A Comparison of Various Forecasting Models in Predicting Indian Rainfall from Spatio-temporal Data

Sanchari Biswas
sanchari.biswas@temple.edu

April, 2022

Abstract

The country of India has been fraught with hydrological draughts. The prediction of Indian rainfall is not only scientifically challenging but also crucial owing to the planning and execution strategies agriculturally and otherwise. There exist a number of works on this prediction, some using Artificial Neural Networks (ANNs) and Multiple Linear Regression (MLR) models. In this work [1], we decide upon ten of the prevalent data mining models: K-Nearest Neighbor (which is our baseline model), Logistic Regression, Decision Tree, Support Vector Machines, Random Forest, Extra Trees, Gradient Boosting, XGBoost, LightGBM, and CatBoost. We then analyze the performances of these models based on precision scores, recall scores, and F1 measures. We then pick the top two performers and hypertune the parameters for both of them. Now, gaining a clearer idea, we take a theoretical and operational approach to analyze both the models and, in conclusion, decide on the optimum model based on performance, overhead, and other such factors. In our work, we have achieved F1 measures of 0.95 for LightGBM and XGBoost before hyperparameter tuning. Our baseline K-Nearest Neighbor has an F1 measure of 0.72. After hyperparameter tuning, we get an F1 measure of 0.99 for XGBoost and 0.97 for LightGBM. However, XGBoost takes approximately ten times more time to train than LightGBM and also needs much more resources. Based on the other important criteria, such as latency and overhead, we choose LightGBM as our model of choice, since in those aspects, it bypasses XGBoost by far.

1 Introduction

India is a country ranking second in the terms of population worldwide. The Indian rainfall (or monsoon, as the terms will be used interchangeably) is known for its uneven distribution of rainfall in space and time [2]. This uneven distribution in often cases result either in a deficit (drought) or in a surfeit (flood). Both of the former are hydrological hazards and cause excessive damage and financial losses to crops, property, and the generation of hydro-electricity and, in turn, impose strain on the Indian economy.

A particular year is termed as an all-India drought year if the average seasonal rainfall anomaly in the country as a whole is less than -10% of its long period average [3]. Based on the former criterion, 17% of the years during the period of 1901 through 2010 were drought years [3]. Despite technological advancements and improved drought mitigation measures, droughts cause adverse effect on the economy of India. The severe drought of 2002 had an adverse effect on the Indian economy by lowering the Gross Domestic Product (GDP) of the country by almost 1% [4]. To improve drought mitigation and preparedness, with our present knowledge about the spatial and temporal variability weather information, we need to construct a rainfall prediction model.

2 Related Work

Historically, many studies have been performed by utilizing ground-based measurements for modeling and monitoring meteorological droughts ([5],[6], [7], [8]).

The study by Barua et al. [8] develops a nonlinear aggregated drought index (NADI) in order to classify the drought condition of a catchment considering all the significant hydrometeorological variables, and then an artificial neural network(ANN)—based drought forecasting approach is developed using the time series of the NADI to forecast the NADI values.

Another study by Cancelliere et al. [9] discusses two methods. The first method uses the auto-covariance matrix of Standard Precipitation Index (SPI) values, which are analytically derived, as a function of the statistics of the underlying monthly precipitation process, to compute the transition probabilities from a current drought condition to another in the future. The other method uses SPI to forecast at a generic time horizon M, in terms of conditional expectation, as a function of past values of monthly precipitation. Forecasting accuracy is estimated through an expression of the Mean Square Error, which allows one to derive confidence intervals of prediction.

Sahai et al. [10] introduces models for forecasting summer monsoon rainfall on monthly and seasonal time scales, using Artificial Neural Network (ANN) techniques. They have used ANNs in a time series approach with the presumption that Indian summer monsoon rainfall (ISMR) is not only related to previous seasonal mean ISMR values but also with the previous monthly mean (June, July, August and September) rainfall values.

Another paper [11] performs rainfall prediction model with the use of empirical statistical technique, MPR. Khandelwal et al. [12] also presented an MLR equation to predict rainfall using four different climatic factors for Jaipur city, Rajasthan, India. In selecting these factors, the author used Pearson correlation coefficient and then predicted the drought possibility.

3 Methodology

The flow diagram of our approach is shown in the figure 1. We start our approach with the collection of our data from the Weather dataset collected from the WeatherAPI. We then perform the preprocessing our data. Then, we divide our data into test and training sets and construct our models. Once we have the performance results of our models, we hypertune the parameters of the top two performing models. In conclusion, we decide which model we choose and why.

3.1 Data Used

The dataset I am using is 'Indian Weather and Astronomy Data' [13]. This dataset consists of three files with Weather, Astronomy, and Location Data. These data are collected from the WeatherAPI [14]. Of the three files, I have worked with the Weather and the Location data. The weather data consists of 34 attributes (Date and time, Temperature in celsius, Temperature in fahrenheit, Weather condition, Maximum wind speed in miles per hour, Wind direction, to name a few) and 29568 distinct values. The location data consists of 8 attributes (name of state, city, country, latitude, longitude, etc) and 1232 distinct values.

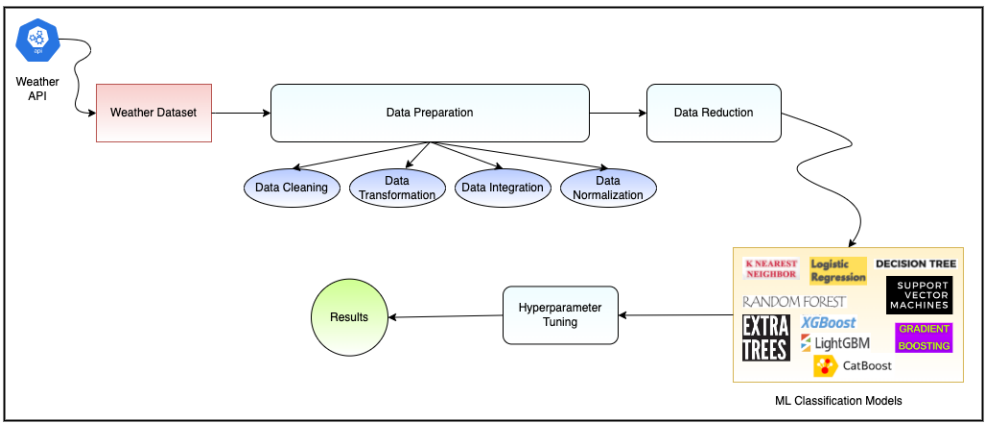

Figure 1: Workflow Block Diagram

The weather dataset is a clean data with a usability of 9.41 and consists of spatio-temporal data.

Spatio-Temporal Data: Space and time are ubiquitous aspects of observation in various domains, such as climate sciences, neurosciences, epidemiology, transportation, and Earth sciences, that are rapidly being transformed by the huge amounts of incoming data [15]. Because of the real-world processes being examined under these domains being inherently spatio-temporal in nature, myriad varieties of data collection methodologies have been devised to record the spatial and temporal information of the features in the data, and hence the data being referred to as spatio-temporal (ST) data [15].

3.2 Data Preprocessing

The input data to a data mining model must be supplied in the amount, structure and format that suits the task accurately. However, in reality, databases are often severely impacted by negative factors such the presence of noise, inconsistent and superfluous data and huge sizes of data. Thus, low-quality data will lead to low-quality performance. Hence, we need to preprocess the data according to our model. The preprocessing of any data involves two broad steps: Data Preparation and Data Reduction [16].

Data Preparation: Data Preparation deals with converting prior useless data to new data such as to fit the data mining model. If the data is not processed, the model might not be able to recieve it or operate on it. It might also erroneously report mistakes during runtime. It might also happen that the model will accept and process the data but the results obtained will not be proper and accurate. The process of data preparation can be further subdivided into Data Cleaning, Data Transformation, Data Integration, Data Normalization, Missing Data Imputation, and Noise Identification [16].

Data Cleaning - Data cleaning or data cleansing involves operations that correct or filter out bad data and reduce the unnecessary detail in data. The dataset we have used consists of clean data without any missing values or erroneous inputs. Thus, our preprocessing did not include data cleaning.

Data Transformation - In this step, the data is converted or consolidated such that the model output may be more efficient. Subtasks of data transformation include smoothing, feature construction, aggregation, normalization, discretization and generalization. In our dataset, we have come categorical attributes such as wind direction (expressed

as SE, NNE, etc.) which we encode in this step.

Data Integration - This step comprises of the merging of data from multiple data stores. Operations within this step include the identification and unification of variables and domains, the analysis of attribute correlation, the duplication of tuples and the detection of conflicts in data values of different sources. Our work consists of the data from a single dataset and hence no integration required.

Data Normalization - The measurement unit used in the attributes can affect the effectiveness of the model. All the attributes should be expressed in the same measurement units and should use a common scale or range. Normalizing the pertinent data gives all the attributes equal weightage and is particularly useful in statistical learning methods. We also perform normalization among our data using feature scaling. This is because different attributes in our data have different ranges and different rates of growth and deviations.

We do not perform any missing data imputation and noise identification as explained previously.

Data Reduction: Data reduction deals with the set of techniques that provide a compressed representation of the original data. In our dataset, we deal with data reduction in three ways. We drop some of the features that essentially carry the same data but have different units of measurement (eg. wind_mph and wind_kph). We also drop the attributes with almost no correlation with our target column, will_it_rain (eg. will_it_snow). We also extract the important features from all of them using Recursive Feature Elimination.

3.3 Building Models

Predictive Modeling: Predictive modeling can be defined as a mathematical process that is used to predict future events or outcomes by analyzing the patterns from input data. We use our work to demonstrate an example of predictive modeling. We aim to predict the rainfall in India from a spatio-temporal dataset, and hence will work with a model, with an algorithm, that helps us with the prediction with more accuracy than randomization. In our project, we aim at a comparison between ten models built with the most used algorithms in data mining. We will discuss them one by one briefly in the next sections.

K-Nearest Neighbor: K-Nearest Neighbor algorithm is our baseline. A baseline algorithm is a simple algorithm that performs better than an entirely random approach and is used to establish minimum expected performance on a dataset. A baseline provides a point of comparison for the more advanced algorithms that we aim to evaluate later. K-nearest neighbors algorithm is a lazy-learning non-parametric supervised learning algorithm that estimates the likelihood of a data point being a member of one collection or another based on which collection the data points nearest to it belong to [17].

Logistic Regression: Logistic Regression, despite its name, is a classification model rather than a regression model. It is a simple efficient model for both binary and linear classification problems. It is very easy to realize and achieves very good performance with linearly separable classes [18].

Decision Tree: Decision Tree algorithm is a supervised learning algorithm. Although unlike other supervised learning algorithms, decision tree algorithms can also be used for solving regression and classification problems. Decision tree algorithm uses multiple algorithms in the decision of splitting a node into two or more sub-nodes. The splitting into sub-nodes increases the homogeneity of resultant sub-nodes.

Support Vector Machines: A Support Vector Machine (SVM) is an algorithm that learns by example to assign labels to objects [19]. It is a mathematical entity for maximizing a particular mathematical function with respect to a given collection of data.

Random Forest: The Random Forest algorithm is a general-purpose classification and regression algorithm that combines several randomized decision trees and aggregates their predictions. This algorithm has shown excellent performance in settings where the number of variables is much larger than the number of observations [20].

Extra Trees: Extra Trees or Extremely Randomized Trees Classifier is a type of ensemble learning technique that aggregates the results of multiple decorrelated decision trees collected in a forest to output its classification result. In concept, it is very similar to a Random Forest Classifier but differs from it in the process of constructing the decision trees in the forest.

Gradient Boosting: Gradient Boosting is a powerful machine-learning technique that consecutively fits new models to provide a more accurate estimate of the response variable. The main idea behind this algorithm is the construction of the new base-learners, which aim to be maximally correlated to the negative gradient of the loss function that is associated with the entire ensemble [21].

XGBoost: XGBoost or eXtreme Gradient Boosting package is an efficient and scalable implementation of the gradient boosting framework. The package includes efficient linear model solver and tree learning algorithm and also supports various objective functions, such as regression, classification and ranking [22].

LightGBM: LightGBM is a gradient boosting algorithm with Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) [23]. It is based on decision tree algorithms and used for ranking, classification and other machine learning tasks.

CatBoost: CatBoost is a gradient boosting algorithm that functions by building each new tree to approximate the gradients of the current model [24]. CatBoost uses oblivious trees as base predictors where the same splitting criterion is used across the entire level of the tree making them balanced and less prone to overfitting [24].

4 Results

We will divide the results section into three subsections. In the first sub-section, we discuss the results from preprocessing our data, how we gather our relevant attributes and perform our data reduction. We will then, in the next sub-section, discuss how we divide our data into training and test sets and plot our performances from the models. We then discuss hypertuning the parameters of two of our models and an aggregate of the final comparisons drawn.

4.1 Data from Preprocessing

At the start, we extract the location data from our dataset and plot it on the map of India. This gives us a diagrammatic representation of the various extremities of the country and the cities from where we are collecting our spatio-temporal data.

We start by dropping one instance from all the attributes that carry the same data but in different units, such as temp_f and temp_c. We then visualize the various continuous distributions, such as wind_degree, pressure_mb, and humidity among more. We can also

visualize the discrete variables such as wind_dir. We then drop irrelevant attribute such as the will_it_snow attribute. We can also see the correlation heatmap generated by plotting the continuous variables against each other.

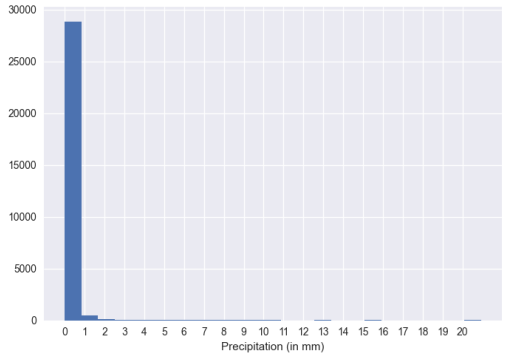

Figure 2: Continuous Variables against Precipitation

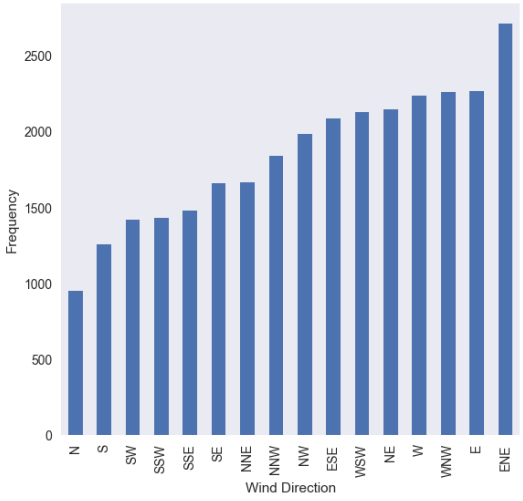

Figure 3: Wind Direction by Frequency

From the heatmap as shown, we can drop the attributes with a correlation of near zero with our target variable, will_it_rain. We then encode the categorical variables and perform feature scaling, hence to obtain our final dataset to imlement our models upon.

4.2 Result from the Models

We now have our final dataset and we divide this into two sets, one for training data and the other for test data. 80% of our data is used for training and the rest 20% for testing purposes. In the figure 5, we can see the true positive, true negative, false positive, and false negative values for the K-Nearest Neighbor model which is our baseline. The other

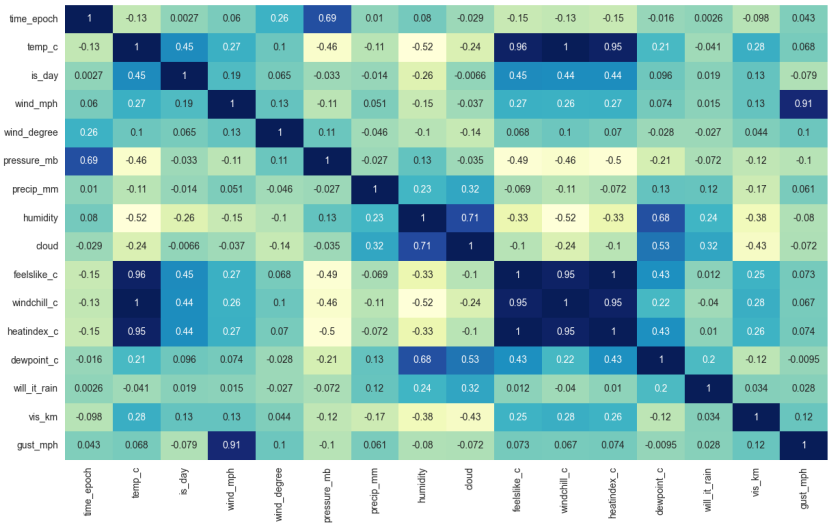

Figure 4: Correlation Heatmap

confusion matrices show the values respectively for the best two (LightGBM and XGBoost) and the worst (Logistic Regression) models from our results in table 1.

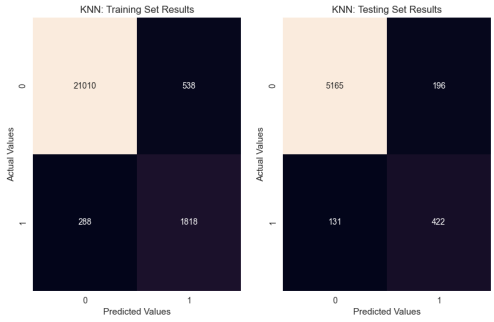

Figure 5: KNN: Confusion Matrix

Figure 6: LR: Confusion Matrix

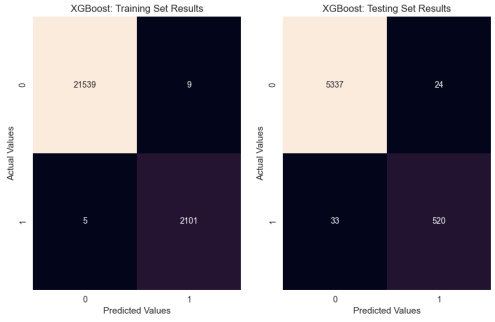

Figure 7: XGB: Confusion Matrix

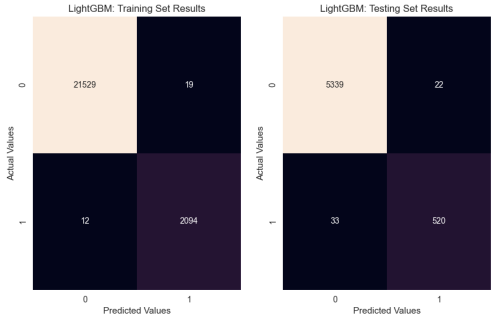

Figure 8: LGBM: Confusion Matrix

We can now sum up the recall score, precision score, and F measures in the following table 1, where we arrange the scores in the descending order of the F measure.

	Train_Recall	Test_Recall	Train_Precision	Test_Precision	F1_Measure
LightGBM	0.99	0.94	0.99	0.96	0.95
XGBoost	1.0	0.94	1.0	0.96	0.95
CatBoost	0.99	0.94	0.98	0.95	0.93
RandomForest	1.0	0.94	1.0	0.88	0.89
GBM	0.89	0.89	0.85	0.87	0.88
DecisionTree	1.0	0.88	1.0	0.87	0.87
ExtraTrees	1.0	0.92	1.0	0.83	0.87
SVM	0.89	0.89	0.69	0.71	0.79
KNN	0.86	0.76	0.77	0.68	0.72
LogisticRegression	0.2	0.18	0.37	0.36	0.24

Table 1: Comparison between models before hypertuning parameters

4.3 Hypertuning the Parameters

Once we have the results from table 1, we choose to hypertune the top two resulting models. We see that our top two models are LightGBM and XGBoost. Hyperparameter tuning or hypertuning the parameters is termed as choosing a set of optimal hyperparameters for a learning algorithm. A hyperparameter is a model argument whose value is set before the learning process begins. In the ongoing process, we provide the model iteratively with a range of values for the parameter and aim to chose the best fit with the best performance. In hypertuning the two models, we choose the parameters(range) as num_leaves (20-120), max_depth (10-60), learning_rate (0.1-2.0), n_estimators (100-350), reg_alpha (0.0-2.0), reg_lambda (0.0-2.0), colsample_bytree (0.0-1.0). We can see the results in the table 2.

	Train_Recall	Test_Recall	Train_Precision	Test_Precision	F1_Measure
XGB_Tuned	1.0	0.98	1.0	1.0	0.99
LGBM_Tuned	1.0	0.96	1.0	0.98	0.97

Table 2: Comparison between models with hyperparameter tuing

5 Discussion

From our initial results, we observe that XGBoost and LightGBM perform the best, with an F measure of 0.99% and 0.97% respectively. GBMs generally give a prediction model in the form of an ensemble of weak prediction models. When a decision tree is the weak learner, the resulting algorithm is called gradient-boosted tree, which usually outperform RandomForest, as we can see. The Logistic Regression model performs the worst. The major limitation of Logistic Regression is the assumption of linearity between the dependent variable and the independent variables. Linearly separable data is rarely found in real-world scenarios. We also observe RandomForest, DecisionTree, and ExtraTrees are the only three to achieve a 1.0 Recall Score in the Test Data, which is due to an overfitted model. After hyperparameter tuning, we have two best contenders: XGBoost and LIGHTGBM. We then look deeper into both of these models in terms of operation and functionality. We

observe that in XGBoost, trees grow depth-wise whereas in LightGBM, trees grow leaf-wise. LightGBM training happens almost ten times faster than XGBoost training. XGBoost needs a lot of resources to train on large amounts of data. LightGBM is lightweight and can be used even on modest hardware. Gradient Boosting uses differentiable function losses from the weak learners to generalize. At each boosting stage, the learners are used to minimize the loss function, given the current model. A downside to LightGBM is that it has less documentation and a narrow user base. But that is changing fast. As we can observe, when tuning the hyperparameters, XGBoost outperforms LightGBM, by a small margin. Going by the above observations, we are inclined on using LightGBM to predict Indian rainfall from our spatio-temporal data.

6 Acknowledgements

Professor Zoran Obradovic is thanked for immense help throughout the project work, Marija Stanojevic for her assistance and generous input regarding the subject matter, and my father, a meteorologist, for raising my awareness regarding this pressing concern.

References

[1] S. Biswas, "Indian Rainfall Prediction Comparison," 02 2022. [Online]. Available: https://github.com/sanchari1992/DataMiningProject

[2] M. S. Tyalagadi, A. Gadgil, and G. Krishnakumar, "Monsoonal droughts in india–a recent assessment," *Papers on global change*, vol. 22, 2015.

[3] K. N. Kumar, M. Rajeevan, D. Pai, A. Srivastava, and B. Preethi, "On the observed variability of monsoon droughts over india," *Weather and Climate Extremes*, vol. 1, pp. 42–50, 2013.

[4] S. Gadgil, P. Vinayachandran, and P. Francis, "Droughts of the indian summer monsoon: Role of clouds over the indian ocean," *Current Science*, pp. 1713–1719, 2003.

[5] H. Shen and G. Tabios III, "Modeling of precipitation-based drought characteristics over california," Tech. Rep., 1996.

[6] A. Belayneh, J. Adamowski, B. Khalil, and B. Ozga-Zielinski, "Long-term spi drought forecasting in the awash river basin in ethiopia using wavelet neural network and wavelet support vector regression models," *Journal of Hydrology*, vol. 508, pp. 418–429, 2014.

[7] J. F. Santos, I. Pulido-Calvo, and M. M. Portela, "Spatial and temporal variability of droughts in portugal," *Water Resources Research*, vol. 46, no. 3, 2010.

[8] S. Barua, A. Ng, and B. Perera, "Artificial neural network–based drought forecasting using a nonlinear aggregated drought index," *Journal of Hydrologic Engineering*, vol. 17, no. 12, pp. 1408–1413, 2012.

[9] A. Cancelliere, G. D. Mauro, B. Bonaccorso, and G. Rossi, "Drought forecasting using the standardized precipitation index," *Water resources management*, vol. 21, no. 5, pp. 801–819, 2007.

[10] A. Sahai, M. Soman, and V. Satyan, "All india summer monsoon rainfall prediction using an artificial neural network," *Climate dynamics*, vol. 16, no. 4, pp. 291–302, 2000.

[11] N. Sethi and K. Garg, "Exploiting data mining technique for rainfall prediction," *International Journal of Computer Science and Information Technologies*, vol. 5, no. 3, pp. 3982–3984, 2014.

[12] N. Khandelwal and R. Davey, "Climatic assessment of rajasthan's region for drought with concern of data mining techniques," *Concern*, vol. 2, no. 5, 2012.

[13] C. Manchanda, "Indian weather and astronomy data," 10 2021.

[14] "Free Weather API - WeatherAPI.com." [Online]. Available: https://www.weatherapi.com/

[15] G. Atluri, A. Karpatne, and V. Kumar, "Spatio-temporal data mining: A survey of problems and methods," *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, pp. 1–41, 2018.

[16] S. García, J. Luengo, and F. Herrera, *Data preprocessing in data mining*. Springer, 2015, vol. 72.

[17] A. Joby, "What Is K-Nearest Neighbor? An ML Algorithm to Classify Data," 07 2021. [Online]. Available: https://learn.g2.com/k-nearest-neighbor

[18] A. Subasi, *Practical Machine Learning for Data Analysis Using Python*. Academic Press, 2020.

[19] W. S. Noble, "What is a support vector machine?" *Nature biotechnology*, vol. 24, no. 12, pp. 1565–1567, 2006.

[20] G. Biau and E. Scornet, "A random forest guided tour," *Test*, vol. 25, no. 2, pp. 197–227, 2016.

[21] A. Natekin and A. Knoll, "Gradient boosting machines, a tutorial," *Frontiers in neurorobotics*, vol. 7, p. 21, 2013.

[22] T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho, K. Chen *et al.*, "Xgboost: extreme gradient boosting," *R package version 0.4-2*, vol. 1, no. 4, pp. 1–4, 2015.

[23] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in neural information processing systems*, vol. 30, 2017.

[24] A. V. Dorogush, V. Ershov, and A. Gulin, "Catboost: gradient boosting with categorical features support," *arXiv preprint arXiv:1810.11363*, 2018.