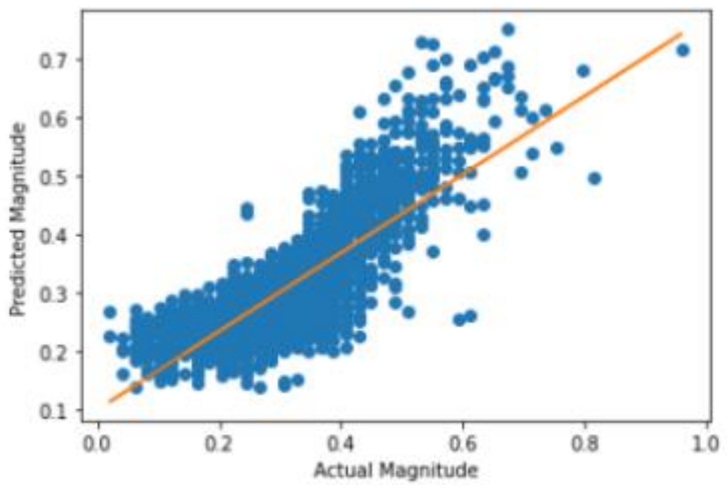




DATASET 2: earthquake2.csv:

1.Linear Regression Model

- Accuracy of Linear Regression model is: 0.6520026760336074
- Mean Absolute Error: 0.0447298826759214
- Mean Squared Error: 0.0034475163509273773
- Root Mean Squared Error: 0.058715554590988726

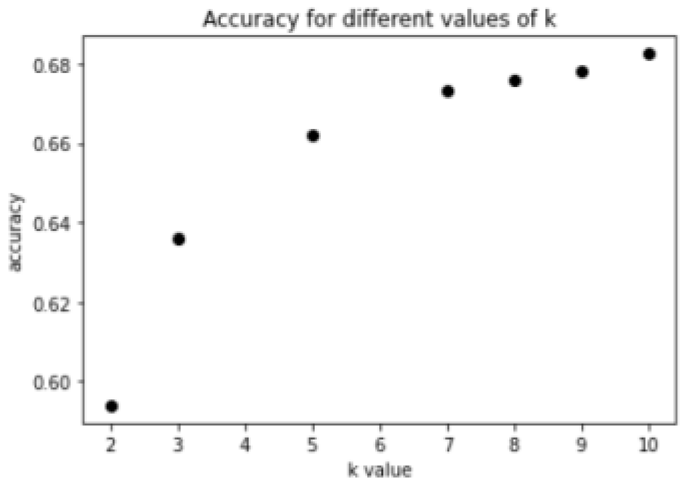


2.Decision Tree Model

- Accuracy of Decision Tree model is: 0.5488812291089398
- Mean Absolute Error: 0.049579341940857974
- Mean Squared Error: 0.004469112926303383
- Root Mean Squared Error: 0.06685142426533172

3.KNN Model

- Accuracy of KNN model is: 0.6828276373795497
- Mean Absolute Error: 0.04780785783701236
- Mean Squared Error: 0.004023436946623714
- Root Mean Squared Error: 0.06343056791976337



CONCLUSION:

The development of a Machine Learning-Based Earthquake Prediction Model represents a critical step towards enhancing disaster preparedness and mitigation efforts. By leveraging historical seismic data and applying machine learning techniques,provides actionable insights into earthquake risk, aiding both authorities and communities in identifying vulnerable regions. With a focus on data quality, model accuracy, and user accessibility, this project aims to contribute to improved safety and resilience in earthquake-prone areas.

APPENDIX:

NumPy: NumPy is a Python library for working with large, multi-dimensional arrays and matrices of numerical data, as well as a large collection of high-level mathematical functions to operate on these arrays.

Pandas: Pandas is a library for data manipulation and analysis in Python, providing efficient data structures and powerful manipulation capabilities.

Matplotlib: Matplotlib is a plotting library for creating static, animated, and interactive visualizations in Python.

Seaborn: Seaborn is a data visualization library for creating attractive and informative statistical graphics in Python, built on top of Matplotlib.

Scikit-learn: Scikit-learn is a machine learning library for Python, providing simple and efficient tools for data mining and data analysis.

REFERENCES:

1. C. Li and X. Liu, ”An improved PSO-BP neural network and its application to earthquake prediction,” 2016 Chinese Control and Decision Conference (CCDC), Yinchuan, 2016, pp. 3434-3438.

2. W. Li, N. Narvekar, N. Nakshatra, N. Raut, B. Sirkeci and J. Gao, “Seismic Data Classification Using Machine Learning,” 2018 IEEE Fourth International Conference on Big Data Computing Service and Applications (BigDataService), Bamberg, 2018, pp. 56-63.

3. K. M. Asim, A. Idris, F. Mart´ınez-A´ lvarez and T. Iqbal, “Short Term Earthquake Prediction in Hindukush Region Using Tree Based Ensemble Learning,” 2016 International Conference on Frontiers of Information Technology (FIT), Islamabad, 2016, pp. 365-370.

4. Kumari, G. T. Prasanna. “A Study Of Bagging And Boosting Approaches To Develop Meta-Classifier.”, Engineering Science and Technology: An International Journal, Vol.2, 2012, pp. 850-855.

5. Ant´onio E Ruano, G. Madureira, Ozias Barros, Hamid R. Khosravani, Maria G. Ruano, Pedro M. Ferreira. ”A Support Vector Machine Seismic Detector for Early-Warning Applications”, IFAC Proceedings Volumes, 2013, pp. 400-405.

6. Nick Minaie. “A Beginner’s Guide to Selecting Machine Learning Predictive Models in Python”, Towardsdatascience, Medium, 16 July 2019.

7. U. Pasupulety, A. Abdullah Anees, S. Anmol and B. R. Mohan, “Predicting Stock Prices using Ensemble Learning and Sentiment Analysis,” 2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE), Sardinia, Italy, 2019, pp. 215-222.

8. The Expert Team. “What is Machine Learning? A definition”, Expertsystem, 7 March 2017.

9. Flach, Peter. Machine Learning: the Art and Science of Algorithms That Make Sense of Data. Cambridge University Press, 2017, pp. 331-333