Forecasting Drought Area Percentage in California using Machine Learning Algorithms

Dharma Teja Kolluri, Manish Kumar Sesetti, Rama Krishna Poluru, and Saranya Gondeli

Department of Applied Data Science, San Jose State University

DATA 270: Data Analyst Process

Dr. Eduardo Chan

May 17, 2023

Abstract

California is highly susceptible to frequent, severe droughts that significantly impact ecosystems, agriculture, and water supplies. To identify the adverse effects, it's crucial to forecast future drought scenarios based on past occurrences in California. Prior research has shown that using SPEI to predict drought is viable, with the majority of studies limited to predicting SPEI or SPI values but not the actual drought percentage. This research aimed to expand on the previous research by creating an accurate model for predicting the area percentage of drought-affected regions beyond just SPEI values for each county from 2000 to 2020 by utilizing climatic, drought percentage, and NDVI values. The calculation of SPEI values involves using Thornthwaite equation along with climatic and NDVI data. After preparing the dataset with climatic and SPEI values, Decision Tree Regressor, Random Forest, LSTM, and ANN models are used to forecast drought area percentage for California. The Random Forest model outperforms other models with MAE value of 6.81 and R^2 value of 0.74. ANN and decision tree models achieved good results, with MAE of 7.43 and 7.21, and an R^2 score of 0.70 and 0.72, respectively. These results suggest that decision tree-based models and traditional neural networks are effective in predicting the target variable. The findings indicate that RF models can assist in identifying drought-prone areas and help to implement effective mitigation measures.

Introduction

Project Background and Executive Summary

California is prone to frequent, severe droughts that can seriously affect ecosystems, agriculture, and water supplies. Kiem et al. (2016) said that because of its many influencing factors that operate at different spatial and temporal scales, drought is a complicated phenomenon and one of the least well-known environmental disasters. Belayneh et al. (2016); Hernández-Espinosa et al. (2018) suggested that in comparison to traditional models, machine learning techniques are better equipped to assess the hierarchical and nonlinear interactions between independent factors and dependent variables. The research aims to solve this issue by creating a machine learning-based model that can precisely predict the proportion of California that will experience a drought. An early warning system for the government and better water management decisions are the driving forces behind the initiative. The research can assist the government in preparing for and lessening the effects of droughts in California by precisely estimating the area percentage of drought. The research also aims to advance understanding of the causes of the current drought in California and shed light on how climate change can affect the frequency and intensity of droughts in the future. The recurrent droughts in California have an adverse effect on the state's water resources, agricultural sector, and ecosystems. The project will enable the government to make better decisions about water management, including water distribution and conservation strategies, by delivering accurate and timely information.

The goal is to create machine learning models that use historical weather data to forecast the drought area percentage of California. The project involves the CRISP-DM methodology, a widely used process model for data mining and analytics projects that includes six phases: Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and

Deployment. After careful examination, given the nature of models and the iterative nature of this machine learning project, a CRISP-DM Agile methodology is more appropriate. Agile techniques are made to be adaptable, iterative, and responsive to shifting needs and new information which will be best-aligned characteristics for a machine learning project. The six phases of the CRISP-DM technique will be divided into smaller, more manageable iterations or sprints to be converted to an Agile approach. Each sprint focuses on one or two CRISP-DM phases, enabling the team to advance the project while maintaining flexibility and teamwork as its top priorities.

According to Wilhite et al. (2007), hydrometeorological variables including precipitation, evapotranspiration, sun radiation, and others can be used to calculate drought indices. A complicated interaction between precipitation, air temperature, water vapor pressure, solar radiation, and other factors causes it. Thus, obtained data from the NASA Power website that included daily data on the temperature of the air, the temperature of the earth, the humidity, the wind speed, and the amount of precipitation in California. This data can be used to calculate the Standardized Precipitation Evapotranspiration Index (SPEI) using the Thornthwaite equation, which displays the balance between precipitation and evapotranspiration. The state's NDVI is calculated from satellite imagery. The approach involves using machine learning algorithms to predict the percentage of drought-prone areas in the California region based on historical weather data. It is highly data-driven and involves collecting the required climatic and historical drought area percentage data from accessible and authorized public websites, as well as satellite data. Exploratory analysis is then implemented to identify patterns and trends that can aid in prediction. Later, four machine learning models are built to analyze the historical data and make accurate predictions. The implementation of the approach involves building, training, and testing

the following models: a Decision tree model, a Long Short-Term Memory(LSTM) model, an Artificial Neural Network (ANN) model, and a Random Forest model using the various Python libraries. The temperature of Earth, two-meter temperature, two-meter specific humidity, and two-meter wind speed Correcting for precipitation are the major features and metrics commonly used in remote sensing studies are NDVI and SPEI.

Based on past meteorological data, the suggested project seeks to create a dependable and accurate machine-learning model for forecasting the proportion of areas in California that will be impacted by drought. The goal is to improve drought forecast accuracy by analyzing historical weather data with sophisticated machine-learning methods. This initiative aims to improve water resource management and preparation planning in California. Because of the considerable effects that drought has on agriculture, water availability, and the state's economy, this project is crucial. The developed machine learning model can aid in better planning and management of events in the state and identify areas that are likely to experience drought conditions in the future, which can aid in early warning and preparedness. The metrics/ attributes that are considered are vast compared to other proposed models. The study also advances knowledge of the connection between climatic factors and the frequency and severity of California's droughts. The SPEI, and vegetation index are important indicators that will be used to forecast the percentage of drought-prone areas in the California region. The initiative also attempts to evaluate how well various machine learning algorithms forecast the percentage of California's land that will experience a drought. The model can also be adapted and used for other regions that experience similar climatic conditions and are prone to droughts.

Project Requirements

Functional Requirements that can be Tested and Measurable

In order to implement this research, it is necessary to collect daily climatic information for 51 California counties, including information on temperature, humidity, wind speed, and other variables. The percentages of weekly drought regions should be calculated for the counties under consideration. The Google Earth Engine API's get_ndvi() method should be used to access the Land Remote Sensing Satellite(LANDSAT) images in order to retrieve the NDVI values for the counties. The NDVI values are calculated as the normalized difference between spectral bands 4 and 5, which stand for red and blue in visible light. Using the Thornthwaite equation and the data gathered, the SPEI value should be determined. The suggested models should be built with a variety of Python libraries. The model had to be able to predict droughts with reasonable accuracy, which was determined by comparing the models and finding which one generated the most accurate results.

AI-powered Requirements that Can be Tested and Measurable

Traditional approaches are less effective at predicting droughts than AI-powered models like decision trees, random forests, LSTM, and ANN models. In order to comprehend the relationships among weather systems, soil moisture, and other elements which contribute to drought conditions, they may handle and analyze huge and complicated datasets, including data from numerous sources and time-series data. Decision-makers can better manage water systems and lessen the effects of droughts by using these models since they are more adaptable and flexible and can change with the environment. AI-powered models can generate precise predictions at the regional and local levels, assisting governments in making important decisions regarding water management and drought action plans.

A decision tree model can assist in determining which characteristics or variables are most crucial for drought prediction. The variables that have the biggest effects on the prediction can be identified by looking at the decision rules in the tree. Song and Lu (2015) have also suggested that decision tree algorithms can be used to select the most relevant input parameters required to build decision tree models. In the case of a Random Forest model, it is a straightforward machine learning approach that typically produces excellent results even without modifying its settings. This method is one of the most popular machine learning algorithms for both classification and regression because of how easily it can be applied and how straightforward it is. Furthermore, it produces excellent results even when its parameters are left alone, according to Lotfirad et al. (2021). This method is among the most popular machine learning algorithms for both regression and classification due to its adaptability.Since LSTM can capture nonlinear correlations between variables, they are useful for predicting droughts that entail intricate, non-linear interactions between weather patterns, soil moisture, and other variables. The LSTM may be able to anticipate time series data sets having non-linear correlations, according to Senanayake et al. (2022). Since LSTM can capture non-linear correlations between variables, they are useful for predicting droughts that entail intricate, non-linear interactions between weather patterns, soil moisture, and other variables. The LSTM may be able to anticipate time series data sets having non-linear correlations, according to Senanayake et al. (2022). While droughts are generally nonlinear in nature, neural networks have proven to be a viable model for comprehending the relationship between drought indices and drought-affecting variables, according to Santos et al. (2014) and Nourani et al. (2014).

Data Requirements

In order to train machine learning models for estimating the proportion of California's land designated as drought-prone, Data for this research was gathered from three open sources spanning the period between 2000 and 2020. To determine the NDVI, which represents the proportion of vegetation on the land, Satellite pictures of the state were collected for the study. The percentage of the nation's land area that was determined to be drought-prone for historical drought data was taken from the U.S. Drought Monitor website at the County level. The SPEI, which incorporates monthly data on the atmosphere's temperature, the earth's surface's humidity, and other environmental factors, was calculated using the California environment dataset from the NASA Power website.

Project Deliverables

The project would include the following deliverables. Each deliverable has an appropriate estimation point as per the study, research, and understanding.

Project Proposal

Before deciding on the problem-solving approach, the researchers discussed several issue statements and situations to determine the most effective strategy. Drought is by far the biggest environmental problem in California, so it's critical to put effective techniques in place for estimating the percentage of California that is susceptible to drought. By analyzing historical weather data, this research attempts to create a machine-learning model that is accurate and dependable for predicting the percentage of Californian regions that are drought-prone. This would benefit and impact a huge portion of society. The CRISP-DM Agile approach is being used for this project, and it consists of six phases: Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment. To transition to an Agile strategy,

each phase will be broken down into smaller, more manageable iterations or sprints. Four machine learning models will be created to examine past data and produce precise forecasts.

Project Breakdown

The following are the six phases of the project breakdown

Business Understanding. Going through the available resources online and considering the factors which majorly impact the drought to identify and rank the project's requirements and objectives.

Data Understanding. Understanding satellite data along with easily accessible and permitted public websites to gather the necessary climatic and historical drought region percentage information. The factors considered for each model will be studied to gain domain knowledge of the project.

Data Preparation. Before training machine learning models to accurately forecast drought-prone area percentages, the acquired data needs to be cleaned and preprocessed. Missing values are dealt with through data imputation and data interpolation techniques, handling outliers, transforming the data into the necessary format, feature engineering, data regularization, and dividing the data into training and testing datasets.

Modeling. Building four machine learning models: a Decision tree model using the scikit-learn library, an LSTM model using various Python libraries, an ANN model using a Python library, and an RF model using the scikit-learn library.

Evaluation. Use data to evaluate the model's accuracy. choosing the model that is most useful for determining the severity of the drought in California in various situations by comparing their accuracy.

Deployment. Making the developed application accessible to end users, making sure it satisfies all technical specifications, and providing essential support are all parts of the deployment process. This procedure makes it possible for the application to be used and accessed by the target market.

JIRA, a tool for project management, will be used. The phase of the project is further divided into epics, tasks, and sub-tasks. A Work Breakdown Structure will be utilized to divide the project into several phases. This will assist the team in precisely defining their tasks and developing an acceptable timetable. The project and effort schedule will be monitored using Gantt and Pert charts.

Project Plan

The project's planning is being carried out using the CRISP-DM Agile approach, and it consists of six stages: Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment. Split each step into shorter, more manageable sprints or iterations. To examine historical data and produce precise forecasts, four machine learning models are used.

Data Collection

Collecting the required climatic and historical drought area percentage data from accessible and authorized public websites, as well as satellite data.

https://power.larc.nasa.gov/data-access-viewer/

https://developers.google.com/earth-engine/datasets

https://droughtmonitor.unl.edu/CurrentMap/StateDroughtMonitor.aspx?CA

Prototype

A system will be developed which utilizes four machine learning models for forecasting. These models include the Decision Tree, LSTM, ANN, and RF.

Testing

The testing process involves assessing the overall effectiveness of the model by considering various inputs, and it helps to pinpoint any possible problems or opportunities for enhancement. Several testing metrics such as Mean Absolute Error (MAE), Mean Square Error (MSE), and R^2 scores will be employed as well.

Final Model

The ultimate objective of this endeavor is to develop a comprehensive system for predicting droughts, which will integrate the most effective machine learning model discovered through this research. This algorithm will utilize climatic data as input to generate precise forecasts regarding the extent to which various regions in California are susceptible to drought conditions. By creating a dependable and user-friendly drought prediction system, there is a significant potential to enhance water resource management and preparedness planning in California. Ultimately, this advancement will benefit both the environment and the individuals who depend on it, contributing to improved sustainability and resilience in the region.

Project Report

The project report will encompass a comprehensive account of the project's objectives, detailing the specific goals that were set out to be accomplished. It will provide a thorough description of the methodologies employed to attain these objectives, outlining the steps, techniques, and tools utilized throughout the project's duration. This section will include a comprehensive analysis and interpretation of the data collected, highlighting key observations, patterns, and trends discovered during the project's execution. In order to provide a clear and organized view of the project's progress Table 1 will outline the project deliverables and timeline in detail.

Table 1

Project Deliverables and Timeline

Deliverable	Description	Due Date
Project Proposal	Prediction of percentage area of drought in California	02/24/2023
Project Breakdown	Creation of WBS in JIRA	03/04/2023
Project Plan	Creation of tasks and task assignment, Gantt chart and PERT chart	03/10/2023
Data Collection	Collection of data from NASA Power, US Drought Monitor and Satellite images	03/13/2023
Prototype	Building ML models: DT, RF, LSTM and ANN	04/07/2023
Testing	Testing the models using the MAE, MSE and R2-Score.	04/21/2023
Final Model	Prediction of future combinations.	05/05/2023
Project Report	Documentation of the project in APA Style	05/17/2023

Technology and Solution Survey

Several research papers and journals have investigated drought prediction using multiple datasets and a range of Machine Learning and Deep Learning algorithms. However, due to the absence of a definitive metric for assessing drought severity or percentage, the approaches and models used in these studies have varied. Given the available data resources, the aim of the

project is to develop a model for forecasting the percentage of drought areas in California using four different Machine Learning and Deep Learning algorithms. Some of the current technologies explored for this paper include an analysis of the reasons behind using these techniques and an evaluation of the limitations associated with each model:

Random Forest

Masroor et al. (2021) assessed the effects of drought conditions on groundwater levels in the Godavari basin using a Random Forest Classifier and Analytical Hierarchy Process (AHP). The factors were given weights, and AHP was used to create maps of possible groundwater zones. Several drought conditions were measured using the Standardized Precipitation Index (SPI), and vulnerable drought zones were identified using weighted sum overlay analysis. In order to evaluate the effects of drought in those zones, the researchers used the Random Forest model for the data, using 80% of the data as training data and 20% of the data as testing data. Because of the Random Forest classifier's ease of use, interpretability, and capacity for handling missing data.

Dash et al. (2022) developed a study aimed at predicting drought in various regions of India using the Random Forest Model and Landsat eight Satellite images. The paper also discusses how water quality will be affected due to drought, with Chlorophyll-a being the primary parameter used to measure this. The study employed a methodology that involved data collection from satellite images, developing a drought prediction model using the extracted data, predicting drought severity, calculating water quality using Sentinel data, and determining the time required for training, along with the model’s accuracy. Satellite images were used to create the dataset, with the images being stored as high-resolution .tif files consisting of 11 bands, with each band having a specific .tif file depicting the value of different variables calculated using

various formulae, including NDVI, SAVI, MSI, and NIR. Soil Moisture Index (MSI) was used as the output variable to calculate drought severity, with the MSI value ranging from 0 to 1, where 0 represents a severe drought-prone area and 1 represents a non-drought-prone area. A drought model was developed using the Random Forest Model for a total of 1,166,400 images, with the data being split into training (80%) and testing (20%) to determine model accuracy. The model achieved an accuracy of 94% with an error distribution of 5.38% and was trained within 15 minutes. The study also applied the model to predict future drought severity in Sangli town, India, with the SMI value being close to 0.4-0.5, indicating high chances of drought. The paper suggested that water quality is also impacted by the likelihood of drought, which was observed in the Sangli town example.

Long and Short-Term Memory

Tian et al. (2021) proposed various models to predict drought severity using feature-based transfer learning and various regression algorithms. In this paper, SPEI is considered as a target feature to predict drought severity and it is considered as four scales such as SPEI-3, SPEI-6, SPEI-9, and SPEI-12. Unlike any other paper, researchers used time series imaging and feature-based techniques to extract the features for SPEI value. The main reason to follow this methodology instead of conventional ML or DL-based modeling is there is no perfect metric to find the drought severity and the models become inefficient with changes in the regions and their data. Due to the huge dependency on the study area, researchers have proposed a novel feature extraction be determined by Time series imaging as their prediction system will be independent of the study area and could be applied anywhere in the world. After feature extraction, this paper used the regression models such as Random Forest, LSTM, Support Vector Regression(SVR), and Wavelet Neural Network(WNN). The main reason to choose Random

Forest and Support Vector Regression in this paper is that these models are simplest to use and also consume less time to model the data. LSTM and WNN are neural network models that are used in this paper as these models are good at modeling nonlinear data with a large number of inputs, especially images. When trained with the above models, it is observed that LSTM and WNN models performed extremely well compared to the other two models for long-term periods such as SPEI-9 and SPEI-12. The limitation of this paper is it needed a lot of computational power to work out the paper in real-time as it involves time series imaging technologies and neural networks. Along with this, it is not possible to integrate this model with a conventional dataset approach which will result in the poor performance of LSTM and WNN models.

Dhyani and Pandya (2021) proposed a deep-learning model for drought and agricultural forecasting using LSTM and Time Distributed Convolutional Neural Network (TD-CNN). Satellite images were used to derive values such as NDVI, SPI, and SPEI, and the LSTM model was employed to predict drought occurrence using the SPEI dataset. The study highlighted how different data could impact the performance of each model and recommended predicting SMI and NDVI values using the TD-CNN model since these values are available in the form of images. One of the key advantages of the TD-CNN hybrid model is its ability to recognize sequential relationships between images. Bypassing the extracted features into the LSTM model, which is a specialized Recurrent Neural Networks (RNN) model, the class of the next sequential image can be predicted. After training the hybrid model of LSTM and TD-CNN, the study achieved an accuracy of 96% with a loss of 0.08. The major findings of the study revealed that the hybrid model used SPEI values to build LSTM for drought occurrence prediction, while the TD-CNN model predicted the severity of drought based on NDVI and SMI values. The SMI value was found to be an important metric for determining the type of crop to be planted. One

significant limitation of this model is its narrow focus on a limited dataset that is specific to a particular area. As a result, using this model with different sets of data may not yield optimal results.

Mokhtar et al. (2021) conducted a study to develop machine learning and deep learning models to predict drought severity in the Qinghai-Tibet Plateau, which is one of the most sensitive areas to global climatic change. To predict drought severity, the researchers used Extreme Gradient Boost (XGB), Convolutional Neural Network (CNN), RF, and LSTM algorithms. Since there is no particular attribute to measure drought, the researchers used various scenarios for different variables as datasets. The output variable chosen in this paper is the SPEI, which is calculated based on various attributes such as precipitation, temperatures, and relative humidity. The SPEI value is categorized into two sections: the 6-month SPEI value and the 3-month SPEI value. For each category, the climatic data has been divided into seven cases, with the 7th case consisting of all climatic data including solar radiation, the 6th case consisting of climatic data, and so on. To split the dataset into training and testing, the researchers used PCA and identified five climatic zones, which were used as testing. For the other 25 climatic zones, the aforementioned ML and DL algorithms were performed to predict the SPEI value. The drought severity is measured as follows: SPEI value > 0 is no drought, <-1.65 is very extreme drought, and so on. After analyzing each scenario and category for each model, this paper suggested that the XGB model with scenario five is superior in the SPEI-3 category, and the XGB model with scenario seven is superior in the SPEI-6 category. Among all the models, the XGB model performed the best, producing the highest NSE value of 0.71, followed by the Random Forest model with a value of 0.68. Scenario seven produced the highest NSE value with the XGB model in the SPEI-3 category. However, in the SPEI-6 category, the RF model

produced the highest NSE value of 0.69 in scenario seven but with an MBE value of -0.08. The major challenge faced by the researchers in predicting drought severity is choosing the correct data, as there is no perfect metric to calculate the drought percentage or severity. This paper demonstrated the usage of various factors in various scenarios to predict drought severity.

Liu et al. (2022) proposed the LSTM model and its variants as algorithms for predicting COVID-19-positive cases in Wuhan and the United States. In this paper, LSTM model prediction is compared with the SEIR model which is an epidemiological model used to predict such contagious diseases and to find the spreading of such diseases. The main reason for choosing LSTM for this paper is that it stores the long-term information in a Cell state such as the previously calculated data and then forgets the information after the cell is updated with other weighted values. In this paper, researchers wanted to use the Bidirectional LSTM model which can store the values in the hidden layer and visible layer which is used for forward and backward propagation respectively. Using this model, researchers were able to achieve an accuracy rate of 94% by training it with the Wuhan dataset and testing it on the dataset from various states in the USA. The Bidirectional LSTM model outperformed the LSTM model and SEIR model in terms of accuracy. The main reason for using this model in the project is its ability to store long-term information and can handle sequential data to find the patterns and relationships among data which is particularly useful in time series forecasting.

Artificial Neural Network

Morid et al. (2007) created a method using a time series of drought metrics and artificial neural networks (ANNs) for forecasting drought episodes. The Standardized Precipitation Index (SPI) and the Palmer Drought Severity Index (PDSI), two drought indices, are used in the study as inputs to the ANNs. The authors used ANNs to predict drought episodes over three-month

intervals using data from four Iranian meteorological stations. Using statistical metrics like the correlation coefficient and root mean square error (RMSE), the performance of the ANNs was assessed (R). This paper's findings demonstrated that using time series of drought indices as inputs, ANNs could predict drought episodes with high accuracy. The PDSI was found to be a more accurate predictor of drought episodes than the SPI, and when both indices were utilized as inputs, the performance of the ANNs increased. As it reveals how well ANNs work at forecasting drought events using time series of drought indices, the paper makes a valuable contribution to the field of drought forecasting. The study also emphasizes how crucial it is to incorporate different drought indices to enhance the precision of drought forecasts.

Barua et al. (2012) produced a study that outlines an ANN-based strategy for predicting drought using a nonlinear aggregated drought index (ADI). In order to give more precise and trustworthy drought predictions, the authors suggest a novel strategy that combines the advantages of the ANN and ADI models. The importance of drought forecasting and the shortcomings of conventional techniques are covered in the opening section of the paper. The ADI model, which is used to depict the general drought conditions in a region, is next introduced. To aggregate many drought indicators into a single ADI value, the authors suggest a nonlinear aggregation method. The study then goes on to describe the ANN model for drought forecasting. The ANN is used to anticipate future drought conditions based on current and predicted climate variables. It is trained using historical climate data and ADI values. The case study that was used to assess the suggested methodology was forecasting drought in Indonesia's Upper Citarum River Basin, as the authors go on to explain. The outcomes demonstrate that the nonlinear ADI-based ANN-based methodology surpasses conventional methods in terms of accuracy and dependability. The possible uses and restrictions of the suggested approach are

discussed in the paper's conclusion, along with some suggestions for future research. Overall, by suggesting a novel strategy that integrates the advantages of two unique models, the study work makes a significant contribution to the field of drought forecasting.

Agana and Homaifar (2017) proposed a model to predict the long-term approach of drought severity using deep learning techniques such as ANN and Deep Belief Network (DBN). This paper discussed the multiple previous papers on machine learning algorithms that have been used to predict drought severity and came up with a deep learning algorithm of ANN for their model. The main reason to choose this algorithm is that the effects of climate change and factors involved in drought need to be addressed with a much more complicated algorithm ANN, which is suitable for complex time series forecasting and approximating any complex function. Despite having these many layers and complexity to implement the model, when there are more than two hidden layers needed for highly complicated systems, the non-convex optimization problem arises. To overcome this issue, the Deep Belief Networks (DBN) algorithm has been used to implement the model. The issue with ANN is that the weights are initialized randomly, which may cause delayed convergence or cause the training model to become stuck at local optimums in hidden layers. In order to establish starting values for the dataset being utilized, the DBN therefore uses greedy unsupervised learning. Then, the DBN uses supervised methods to adjust the existing network. Restricted Boltzmann Machines (RBM), generative energy-based models that are used to configure the hidden and visible layers, are employed to structure this DBN. The multilayer neural network can be trained effectively by using feature activations from one layer as the training data for the subsequent layer. Layer-wise unsupervised training can result in initial weight values for all layers that are better than random. The DBN was used to start setting the

parameters, then unsupervised learning was used to train the DBN, and the backpropagation algorithm was used to fine-tune the entire network. This paper also used
MLP and SVR models to compare the accuracy of the DBN model, which outperformed the other models with 76% of accuracy. The major drawback of the paper is that the data is minimal, and for models like DBN, vast amounts of data are required to improve the performance. The only use of a standardized data flow index could be another drawback. The main reason for choosing ANN was their parallel processing capability and suitability for time-complex problems due to their multi-layer functionality, which is expected to result in high performance when trained with the data

Nabipour et al. (2020) presented a model to predict the hydrological drought which affects the water management system using ANN with multiple optimization techniques. This paper analyzed various previous research to understand physical, conceptual, and data-driven models that are used to predict drought percentages. Researchers believed that physical and conceptual-based models are data intensive and could be performed well when there is all the data available to predict. Since there is no concrete data to predict drought, it is better to develop a data-driven model which can perform well with limited data available. The researchers opted to use ANN as a model due to its capabilities in parallel processing, handling limited and noisy data, and pattern recognition. Despite its advantages, using ANN has certain disadvantages, as the architecture of ANN is considered based on a trial and error method
and it is a black-box model that does not consider the input and output intermediate processes. To address these issues, four optimization techniques are hybridized with ANN which are Grasshopper Optimization Algorithm (GOA), Salp Swarm algorithm (SSA), Biogeography-based optimization (BBO), and Particle Swarm Optimization (PSO). The data is

taken from the Dez Dam basin which is used to train the data using the conventional ANN model and hybridized models. After training the data, research showed that the hybridized models performed extremely well compared to the conventional model especially the ANN-PSO model showed great results with an accuracy rate of 84%. The main disadvantage of this paper is that the model takes a lot of time to train the data and huge computational power is required. Along with that, it needs lots of data to get better results.

Decision Tree

Prasad et al. (2013) proposed a methodology to forecast rainfall by leveraging data mining techniques and a decision tree classifier. The authors selected this algorithm due to its ease of implementation, effectiveness, and compatibility with data mining techniques. The researchers utilized the Supervised Learning in Quest (SLIQ) approach to training the model, which eliminates the need for sorting data by storing it in a separate memory location for each attribute in the dataset. The study yielded an accuracy rate of 72%, demonstrating the model's effectiveness in predicting precipitation, which is a key attribute in rainfall forecasting. However, the authors acknowledge that this approach may perform better with less complex data due to its simplicity and ease of implementation.

Chopda et al. (2018) proposed a technique to identify cotton crop disease by employing a decision tree classifier that considers various parameters such as temperature, soil moisture, precipitation, and more. The authors chose to use the decision tree classification algorithm for its ease of use, as it can be visualized and interpreted to make decisions, which is a primary objective of the study. The paper demonstrates how a decision tree classifier can provide quick and easy predictions of cotton crop disease using basic parameters. The Decision Tree classifier

was selected for the study primarily due to its simplicity in interpretation, faster training time, and lower risk of overfitting compared to other neural network algorithms used in this research.

Kim et al. (2020) authored an article with the objective of creating a drought index forecasting model utilizing decision tree-based techniques. In the paper, the significance of drought forecasting for effective water resource management is discussed, and a decision-tree-based method for forecasting the SPEI is suggested. In order to estimate SPEI values, the study employs data from the Korea Meteorological Administration and decision tree-based methods, such as classification and regression tree (CART), chi-squared automatic interaction detector (CHAID), and RF. The authors compare the effectiveness of different algorithms using statistical indicators such as correlation coefficient (R), mean squared error (MSE), and mean absolute error (MAE). Since the RF model produces the lowest MAE, MSE, and greatest R values, the results demonstrate that it outperforms the other models in terms of forecasting accuracy. The study finds that RF is an efficient way of forecasting SPEI values and that decision tree-based algorithms can be useful in predicting drought indicators. In order to create drought prediction models that can help with water resource management, the authors propose that their methodology can be applied to different regions. Overall, the study provides a technical assessment of the use of decision tree-based algorithms for forecasting drought indices that may be helpful to academics and professionals with an interest in water resource management and climate change.

Comparison of Existing Technology

According to Dash et al. (2022) and Masroor et al. (2021), both authors proposed research papers to assess drought severity using a Random Forest classifier and satellite images, but their approaches are entirely different. In Dash et al. (2022), sequential images were used as

input data for the model, achieving a high accuracy rate of 94%. On the other hand, Masroor et al. (2021) calculated the SPI value from the images and trained the extracted data with the model, achieving an R square value of 0.858. When comparing both papers, Dash et al. (2022) achieved higher accuracy by training the images based on the bands than by extracting SPI values. The above two papers by Nabipour et al. (2020) and Agana and Homaifar (2017) used ANN as their model in which one of the papers used optimization techniques and the other paper used the DBN algorithm to predict drought. However, each paper used hybrid ANN for modeling and for training the data but the approach followed by Nabipour et al. (2020) is the conventional way of taking the training data from one basin and training the data to calculate the accuracy of the model but the approach followed by Agana and Homaifar (2017) is that they used a backpropagation algorithm to fine-tune and train the data. The reason behind choosing hybrid models for each paper is to handle the architecture of the ANN model which is a major challenge in both cases. Furthermore, the accuracy rates of both papers were impacted by the inadequacy and restricted nature of the data, since ANN necessitates an enormous amount of data to train the model and produce better outcomes. It would be inappropriate to evaluate which approach is superior since each approach has its own set of advantages and disadvantages.

Literature Survey of Existing Research

Numerous papers, academic journals, and research have been published and disseminated regarding the root causes and consequential impacts of drought, along with the utilization of various machine learning algorithms for accurately predicting such circumstances. Although interpolation techniques based on multiple datasets could potentially yield insight into drought percentages, the utilization of machine learning and deep learning models has significantly improved precision levels. Many machine learning methods exist, including the Support Vector

Machine (SVM), ANN, Decision Tree, boosted trees, bagging, Matern 5/2 Gaussian process regression (GPR), and M5P Regression. However, obtaining extensive historical data for the particular regions under study is a considerable difficulty for researchers. Based on the available data, many factors were taken into consideration in the papers and research. As an illustration, many studies have used the SPI, NDVI, Atmospherically Resistant Vegetation Index (ARVI), Soil Adjusted Vegetation Index (SAVI), SPEI, and other pertinent variables to predict the percentage of drought. This section discusses the existing research papers used and their results.

Alireza et al. (2012) conducted a study to compare the effectiveness of ANN, SVM, Least Squares Support Vector Machines (LS-SVM), and three-layered Feed Forward Neural Networks (FFNN) in predicting droughts. Despite the drawbacks of ANN like training the complex structure components, LS-SVM presents an advantage in computation over standard SVM since it converts a quadratic problem into a linear equation. To evaluate the models he used meteorological data and lagged data, and statistical performance evaluation measures. His study discovered both LS-SVM and ANN are efficient and outperform traditional statistical models. LS-SVM models use various normalization types which lead to lower RMSE and Weighted Mean Absolute Percentage Error (WMAPE) values and higher R2, adj. R2, and Nash–Sutcliffe coefficient (NS) values than the FFNN models. LS- SVM got an accuracy score R2 of 0.84 and 0.69, whereas the accuracy score of FFNN is 0.83 and 0.67.

Ganguli and Reddy (2014) create ensemble drought prediction models that, by using numerous models for various lead times, will boost accuracy. using a SVM - copula technique, which analyzes climatic indices like El Nio Southern Oscillation (ENSO), Indian Ocean Dipole Mode (IOD), and Atlantic Multidecadal Oscillation and uses the SPI to identify meteorological droughts (AMO). To obtain nonlinearity and nonstationarity, the Least Square Support Vector

Regression (SVR) has been applied. Two forecasting case models were developed, first case model is to predict drought overall the year without considering the seasons, and the second case model has three models which predict droughts in each season in a year. Further proceeding three types of copulas (Clayton, Frank, and Plackett) are used for drought modeling, and the best model is used for ensembling, for each different season with various lead months different copula-type models have been selected as the best model where p-values are higher. Generated time series comparisons for one, two, and three months lead which shows ensemble SPI value via copula approach and boxplot that shows the uncertainty of the drought prediction. The inclusion of climatic variables helps to predict the droughts accurately when compared to the model without seasonal partition. The uncertainty in predicting the drought has been reduced by adding climatic variables and also implementing the best copula model for up to three months of lead time.

Masinde (2014) proposes a combined solution of ANN and Effective Drought Index(EDI) to develop a forecasting model that can predict drought in a range from one day to four years. Using the historical weather data of 36 years from four weather stations located in Kenya. The accuracy of the model is in the range of 75 % to 98 %. The study outlines three phases involved to generate ANN models. In Phase-I, 30 years of historical data from one data station were used to create 21 neural networks via various combinations of Rainfall (Precipitation), EDI, and Available Water Resource Index(AWRI) values, in which the 10 best neural networks which are performing great has been selected to investigate further in Phase-II. In Phase-II, the data station used in Phase-I along with two new Data stations weather data has been taken to forecast the best two neural networks for EDI and AWRI to predict the drought from a range of one day to four Years. Facing climate change issues such as changes in

precipitation leads to difficulty in forecasting drought, with an average accuracy of 70%. In Phase III, he compared the forecasted data on the last weather station data which is not used in any of the phases before, then accuracies of different time spans have been increased consistently and have the highest accuracy of 98%, and currently, the research is still going on.

Vicente-Serrano et al. (2015) presented the relative importance of precipitation (P) and evapotranspiration (ETo) to four drought indices: the standardized Palmer Drought Index (SPDI), the Reconnaissance Drought Index (RDI), the palmer drought severity index (PDSI), and the SPEI. The research covered widespread temperature and precipitation across 12 distinct climate zones. The importance of P and ETo to decide drought conditions varied depending on climate zone and year. The research shows that precipitation was the main component in wet regions, whereas ETo played an important role in drought indices in dry regions where the supply of water is limited. And ETo on drought indices influences the warm season more when compared to the cold season specifically in dry regions. This tells that ETo plays a major role than P to predict drought during the warm season. As many predict droughts by precipitation (P), involving Evapotranspiration (ETo) as one of the features for predicting droughts has given more accurate results, especially in dry climates with high evapotranspiration rates. The limitation of this research is the dataset uses only limited environmental factors, which might not be sufficient to predict droughts in different climatic zones across regions.

Swain et al. (2018) aim to investigate the future trends in precipitation in California using climate model projections. The authors look at how precipitation events have changed in California over the past twenty years in terms of their frequency, intensity, and seasonality. According to the study, while the amount of precipitation in California is expected to rise on average, it will do so with a different distribution, with longer dry spells and more frequent and

intense precipitation events. The paper finds that California's precipitation will become more volatile, increasing the risk of droughts and floods and making rainfall patterns less predictable. According to the authors, California's current infrastructure for managing water resources may not be sufficient to handle the rising variability of precipitation, requiring the development of new strategies for water allocation and storage. The study also identifies anthropogenic climate change as the primary cause of the rise in precipitation volatility, which has already increased the likelihood of droughts in California. The paper argues that the government needs to consider the increasing volatility of California's precipitation in its long-term planning for water resources and infrastructure. In conclusion, the paper emphasizes the necessity of proactive planning and adaptation to lessen the effects of future changes in precipitation on California's water resources and economy using any available statistics and technologies.

Xu et al. (2018) use information from the North American Multi-Model Ensemble (NMME), which employs statistical and hybrid models, to assess the effectiveness of drought forecasting in China. By incorporating the SPI into the four ML models Bagging (BG), Random Subspace (RSS), Random Tree (RT), and Random Forest, the objective is to anticipate droughts with a lead time of up to six months (RF). A variety of datasets based on three different elevations were used to implement each model. These models were trained using classifiers like M5P, Additive Regression (AR), Gaussian Process Regression (GPR), REPTree, and SMO-Support Vector Machine (SMO-SVM), with the results demonstrating that REPTree is the best classifier. Once the data is trained with different elevation data on all models, during the training phase the performance of these models is as follows: RT > RF > BG > RSS. In the testing phase, RT became the worst performing, and rest three models improved their accuracy as BG > RF > RSS > RT and BG and RF showed great improvement in correlation and lower

RMSE values. In the Validation phase, BG has proved to predict better and the BG has been used in other different data station data sets to predict droughts. Limitations in this research are completely dependent on the SPI values which don't consider other important environmental factors such as Temperature, Evapotranspiration, and Radiation. The author suggested adopting rainfall, temperature, and relative humidity as features so the models can predict more accurately compared with BG and the rest of the other algorithms.

Wang et al. (2019) proposed an analysis of drought severity using temporal and spatial variation techniques, using the SPEI value as a target metric. According to the study, three kinds of metrics could be used to measure drought severity. They are SPI, Palmer Drought Severity Index (PDSI), and SPEI. The main reason to choose SPEI as the target metric is that the SPI value can only be calculated at different time intervals and will not consider any other variables that could be the factors of drought. PDSI is based on the soil water balance equation, which does not indicate the multi-scalar character. Since SPEI overcomes all these issues, this metric is used as an indicator for drought analysis based on spatial and temporal techniques. The SPEI was chosen as one of the project measures due to its fit for the research goals, as it will cover all the factors involved in drought prediction.

Granata (2019) examines the effectiveness of several machine learning methods for evaluating evapotranspiration. For comparison purposes, three models were developed, each using a different combination of the four ML techniques M5P Regression Tree, Bagging, Random Forest, and Support Vector Regression. The Nash-Sutcliffe model efficiency coefficient (NSE), MAE, RMSE, and relative absolute error are used to determine how effective the aforementioned models are (RAE). Each model reduces the input features in order to assess the effectiveness and mistake rates of each approach. All algorithms in Model-1 were trained using

the input features of net solar radiation, sensible heat flux, soil moisture content, wind speed, mean relative humidity, and mean temperature. The results showed that M5P is the best, with a great accuracy of NSE=0.987 and a lower error rate (MAE =0.14 mm/day, RMSE =0.179 mm/day, RAE = 15.4%), and Bagging is the least accurate among the four, with a low accuracy of NSE=0.98. In Model-2, all algorithms have been trained on input features of net solar radiation, wind speed, mean relative humidity and mean temperature, compared with the results as Random Forest is the finest with high accuracy of NSE=0.951 and lower error rate (MAE =0.267 mm/day, RMSE =0.343 mm/day, RAE = 29.3%) and whereas Support Vector Regression is the least efficient between four with low accuracy of NSE=0.933 and high error rates (MAE =0.321 mm/day, RMSE =0.399 mm/day, RAE = 35.4%). The results show that M5P Regression Tree, Bagging, and Random Forest have higher efficiencies of NSE= 0.949 with comparable error rates, whereas Support Vector Regression is the least accurate among the four, with low accuracy of NSE=0.932 and high error rates (MAE =0.322 mm/day, RMSE =0.400 mm/day, RAE = 35.4%). In Model-3, all algorithms were trained on input features of net solar radiation, mean relative humidity, and mean temperature. For forecasting actual evapotranspiration and implementing ML models using ANN and ELM, the author proposed the input features: net solar radiation, mean relative humidity and mean temperature are relevant values when compared to wind speed.

Wang et al. (2019) performed a study to optimize the selection of the predictor process to develop a new model that should predict droughts with a lead time of up to six months. They used information from 32 sites in Shaanxi province, China, together with environmental variables like humidity, pressure, and temperature between the years 1961 and 2016. employing techniques like cross-correlation function (CCF) and a conventional statistical model distributed

lag nonlinear model (DLNM) to choose the best predictors and lag time. The result obtained has DLNM outpaced CCF for selecting optimal predictors and lag time, so DLNM was selected and also implemented on two ML models of ANN and XGBoost. These models can distinguish between three different forms of drought by using a 10-fold cross-validation method to predict SPEI. SPEI predictions for lead times up to 6 months were made by comparing the model validation findings. Using cross-validation R2 values of 0.68-0.82, 0.72-0.89, 0.81-0.92, 0.84-0.95 for three, six, nine, and twelve-month time scales, the results demonstrated that XGBoost predicted more accurately than DLNM and ANN. For general droughts, XGBoost's accuracy values ranged from 89% to 97%, while for specific types of drought, such as moderate, severe, and extreme, they ranged from 76% to 94%. SPEI and drought predictions using the XGBoost model have improved thanks to the integration of nonlinear and lag effects of predictors.

Poornima and Pushpalatha (2019) did a study on long-term drought forecasting since Holt-Winters and ARIMA statistical models, which have been used by many in the past, are not accurate enough for long-term predictions. Using data from the Indian Cauvery River basin and advanced applications of ANN and Deep Neural Networks (DNN), a model for long-term drought prediction was built. A statistical model called ARIMA and a deep learning model called LSTM recurrent neural networks (RNNs) were developed to predict droughts using the SPI and the SPEI over timescales of one, six, and twelve months, respectively. In the ARIMA model, they have not used Humidity and Temperature for the calculation of SPI and SPEI, as it is a univariate approach, whereas in LSTM RNN uses a multivariate approach in which Humidity and Temperature as they have a positive correlation to SPI and SPEI. For one monthly scale, ARIMA has given a good accuracy of 99% and RMSE of 0.1 whereas for six and 12 months

accuracy has deteriorated because of the ARIMA working principle which is an univariate prediction model. But in LSTM using input features as a combination of SPI, SPEI, and Humidity, Temperature gave good accuracy scores of about 99% which is computationally expensive, and for long-term time scales LSTM vastly outpaced the accuracy and RMSE of ARIMA. The authors suggested future work has to be implemented on large datasets with ensemble deep learning models that might predict better results.

Khan et al. (2020) used data from 861 grid locations in the National Centres for Environmental Prediction/National Centre for Atmospheric Research (NCEP/NCAR) reanalysis data from 1948 to 2016 to create a model for predicting droughts in Pakistan. The research uses SVM, ANN, and k-Nearest Neighbour (KNN) to predict several categories of droughts in two seasons named Rabi and Kharif within a time range of one to six months. SVM has given the best results with a high spatial correlation of 0.94 and low NRMSE of 0.4 when compared to ANN and KNN with a spatial correlation of 0.87 and 0.85 with NRMSE of 0.6 and 0.7. Future drought forecasting models in Pakistan are advised to try ML techniques like Random Forest (RF), Genetic Programming (GP), Parallel Multi Population Genetic Programming (PMPGP), and Extreme Learning Machine (ELM). SVM-based models outperformed KNN and ANN-based models in this study's validation data in terms of accurately capturing the temporal and geographic characteristics of droughts in Pakistan.

Sardar et al. (2022) employed the Barnacles Mating Optimizer (BMO) method in conjunction with a Convolutional Neural Network (CNN) model to predict drought in Karnataka, India, using satellite photos. In this paper, an ensemble hybrid algorithm was used to attain high accuracy for the prediction. The study involved a methodology comprising data collection, computation of drought indices, dataset preparation, ensemble model creation, and performance

analysis. Satellite images were sourced from the Indian government's official website and processed into a database object using Rasterio's function from the Python library, with multiple values such as NDVI, SAVI, EVI, and ARVI being derived based on the green vegetation color, red color, and blue color. The extracted data is stored in the database once the required values are preprocessed and calculated. After being stored in the database, the whole set of data was loaded, split into training and testing datasets with a 70:30 ratio, and then trained using the CNN model. Based on the NDVI value, the output was classified as either severe, moderate, or low drought. The accuracy and processing time of the model were utilized to examine its performance. The accuracy of the CNN model alone was 91%, but when paired with the BMO method, it increased to 94%. The study also hypothesized that employing a hybrid model built on various bio-inspired algorithms could improve prediction precision even more.

Comparison of Relevant Research Papers

Drought prediction is a challenging task as the complex differences between atmospheric, hydrological, and ecological processes. Recent years have seen the use of ML models to forecast droughts utilizing a variety of data, including meteorological, remote sensing, and soil moisture data. Same ML models can predict different results of droughts because of differences in input data, differences in model configurations, and differences in evaluation metrics. For example, a Random Forest model trained on meteorological data might have higher accuracy in predicting droughts in one region compared to another region due to differences in climatic conditions and weather patterns. And additionally, each paper might have a different best-performing model based on their specific research question, input data, and evaluation metrics. Additionally, because of variations in the data used for training and testing, the accuracy of the same ML model can fluctuate between studies. As an illustration, Khan et al. (2020) and Mokhtar et al.

(2021) both employed machine learning algorithms to predict droughts, however, Mokhtar et al. used SPEI meteorological drought data whereas Khan used drought data from Pakistan. The evaluation of statistical, NMME, and hybrid models for drought prediction in China was also conducted by Xu et al. (2018). Granata (2019) evaluated machine learning-based evapotranspiration evaluation models, whereas Ganguli and Reddy (2014) predicted regional droughts using climatic inputs and the SVM-copula method. Masinde (2014) employed artificial neural networks to forecast the effective drought index, while Wang et al. (2019) used a statistical model and machine learning approaches to predict the meteorological drought in Shaanxi Province, China. Moreover, Alireza et al. (2012) used support vector machines to forecast seasonal meteorological drought Finally, Breiman (2001) employed random forests for classification and regression analysis, and Poornima and Pushpalatha (2019) forecast drought based on SPI and SPEI using LSTM recurrent neural network. Therefore, it is important to carefully consider the data sources and methodology when using machine learning models for drought prediction. Additionally, it is important to note that the same machine learning models can be useful in predicting drought, they should not be the only method used to assess drought risk.

Data and Project Management Plan

Data Management Plan

Data Collection Approaches

The objective of this project is to examine the relationship between weather conditions and drought in the state of California by analyzing a time-series dataset spanning from 2000 to 2020. The dataset was compiled from three sources:

- NASA Power
- US Drought Monitor
- Earth Engine Data Catalog.

The data for this project was obtained from three sources. The first source, NASA Power, provided continuous data in CSV format on various climatic features such as temperature, humidity, wind speed, and more. The US Drought Monitor provided data in CSV format on the percentage of California regions affected by drought at various levels during the specified time period for each county. The third source involved the extraction of additional data by analyzing satellite photographs of California's terrain, including the vegetation index and soil moisture index, through the use of the Earth Engine Python API for each county. The data quality was maintained through the use of the Pandas library in Python to handle missing values, duplicate values, and outliers, and validation was performed to ensure that the data met certain standards such as data type, range, and format. Access to and downloading of the dataset does not require authorization, as it is easily accessible to the public. The information contained in the dataset pertains to climate, which is not considered sensitive information. The license for making these datasets accessible to the general public is held by both data source repositories.

Data Management Methods

The data used for this project was obtained from observational sources where sensors capture real-time information. The datasets consist of numeric values representing various climatic features such as temperature, humidity, wind speed, precipitation, and more. The US Drought Monitor data is classified into five drought levels, varying in the range between abnormally dry and exceptional drought, denoted as D0, D1, D2, D3, and D4, respectively, to indicate the severity of drought conditions. As these datasets are time-series data, they are continuously growing, with new data being added over time, while the old data remains unchanged. To understand these datasets, it is essential to have knowledge about the units and attributes associated with them. For instance, temperature parameters in the first dataset are measured in Celsius, while wind speed is measured in meters per second (m/s). Since the data is self-explanatory, there is no need to document metadata.

The data used in this project is publicly available and can be downloaded and shared without any restrictions or licenses. The data has been carefully selected for its relevance and reliability and is fully de-identified to preserve the anonymity of the participants. The data is stored on Google Drive, which provides a secure and cost-effective platform for data sharing and storage. There are no additional charges for data storage, as long as the data does not exceed the storage limit. To ensure efficient data management, specific data management tasks, such as data collection, processing, management, and exploration, have been assigned to each team member. These protocols are in place to ensure the accuracy and integrity of the data. Table 2 shows the activity performed by each team member for Data Management.

Table 2

Roles and Assignees Used for Data Management

Roles	Assignee
Data Collection	Saranya Gondeli
Data Processing	Rama Krishna Poluru
Data Management	Dharma Teja Kolluri
Data Exploration	Manish Kumar Sesetti

To ensure data consistency and versioning, the final datasets were saved on Google Drive, with regular backups made by duplicating the original file. The data files can be shared among team members without encryption or access permissions and can be accessed for long-term use.

Data Storage Methods

Following data collection, the data was stored in a Google Drive account accessible only by the research team. Three different folders were created to store the raw data, processed data, and backup copies. The uploaded files included power_regional_data_raw.csv, containing climatic data obtained from the public NASA Power website in CSV format with a size of approximately 300 MB, and U.S. Drought Monitor.csv, containing weekly data on the percentage of areas affected by drought conditions, in CSV format with a size of 26 MB.

Figure 1- 3 shows raw datasets of climate data, drought data, and NDVI data that were collected.

Figure 1

Raw Climatic Dataset

-BEGIN HEADER-
NASA/POWER CERES/MERRA2 Native Resolution Daily Data
Dates (month/day/year): 09/02/2006 through 12/31/2006
Location: Regional
Elevation from MERRA-2: Average for 0.5 x 0.625 degree lat/lon region = na meters
The value for missing source data that cannot be computed or is outside of the sources availability range: -999
Parameter(s):
T2M MERRA-2 Temperature at 2 Meters (C)
T2MDEW MERRA-2 Dew/Frost Point at 2 Meters (C)
TS MERRA-2 Earth Skin Temperature (C)
T2M_RANGE MERRA-2 Temperature at 2 Meters Range (C)
T2M_MAX MERRA-2 Temperature at 2 Meters Maximum (C)
T2M_MIN MERRA-2 Temperature at 2 Meters Minimum (C)
QV2M MERRA-2 Specific Humidity at 2 Meters (g/kg)
RH2M MERRA-2 Relative Humidity at 2 Meters (%)
PRECTOTCORR MERRA-2 Precipitation Corrected (mm/day)
PS MERRA-2 Surface Pressure (kPa)
WS10M MERRA-2 Wind Speed at 10 Meters (m/s)
WS10M_MAX MERRA-2 Wind Speed at 10 Meters Maximum (m/s)
WS10M_MIN MERRA-2 Wind Speed at 10 Meters Minimum (m/s)
WS10M_RANGE MERRA-2 Wind Speed at 10 Meters Range (m/s)
WD10M MERRA-2 Wind Direction at 10 Meters (Degrees)
-END HEADER-

LAT	LON	YEAR	MO	DY	T2M	T2MDEW	TS	T2M_RANGE	T2M_MAX	T2M_MIN	QV2M	RH2M	PRECTOTCOI	PS	WS10M	WS10M_MA	WS10M_MI	WS10M_RAI	WD10M
32.75	-123.75	2006	9	2	17.28	15.57	18.65	0.45	17.47	17.02	10.89	89.56	0.25	101.44	7.73	8.61	7	1.61	331.97
32.75	-123.25	2006	9	2	17.05	15.55	18.24	0.49	17.24	16.75	10.89	90.69	0.3	101.4	8.02	8.98	7.2	1.78	331.47
32.75	-122.75	2006	9	2	16.9	15.57	18.04	0.51	17.1	16.59	10.91	91.68	0.25	101.36	8.4	9.38	7.5	1.88	330.67
32.75	-122.25	2006	9	2	16.78	15.52	17.8	0.49	16.97	16.48	10.88	92.12	0.14	101.31	8.76	9.74	7.82	1.91	329.69
32.75	-121.75	2006	9	2	16.66	15.41	17.5	0.44	16.82	16.37	10.8	92.19	0.03	101.25	9.08	10.02	8.1	1.91	328.38
32.75	-121.25	2006	9	2	16.55	15.39	17.32	0.41	16.69	16.28	10.8	92.72	0.04	101.19	9.34	10.22	8.27	1.95	326.25
32.75	-120.75	2006	9	2	16.66	15.38	17.41	0.42	16.82	16.4	10.8	91.97	0.04	101.13	9.61	10.49	8.38	2.11	323.15
32.75	-120.25	2006	9	2	16.79	15.36	17.43	0.35	16.93	16.59	10.8	91.12	0.02	101.06	9.55	10.4	8.22	2.18	318.59
32.75	-119.75	2006	9	2	17.13	15.49	17.77	0.32	17.3	16.98	10.9	89.85	0.01	100.99	8.98	10.14	7.53	2.61	313.62
32.75	-119.25	2006	9	2	17.73	15.88	18.51	0.43	17.96	17.54	11.19	88.75	0	100.92	7.93	9.63	6.33	3.3	308.16
32.75	-118.75	2006	9	2	18.42	16.66	19.49	0.68	18.76	18.08	11.75	89.38	0	100.88	6.6	8.49	4.86	3.63	300.94
32.75	-118.25	2006	9	2	19.46	17.45	20.28	1.24	20.11	18.87	12.38	88.18	0.03	100.78	4.79	7.39	2.68	4.71	298.41
32.75	-117.75	2006	9	2	21.3	17.24	21.92	3.86	23.44	19.59	12.36	79.66	0.23	100.47	3.3	6.21	1.35	4.86	289.31
32.75	-117.25	2006	9	2	24.85	14.58	25.77	9.14	29.85	20.73	10.86	59.16	0.52	98.32	2.23	4.61	0.61	4.01	270.43
32.75	-116.75	2006	9	2	29.26	10.74	30.8	14.76	37.2	22.45	8.7	34.91	0.73	95.01	1.81	3.2	0.44	2.76	233.55

Figure 2

Raw Drought Dataset

county	dates	None	D0-D4	D1-D4	D2-D4	D3-D4	D4	DSCI
Mariposa County	1/10/00	28.6	71.4	0	0	0	0	71
Sonoma County	1/10/00	68.2	31.8	0	0	0	0	32
Shasta County	1/10/00	22.29	77.71	0	0	0	0	78
Del Norte County	1/10/00	0	100	0	0	0	0	100
Inyo County	1/10/00	0	100	0	0	0	0	100
San Bernardino Cour	1/10/00	0	100	0	0	0	0	100
Solano County	1/10/00	69.16	30.84	0	0	0	0	31
Imperial County	1/10/00	0	100	0	0	0	0	100
El Dorado County	1/10/00	0	100	0	0	0	0	100
Los Angeles County	1/10/00	19.04	80.96	0	0	0	0	81
Mono County	1/10/00	1.25	98.75	0	0	0	0	99
Orange County	1/10/00	0.3	99.7	0	0	0	0	100
Monterey County	1/10/00	99.03	0.97	0	0	0	0	1
Ventura County	1/10/00	96.01	3.99	0	0	0	0	4
Lassen County	1/10/00	1.99	98.01	0	0	0	0	98
Siskiyou County	1/10/00	100	0	0	0	0	0	0
San Benito County	1/10/00	24.8	75.2	0	0	0	0	75
Santa Barbara Count	1/10/00	0	100	0	0	0	0	100

Figure 3

Raw NDVI Dataset

County	date	ndvi_value
Alameda County	1/5/00	0.098712023
Alameda County	1/21/00	-0.035884832
Alameda County	2/6/00	0.108218886
Alameda County	2/22/00	0.10464126
Alameda County	3/9/00	0.183525157
Alameda County	3/25/00	0.31148064
Alameda County	4/10/00	0.312395324
Alameda County	4/26/00	0.311211034
Alameda County	5/12/00	0.311181498

Data Usage Mechanisms

Regulations regarding the retention or deletion of data gathered from public sources are not specific. As a result, this data can be used over the long term, such as in the development of machine learning models that can forecast future percentages of drought areas in California. To improve the accuracy of these models, additional data can be added to the dataset. While this information can be freely downloaded from public sources, there are no obligations for sharing it. To properly credit the sources, only citations to the original data sources are included. This ensures that the sources are recognized and acknowledged appropriately. Anyone can access the information directly from the source.

Project Development Methodology

Intelligent System Life Cycle

The hybrid methodology of combining CRISP-DM and Waterfall methodologies has been adopted for the project. This approach is commonly used in data science projects due to its structured approach to planning, execution, and evaluation of machine learning or data mining projects. According to Saltz (2021), this framework is popular among teams because it allows for looping back to a previous stage and defining useful milestones. The loopback process is an

iterative approach that can help reduce the risk of expensive mistakes and ensure the project remains on track. Additionally, the CRISP-DM methodology's structured approach and documentation of all steps can aid in ensuring the project results are reproducible, which is important for both internal and external stakeholders who may need to replicate or build upon them. This framework consists of six frameworks such as

Business Understanding. Project goals and needs are established during this phase. The comprehension of the business issue, the discovery of the data sources, and the specification of the project's parameters are the main concerns.

Data Understanding. This stage involves data collection, exploration, and familiarization with the data. The data is assessed for quality, completeness, and relevance to the business problem. This stage helps to identify any data issues or gaps that need to be addressed.

Data Preparation. In this stage, the data is cleaned, transformed, and formatted to be suitable for analysis. This stage includes tasks such as selecting variables, handling missing data, and dealing with outliers.

Modeling. This step involves applying different modeling methods to the prepared data to produce models that can forecast results or categorize data. The models are evaluated and refined until a satisfactory level of accuracy is achieved.

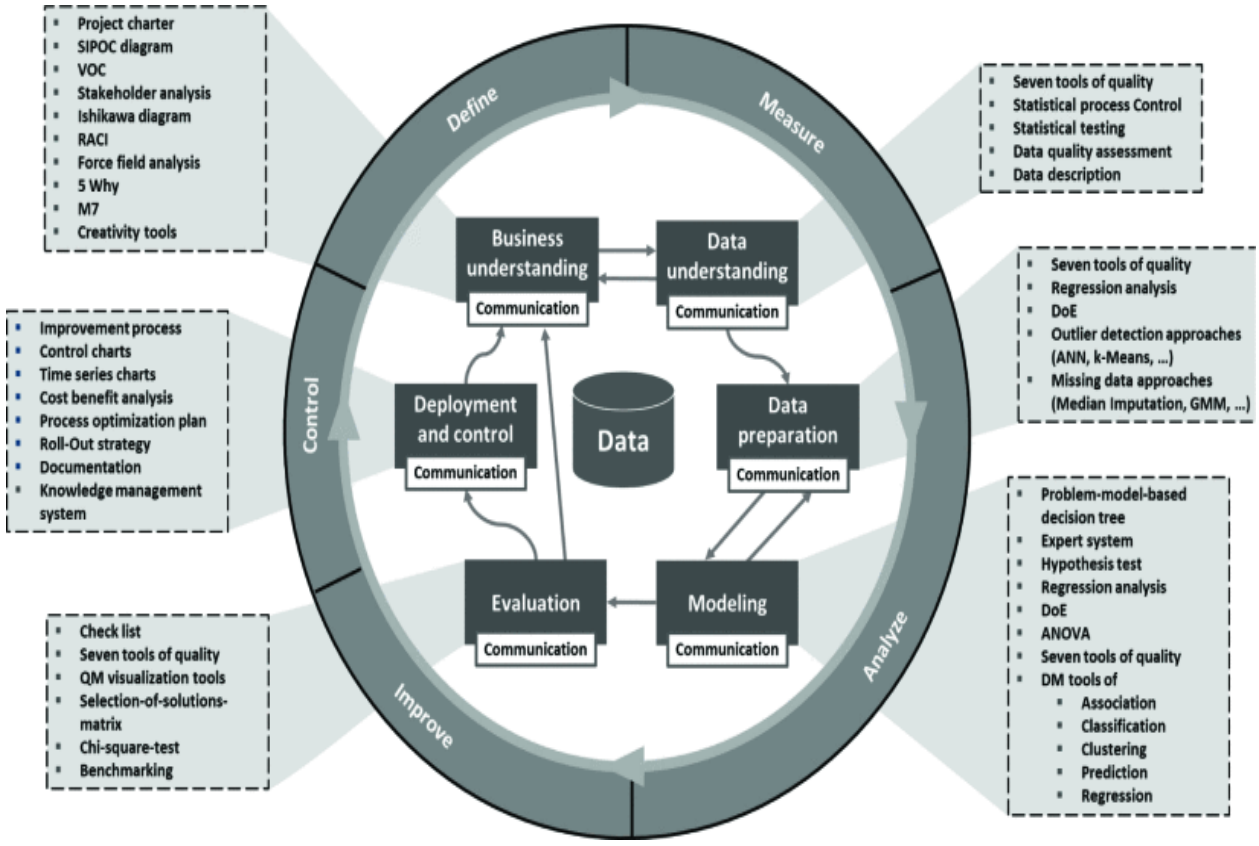
Evaluation. In this stage, the models are evaluated to determine whether they meet the business objectives. The models are compared against each other, and their performance is assessed based on predefined criteria. If the models do not meet the requirements, the process is repeated from the Modeling stage.

Deployment. The models are integrated into the business process. The deployment plan is created, and the models are monitored to ensure they continue to perform as expected.

A detailed explanation of the CRISP-DM Methodology is shown in Figure 4.

Figure 4

CRISP-DM Methodology



Note. CRISP-DM Methodology. From "Synthesizing CRISP-DM and Quality Management: A Data Mining Approach for Production Processes" by Schäfer et al. (2018). *IEEE International Conference on Technology Management, Operations and Decisions (ICTMOD).* (https://doi.org/10.1109/itmc.2018.8691266)

Planned Project Development Processes and Activities

The Scrum methodology was adopted for the project, and JIRA Software was used as the Project Management tool. To guide the work, Epics, Stories, Tasks, and Sub-Tasks were created based on the CRISP-DM framework, with each stage of the project assigned to a different sprint.

The project was planned for six sprints, with five 2-week sprints and one 1-week sprint. Each stage of the CRISP-DM methodology was designated as an Epic. From there, large tasks were created as either a Story or Task, with Sub-Tasks then being created underneath them. The following activities are being performed in each phase of the project:

Business Understanding. The initial phase of the project involves gathering the business requirements. The activities performed in this phase include understanding the problem by identifying the factors that contribute to droughts and determining the importance and relevance of the problem. Evaluating the current technologies and solutions is also a key objective of this phase. In addition, identifying requirements such as obtaining authentic climatic data for California and defining technical and functional requirements are crucial tasks to be carried out. Another important step is to explore existing literature surveys and analyze past solutions and technologies used for drought prediction. Numerous papers and journals have been published worldwide with different approaches and datasets being used. All of the above activities will be performed during this phase, and tasks will be assigned to each team member with the same effort.

Data Understanding. This phase emphasizes the collection and exploration of data to develop a better understanding of it. In this project, the geemap python library is utilized to obtain data from satellite images and compute the vegetation index. Subsequently, climatic data from public websites for California state over the past 20 years is extracted, and the data is combined based on shared features extracted from various websites. Following this, the dataset's missing values are identified, and an outlier analysis is conducted.

Data Preparation. This phase involves cleaning, transforming, and formatting the data to prepare it for analysis and modeling in subsequent steps. The cleaning process includes

handling missing values, inconsistent data, and outliers. Furthermore, various Python libraries will be utilized to transform and format the data. Additionally, data exploration will be conducted to identify similarities and trends within the dataset.

Modeling. In this phase, the dataset will undergo modeling using various machine learning and deep learning algorithms. For the project, the team has decided to use Decision Tree, Random Forest, LSTM, and ANN algorithms to model the data.

Using a decision tree model is a helpful approach for identifying the critical variables or characteristics in drought prediction. By examining the decision rules within the tree, it is possible to determine which variables have the most significant impact on the prediction outcome.

A Random Forest model is a popular machine-learning algorithm for both regression and classification tasks due to its straightforwardness and easy applicability. For this reason, the team has chosen to utilize it in a drought prediction project.

LSTM models are effective for predicting droughts that involve complex, non-linear relationships between variables such as weather patterns, soil moisture, and other factors. This is due to their ability to capture non-linear correlations between variables.

ANN are better suited for predicting droughts because of their proficiency in approximating complex functions and conducting complex time series forecasting. As drought prediction involves numerous factors and the effects of climate change, ANN is a more fitting choice for this type of prediction.

Evaluation. In this phase, the models mentioned above will be assessed and evaluated for their accuracy to determine which algorithm performs the best. Several testing techniques will be employed, including MAE, MSE, and R^2 Score to compare and evaluate each model's

performance. Once the best-performing model has been identified, predictions will be generated using that model to forecast future drought conditions.

Deployment. The integration of the models with the business process occurs during this phase. In the project, the deployment of the model will be implemented using Google Collab, and the resulting predictions will be utilized to inform decision-making.

Project Organization Plan

Work Breakdown Structure

A Work Breakdown Structure (WBS) is a tool of utmost importance as it helps in dividing a complex project into smaller, more manageable sub-tasks. With a WBS, planning and scheduling resources becomes easier, ensuring that the project is completed within the designated timeline. It also enables micro-management of the project by identifying the resources required to accomplish each task, thereby reducing the risks of over and under-allocation of resources. A well-defined WBS provides a framework for monitoring and controlling the project, ensuring that it stays on track and achieves the project objectives.

The WBS plays a crucial role in organizing the business needs, data acquisition, data processing, model selection, model training, model assessment, and deployment stages in this project. To provide a well-structured framework for this project, the CRISP-DM methodology was utilized and each phase of the methodology was assigned as a stage in the WBS structure using the Waterfall model. Each of the six phases in the CRISP-DM methodology corresponded to a stage in the WBS structure, with several sub-tasks allocated equitably among team members, avoiding any over or under-allocation of work.

Figure 5 depicts the different stages of the WBS structure. The first stage, Business Understanding, concentrates on identifying the factors that influence droughts and determining

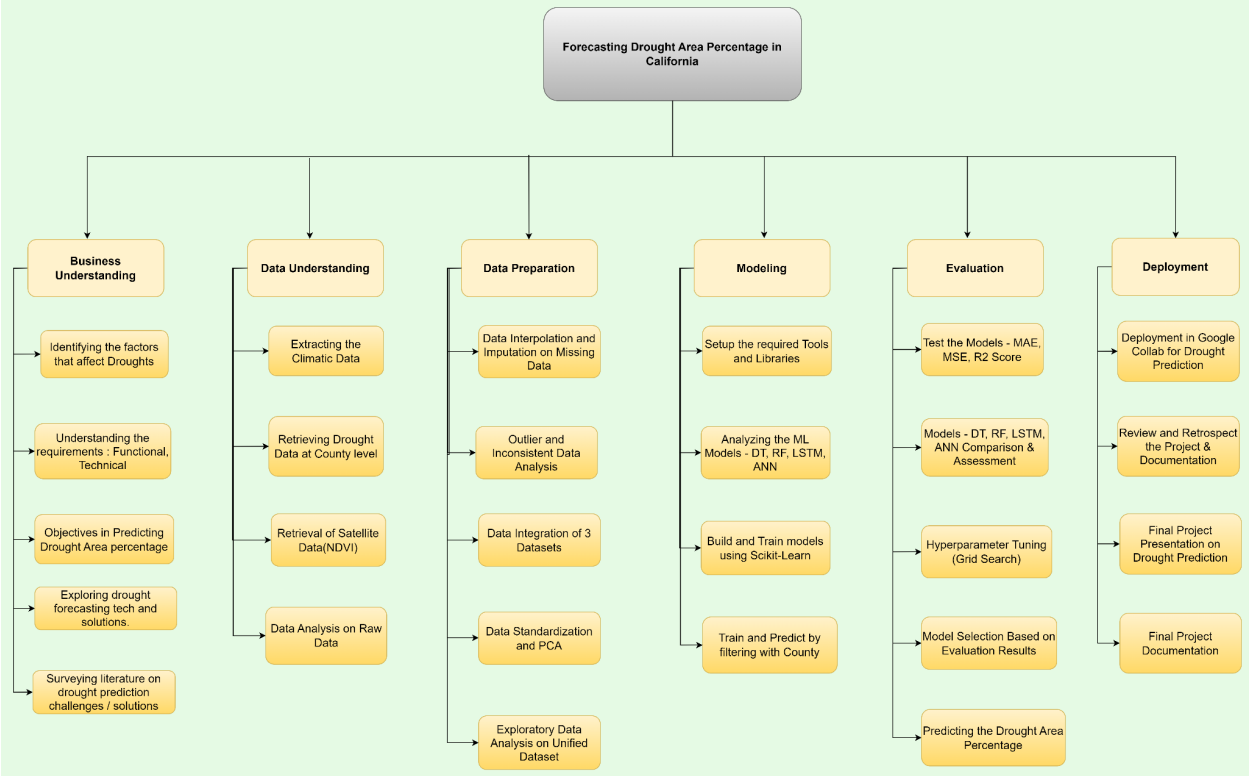
the data requirements, function requirements, technical requirements, and objectives that must be met. The second and third stages are centered around data-related tasks, such as Data Understanding and Data Preparation, which are concerned with how data is collected, extracted, cleaned, and transformed. In this particular project, the data has been collected from three different sources: one website provides climatic data for the past 20 years, another website provides the percentage of drought areas in California over the last 20 years, and the last website provides satellite image data for the past 20 years from which this project calculate vegetation index and soil moisture index. Once the data is gathered from various websites, it will be combined based on common features, and the data accuracy will be calculated. In stage two, all the above-mentioned steps are carried out, and in stage three, the combined dataset will be subject to missing value and outlier identification, as well as data transformation using various Python libraries. Finally, exploratory data analysis will be conducted on the dataset to identify patterns and trends.

Stages four and five of the WBS structure involve the Modeling and Evaluation of models to predict future drought combinations based on the best-performing model. In Stage four, various Machine Learning and Deep Learning models, including Decision Tree, Random Forest, LSTM, and ANN, will be modeled using Jupyter Notebook Building and training models for the training dataset will be carried out using Scikit-Learn in this stage. Stage five involves testing all the models based on different metrics such as MAE, MSE, R^2 score to evaluate the performance of the model. Hyperparameter Tuning will be carried out to find the optimal combinations for the models to avoid overfitting or underfitting the data. After comparing and assessing each model, the best-performing model will be used to predict future drought combinations. In the final stage, the deployment will be done using TensorFlow, and dashboards

will be created using Tableau to visualize the existing data and predicted data. The final project presentation and documentation will be carried out in this stage based on the achieved results and visualization.

Figure 5

Work Breakdown Structure (WBS)



Project Resource Requirements and Plan

Hardware Requirements

The details in Table 3 below give a quick rundown of the gear configurations used. The chart provides information on the equipment's various components, including the RAM, storage, CPU, GPU, and storage space. Overall, the data presented in Table 3 is essential for comprehending the technical details of each gear setup, which is important for choosing the right gear for different jobs.

Table 3

Hardware Requirements of the Project

Hardware	Configuration
CPU Model Name	Apple M1 chip
CPU Frequency	3.2 GHz
No. of CPU Cores	8
GPU	Integrated 8-core GPU
GPU Memory	8 GB
Available RAM Size	8 GB
Disk Space	256 GB

Software Requirements

By utilizing the identical versions mentioned in Table 4, the team has guaranteed consistency in the libraries loaded. Any deviation from these numbers causes the team to use pip to load the necessary version. A complete inventory of software dependencies is provided in Table 4, along with information on their use and version. Understanding the particular libraries needed for the project and their respective functions requires this knowledge. It is crucial to keep in mind that as the project advances, the requirements might alter, necessitating modifications to the existing needs. These modifications might involve introducing new modules, eliminating superfluous ones, or upgrading the versions of already existing ones. By making these changes, the project is kept current and pertinent to its original objective.

Table 4

Libraries Used in Project.

Packages / Libraries	Motive	Version
Python	Data Cleaning, Data Preparation, and Data Analysis	3.8
Pandas	Create Dataframe	1.5.3
Numpy	Dealing with Dataframe arrays	1.24.2
ee	Earth Engine python API	0.1.347
folium	Access to interactive leaflet Map	0.14.0
Matplotlib	Data Interpretation using plots	3.7.1
Seaborn	Advance Interactive plots	0.12.2
scikit-learn	Building Machine Learning Model	1.2.2
keras	Developing DL models	2.12.0
tensorflow	Collection of ML Models and Metrics	2.10.0

A summary of the databases used in the undertaking and their various uses are shown in Table 5. knowing the particular databases used and how they add to the project's general goals requires knowing this list. It is crucial to remember that the inventory of databases used may alter as the project develops and its purview shifts. This might entail creating new datasets, eliminating outdated ones, or changing current ones to better meet the project's changing needs. The project team can respond to new obstacles and shifting conditions by keeping the database specifications flexible, which will help to ensure the project's effective completion.

Table 5

Databases Used in Project.

Databases / Warehouse	Motive	Version
MySQL Workbench	Dataset Modification and Storage	8.0.30

A summary of the user interface programs used in the project is shown in Table 6, along with information about their individual uses and versions. Understanding the particular visualization software used and how it contributes to the project's overall goals requires this knowledge. It is significant to note that the specifications for the user interface software may alter as the project develops and its purview shifts. This might entail incorporating new visualization software, getting rid of outdated programs, or altering current ones to better meet the project's changing needs. The project team can respond to new challenges and shifting conditions by keeping the requirements for the user interface software flexible. This will help to ensure the project's effective conclusion.

Table 6

Visualization Software Used in Project.

Visualization Software	Motive	Version
Tableau	Data Visualization	2022.4

Tools and Licenses

A complete inventory of the tools being used in the project is provided in Table 7, along with information on their individual uses, licenses, and expiration dates. Understanding the particular tools used and how they add to the project's overall goals requires this knowledge. It is

significant to observe that the tool needs may alter as the project develops and its purview shifts. This might entail incorporating new tools, eliminating obsolete ones, or changing current ones to better meet the project's changing needs. The project team can respond to new obstacles and shifting conditions by keeping the tool specifications flexible, which will help to ensure the project's effective completion. It is crucial to review the table regularly to ensure that the tools being used are up-to-date and compliant with their respective licenses. The validity of licenses is also an important factor to consider, and any expired licenses should be renewed or replaced promptly.

Table 7

Tools and Licenses for Project.

Application Tools	Motive	License
Jupyter Notebook	Code	Open Source
Jira	Maintain and Monitor Project Workflow using WBS, Gantt, PERT	Free
GitHub	Repository	Open Source
Zoom	For team collaboration	Free
Google Meet	For team collaboration	Free
Draw.io	For designing	Free
Google Docs	Documentation	Free
Grammarly	For checking plagiarism and grammatical mistakes.	Paid

Table 8 which shows the estimated cost of a MacBook laptop with OS 13.1, the team has been using in the project.

Table 8

Project Cost and Justification.

Product	OS Version	Cost
MacBook Air	13.1	$400

Project Schedule

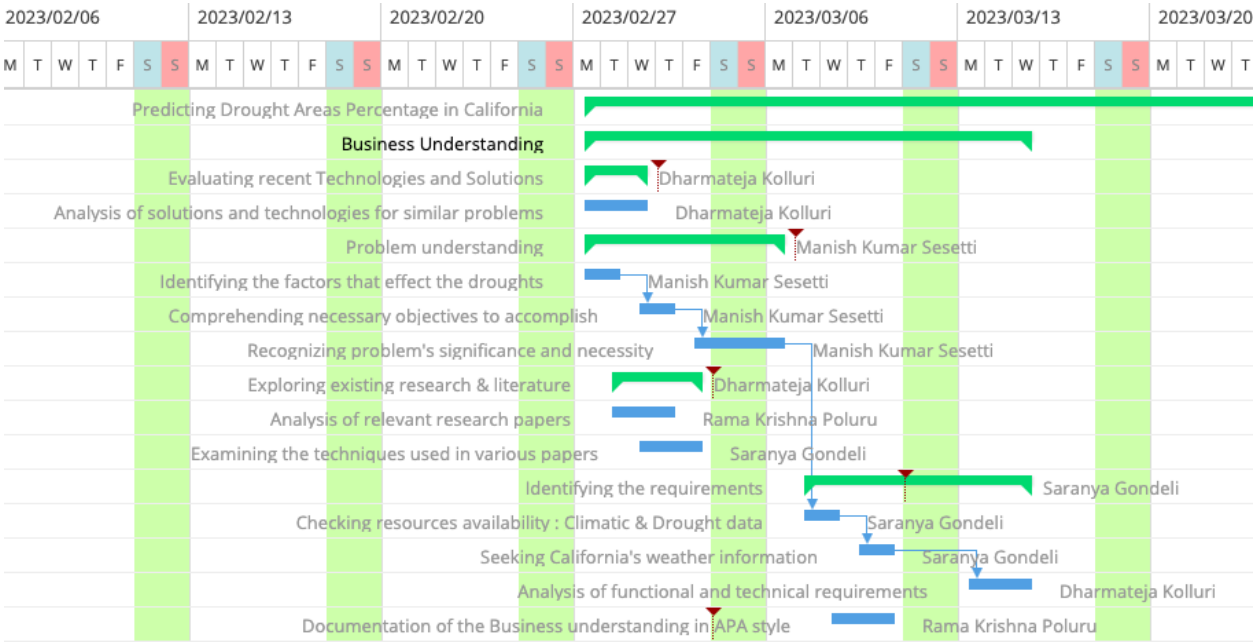
Gantt Chart

According to the Figure 6-11 presented, the utilization of a Gantt Chart facilitates the identification of all necessary tasks and their corresponding dependencies, ensuring the maintenance of the project's scope, the definition of the timeframe, and provision of additional clarity. The interdependence of the tasks serves as a reminder for completing prerequisites when undertaking a specific task. The distribution of project tasks is evenly allocated among the team and the due dates for completion fall within the range of one to three days.

The Gantt chart is divided into six phases, each containing tasks, subtasks, and dependencies among themselves. Coming to the first phase, which is *Business understanding*, Figure 6 shows the tasks that play a major role in the project development. The business understanding phase is from 02/27/2023 to 03/15/2023. The major tasks involved in this phase are Problem Understanding, Evaluating Recent Technologies and Solutions, Identifying the Requirements, Exploring Existing Research & Literature, and Documentation of the Business Understanding. Each of these tasks will have subtasks that will allow the team to work at a minute level for the project's progress.

Figure 6

Gantt Chart - Business Understanding Phase



The following Table 9 gives the status of the deliverables involved in the first phase. The tasks are owned by responsible members, but the team and the individual will work on subtasks based on what is assigned.

Table 9

Business Understanding Phase Tasks

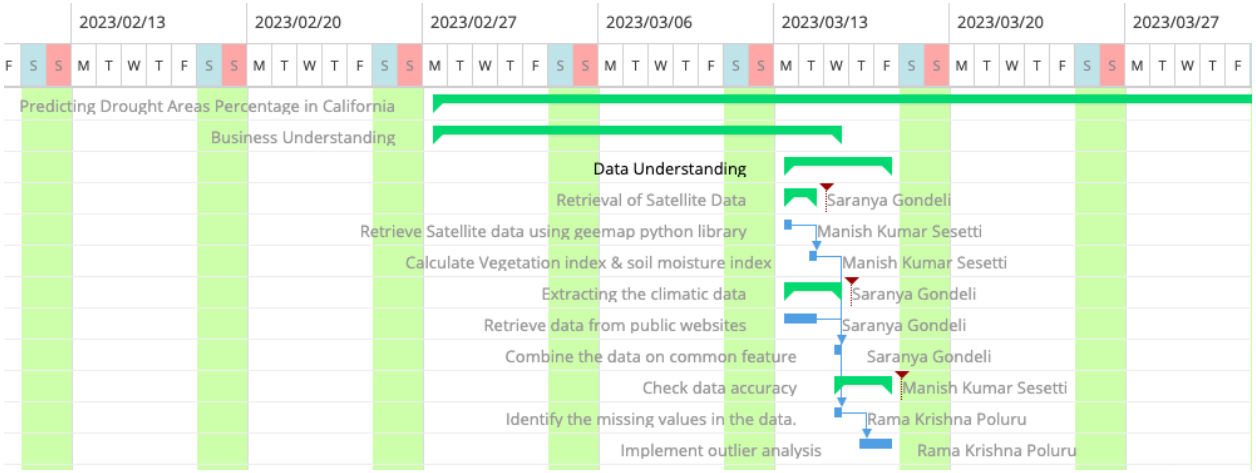
Task	Timeline	Responsible Member	Status of the deliverable
Problem understanding	02/27/2023 - 03/06/2023	Manish Kumar Sesetti, Dharma Teja Kolluri, Saranya Gondeli, Rama Krishna Poluru	Done

Task	Timeline	Responsible Member	Status of the deliverable
Evaluating recent Technologies and Solutions	02/27/2023 - 03/01/2023	Dharma Teja Kolluri	Done
Identifying the requirements	03/07/2023 - 03/15/2023	Saranya Gondeli, Rama Krishna Poluru	Done
Exploring existing research & literature	02/28/2023 - 03/03/2023	Dharma Teja Kolluri, Rama Krishna Poluru	Done
Documentation of the Business understanding	03/08/2023 - 03/10/2023	Rama Krishna Poluru	Done

The second phase is Data understanding. Figure 7 shows the tasks that play a major role in this phase. The data understanding phase is from 03/13/2023 to 03/17/2023. The major tasks involved in this phase are the retrieval of satellite data, extraction of climatic data, and checking data accuracy. Each of these tasks will have subtasks that will allow the team to work at a detailed level to progress the project.

Figure 7

Gantt Chart - Data Understanding Phase



The following Table 10 provides the status of the deliverables involved in the second phase. The tasks are owned by responsible members, but the team and individuals will work on subtasks based on assigned responsibilities.

Table 10

Data Understanding Phase Tasks

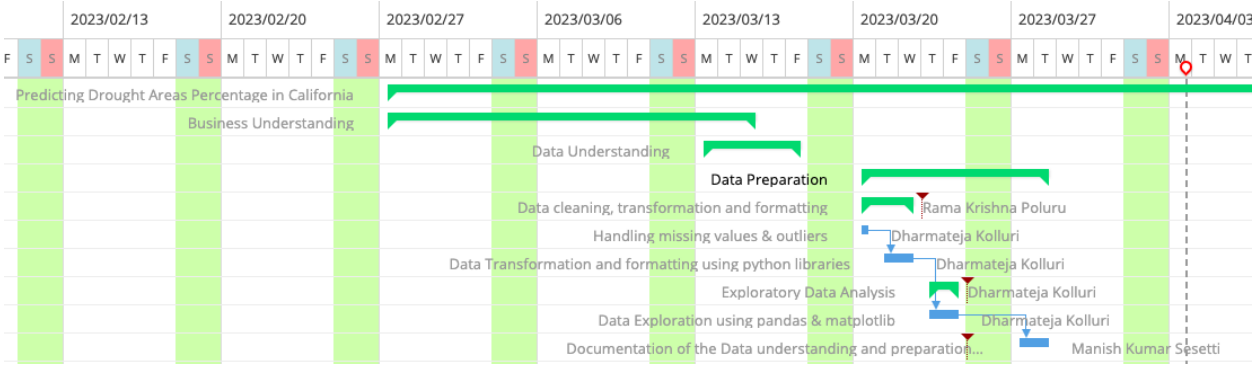
Task	Timeline	Responsible Member	Status of the deliverable
Retrieval of Satellite Data	03/13/2023 - 03/14/2023	Saranya Gondeli	Done
Extracting the climatic data	03/13/2023 - 03/15/2023	Saranya Gondeli	Done

Task	Timeline	Responsible Member	Status of the deliverable
Check data accuracy	03/13/2023 - 03/17/2023	Manish Kumar Sesetti	Done

The third phase is Data preparation. Figure 8 shows the tasks that play a major role in this phase. The data preparation phase is from 03/20/2023 to 03/28/2023. The major tasks involved in this phase are documentation of the data understanding and preparation, data cleaning, transformation and formatting, and exploratory data analysis. Each of these tasks will have subtasks that will enable the team to work at a detailed level for the project's progress.

Figure 8

Gantt Chart - Data Preparation Phase



The following Table 11 provides the status of the deliverables involved in the third phase. The tasks are owned by responsible members, but the team and individuals will work on subtasks based on assigned responsibilities.

Table 11

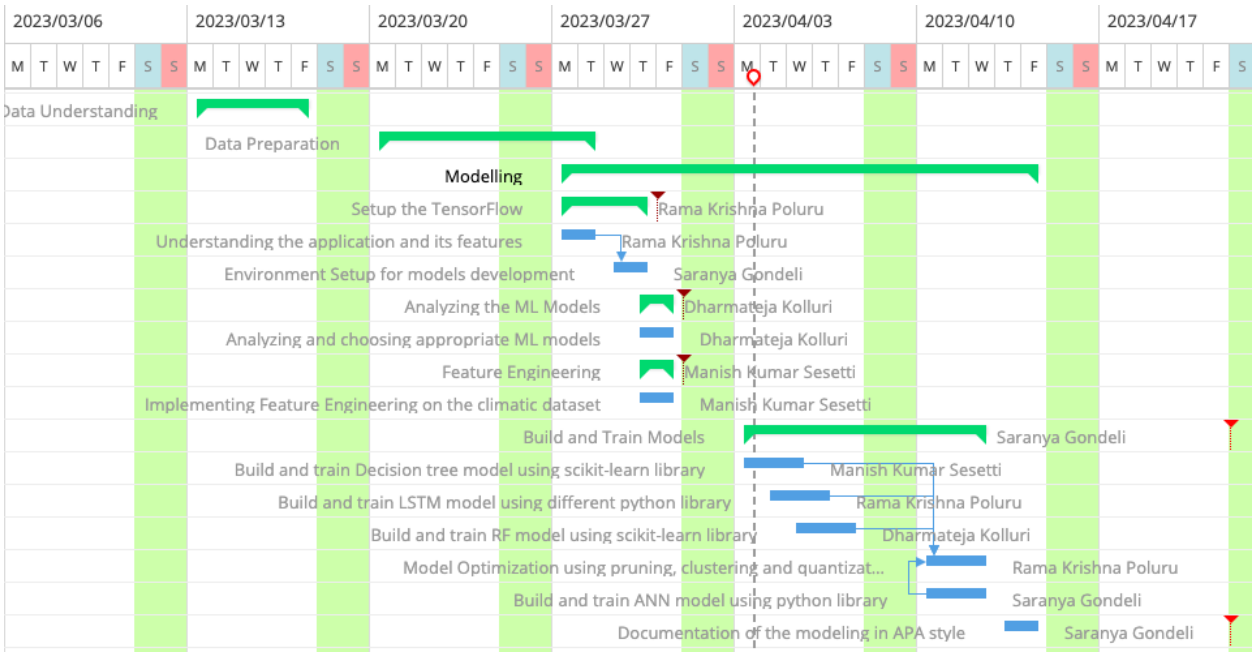
Data Preparation Phase Tasks

Task	Timeline	Responsible Member	Status of the deliverable
Documentation of the Data understanding and preparation	03/27/2023 - 03/28/2023	Manish Kumar Sesetti	Done
Data cleaning, transformation and formatting	03/20/2023 - 03/22/2023	Rama Krishna Poluru	Done
Exploratory Data Analysis	03/23/2023 - 03/24/2023	Dharma Teja Kolluri	Done

The fourth phase is Modeling. Figure 9 shows the tasks that play a major role in this phase. This is the key phase of the entire project, which involves building and training ML models. The modeling phase is from 03/27/2023 to 04/14/2023. The major tasks involved in this phase are setting up TensorFlow, analyzing the ML models, building and training models, feature engineering, and documenting the modeling. Each of these tasks will have subtasks that will enable the team to work at a detailed level for the project's progress.

Figure 9

Gantt Chart - Modeling Phase



The following Table 12 provides the status of the deliverables involved in the fourth phase. The tasks are owned by responsible members, but the team and individuals will work on subtasks based on assigned responsibilities.

Table 12

Modeling Phase Tasks

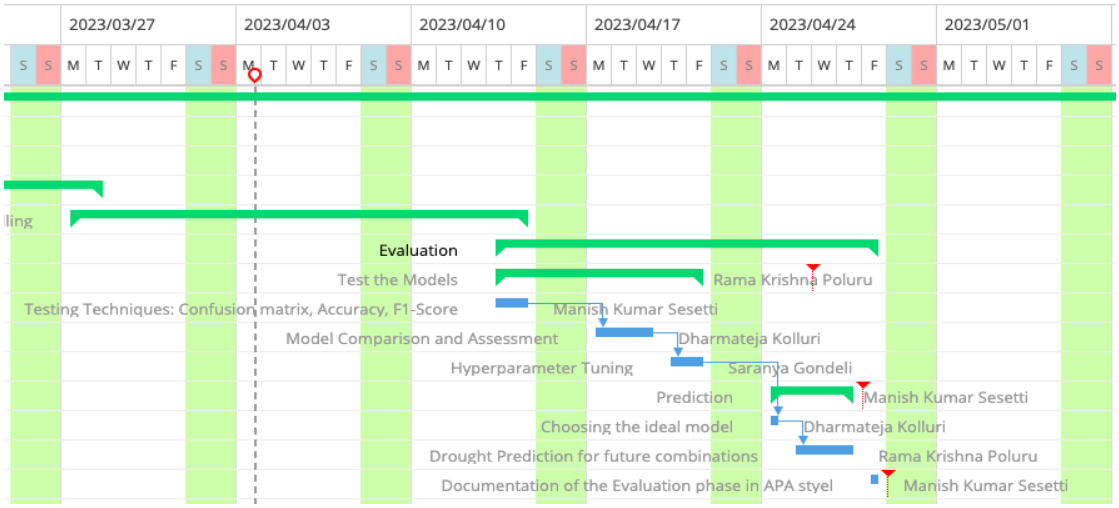
Task	Timeline	Responsible Member	Status of the deliverable
Setup the TensorFlow	03/27/2023 - 03/30/2023	Rama Krishna Poluru	Done
Analyzing the ML Models	03/30/2023 - 03/31/2023	Dharma Teja Kolluri	Done

Task	Timeline	Responsible Member	Status of the deliverable
Build and Train Models	04/03/2023 - 04/12/2023	Saranya Gondeli	In Progress
Feature Engineering	03/30/2023 - 03/31/2023	Manish Kumar Sessetti	To Do
Documentation of the modeling	04/13/2023 - 04/14/2023	Saranya Gondeli	To Do

The fifth phase is Evaluation. Figure 10 shows the tasks that play a major role in this phase. The evaluation phase is from 03/13/2023 to 04/28/2023. The major tasks involved in this phase are testing the models, making predictions, and documenting the evaluation phase. Each of these tasks will have subtasks that will enable the team to work at a detailed level for the project's progress.

Figure 10

Gantt Chart - Evaluation Phase



The following Table 13 provides the status of the deliverables involved in the fifth phase. The tasks are owned by responsible members, but the team and individuals will work on subtasks based on assigned responsibilities.

Table 13

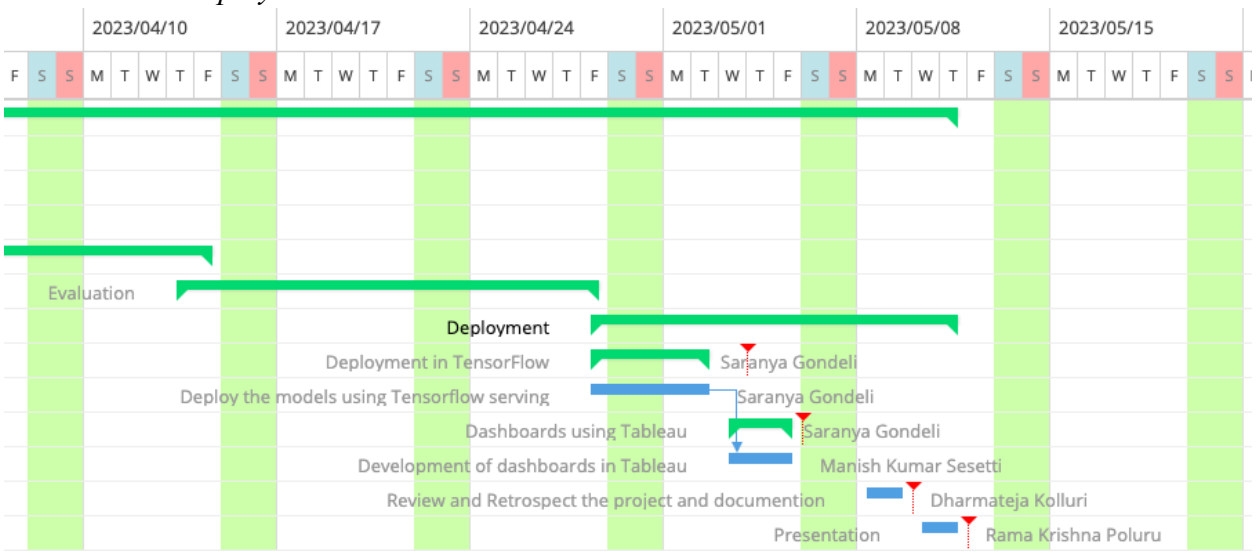
Evaluation Phase Tasks

Task	Timeline	Responsible Member	Status of the deliverable
Test the Models	04/13/2023 - 04/21/2023	Rama Krishna Poluru	To Do
Prediction	04/24/2023 - 04/27/2023	Manish Kumar Sessetti	To Do
Documentation of the Evaluation phase	04/28/2023 - 04/28/2023	Manish Kumar Sessetti	To Do

The final phase is Deployment. Figure 11 shows the tasks that play a major role in this phase. The deployment phase is from 04/28/2023 to 05/11/2023. The major tasks involved in this phase are deployment in TensorFlow, creating dashboards using Tableau, reviewing and retrospecting the project and documentation, and presentation. Each of these tasks will have subtasks that will enable the team to work at a detailed level for the project's progress.

Figure 11

Gantt Chart - Deployment Phase



The following Table 14 provides the status of the deliverables involved in the sixth phase. The tasks are owned by responsible members, but the team and individuals will work on subtasks based on assigned responsibilities.

Table 14

Deployment Phase Tasks

Task	Timeline	Responsible Member	Status of the deliverable
Deployment in TensorFlow	04/28/2023 - 05/02/2023	Saranya Gondeli	To Do
Dashboards using Tableau	05/03/2023 - 05/05/2023	Manish Kumar Sessetti	To Do

Task	Timeline	Responsible Member	Status of the deliverable
Documentation of the Evaluation phase	05/03/2023 - 05/05/2023	Dharma Teja Kolluri	To Do
Presentation	05/10/2023 - 05/11/2023	Rama Krishna Poluru	To Do

Pert Chart

The PERT (Program Evaluation and Review Technique) chart was created using the Lucid chart application to aid in understanding and visualizing critical path analysis (CPA), which is the primary and vital task of the work. PERT enables the determination of a project's critical path analysis (CPA) and non-critical path tasks. Critical path tasks in a project are the most significant and essential tasks that must be completed to proceed with the work.

The CPA for the project entails determining the reason and likelihood of solving the business problem, conducting a literature review, allocating resources, acquiring data, constructing a model, tuning hyperparameters, validating the model, and deploying it in a production environment. To guarantee the project's timely completion, these tasks must be finished within a set time limit.

The non-critical path tasks with feature engineering and selecting the most accurate and reliable model are important but do not have a significant impact on the completion of the project. Without affecting the project's overall schedule, these jobs can be finished later. Before

the final step of merging the code and deploying the models in the production environment, these steps must be finished, nevertheless.

The critical path in the project starts with establishing the motivation and possibility of solving the business problem, followed by addressing unsolved ways for real-world problems, conducting a literature review and technological survey, and allocating resources for project execution. The next step is gathering historical weather data, vegetation, soil, and SPEI index data, followed by combining all datasets and performing data understanding and preparation. The critical path then moves to performing exploratory data analysis to gain insights into the data and selecting relevant features for prediction through feature engineering. The next critical step is to analyze and build machine learning models for forecasting drought area percentage in California. This includes hyperparameter tuning and validating models with data to compare accuracy and performance of all models.

Finally, the critical path includes merging the final development code to the master branch, deploying the models on the production environment, and preparing the project report and PPT for presentations. The non-critical path includes selecting the most accurate and reliable model, which is joined with the final step.

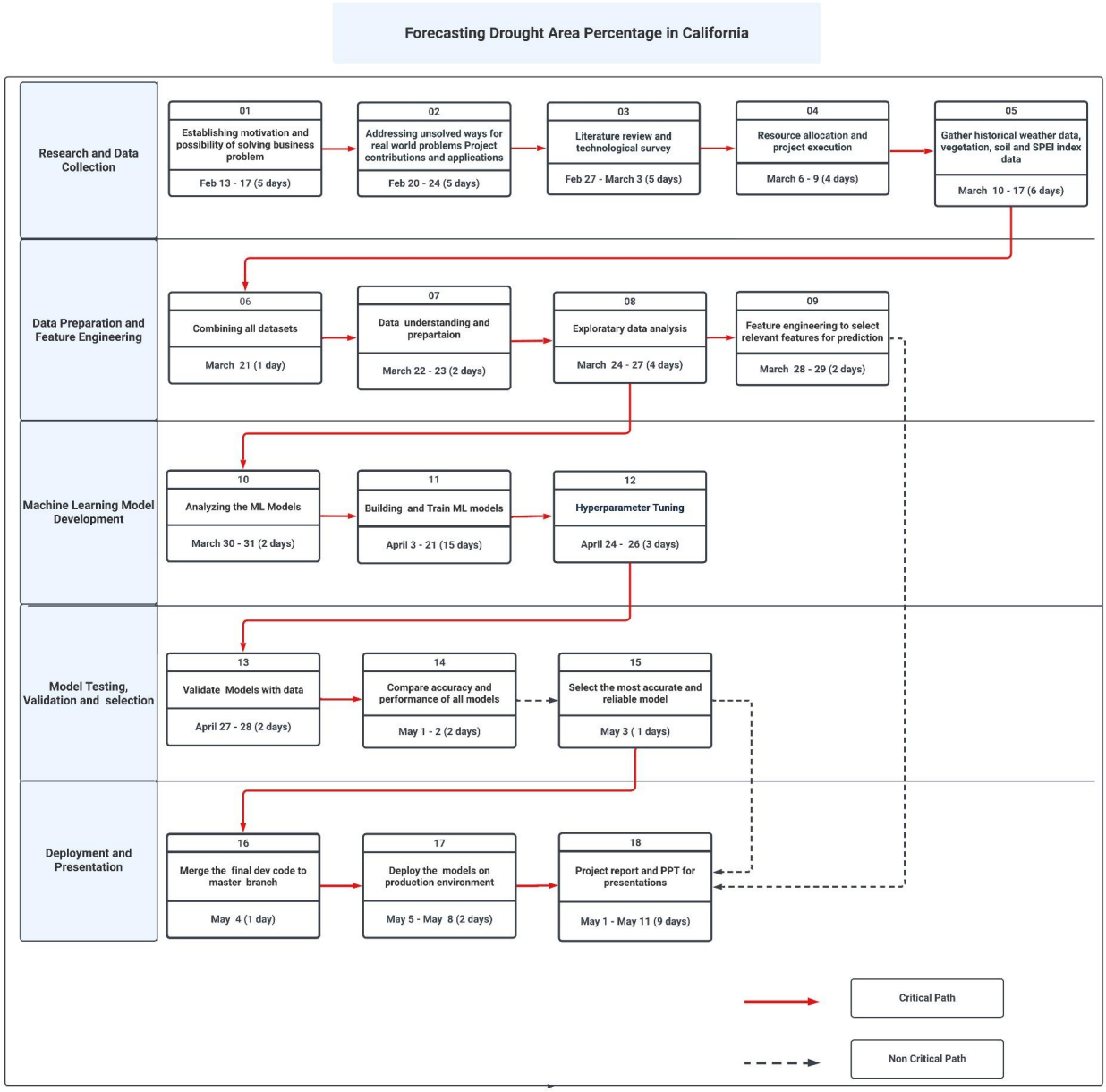
Figure 12 shows the PERT chart where the boxes with the three fields contain the following information about the PERT chart. First, it contains the task number, and second the task description which tells you what one needs to achieve through the task. Finally, the last segment involves the start and end dates of the task where it gives a timeframe for the task to be completed. The arrows that connect the tasks without any break in the line indicate the critical path of the project and the dependencies of the task on how they should be worked upon. The

arrows with breaks in the lines when they are connected to tasks indicate the non-critical path. Like the critical path, the arrow of the non-critical path defines dependencies.

The team needs to consult the existing technologies and solution survey for a different strategy if any problems with the model outcome develop. The PERT chart assists in identifying task dependencies and crucial activities, ensuring the project is finished on schedule. The team can gain knowledge of critical path analysis (CPA), the most significant and crucial task without which they are unable to continue working, by adopting a task-based approach.

Figure 12

PERT Chart - Forecasting Drought Area Percentage in California



Data Engineering

Data Process

This project's goal is to study climatic data over the last 20 years to discover drought patterns and anticipate the percentage of each county in California that may experience drought conditions. Climate data and area percentage statistics were gathered from a variety of sources, including the NASA Power website, which provided daily climate data for each county in California, including temperature, humidity, wind speed, and precipitation, among other things. The Google Earth Engine for computing NDVI values for each county using LANDSAT satellite images, as well as the US Drought Monitor website, which contains weekly and monthly statistics on the area percentages of each county in California that is affected by drought conditions. For further analysis, the data were combined into one dataset based on the county and time dimension. But following data collection, the data was verified for consistency and accuracy, and several issues such as missing values, inconsistent formatting, and outliers were discovered.

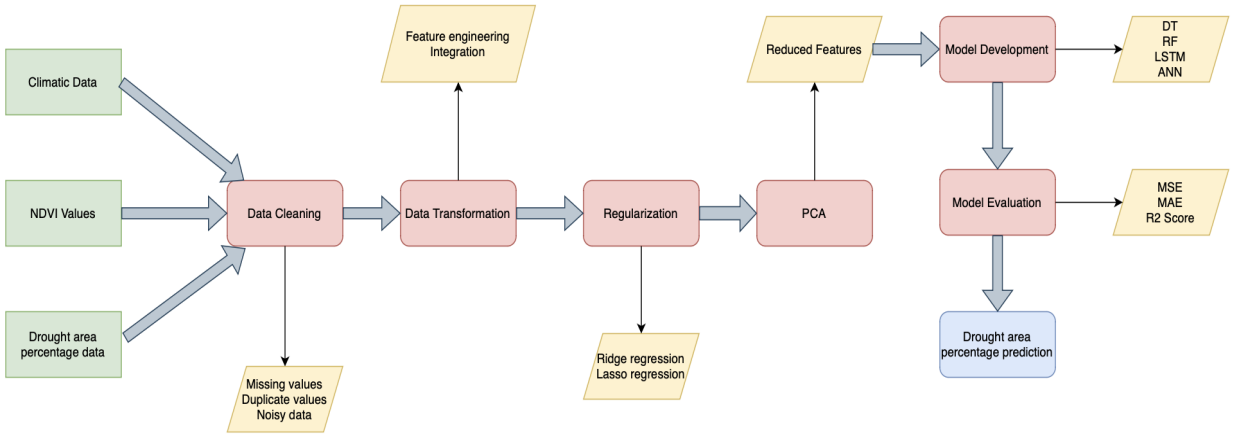
To make the data suitable for analysis, a method was used that involved creating a date range covering the entire dataset and filling in any missing values by using weekly values. For each week, the drought area percentage value was assigned to all dates in that week to ensure consistency in handling missing data across all counties and years. Additionally, irrelevant or duplicate data was removed, and outliers beyond the interquartile range were eliminated. When a machine learning model fits the training data too closely, it can lead to overfitting, which is characterized by poor generalization to new data. To overcome this issue, regularization techniques such as ridge and lasso regression are utilized to alter the model's complexity.

The dataset was subsequently reduced in dimensionality by using principal component analysis (PCA) to determine the most critical features contributing to drought severity. PCA is a useful approach for finding the essential variables and lowering the dataset's dimensionality. And determined the appropriate number of principal components to use for further analysis by generating two plots, the cumulative explained variance plot and the scree plot.

The dataset was then divided into three sections: training, validation, and testing. The training set included prior-year data, whereas the test set included data from the most recent year. The training set was used to train the machine learning models, while the validation set was used to tweak the hyperparameters and evaluate the models performance. Experimented with several machine learning algorithms, including Random Forest, Decision Tree ensemble with XGBoost, LSTM, and ANN to predict the area percentage of each county in California that may face drought conditions. Figure 13 shows the flow of the project.

Figure 13

Flow of the Project



Data Collection

Sources

Data has been collected from three sources, including the climatic data of California, the Earth Engine Data Catalog, and the Drought area percentage dataset of California, all of them retrieved from official websites covering the period from 2000 to 2020. The climatic variables and Drought area percentages have been sourced from the following websites respectively:

1. The data for climatic variables from January 2000 to December 2020 has been obtained from the website https://power.larc.nasa.gov/data/.This data comprises daily values covering the entire period from 2000 to 2020.
2. The data on drought area percentage in California from January 2000 to December 2020 has been sourced from the website https://droughtmonitor.unl.edu/DmData/DataTables.aspx?state,ca. This data is available on a weekly basis and has been retrieved for each county in California.
3. The data of NDVI values for California from January 2000 to December 2020 has been sourced from the Earth Engine Data Catalog using Python API by importing the ‘ee’ package in Python.

The data collection plan for the climatic dataset is shown in Figure 14, including all the necessary details based on the dataset features. The collection plan specifies key variables as many variables available in the dataset don’t have much significance to calculate SPEI or drought area percentage. The dataset has a total of 3 Million records of numerical time series climatic variables such as Temperature, Specific humidity, Precipitation, Wind Speed at 10 meters, etc. This dataset has historical data from 2000 to 2020 with 20 climatic variables used to calculate the SPEI value. The dataset includes latitude and longitude values, along with date

values in separate columns, which are not practical for Pandas operations. To calculate the SPEI values and drought area percentages at the county level instead of latitude and longitude, the next step is to combine the date columns into a single column and obtain county names based on the latitude and longitude values. After obtaining the county names, the dataset will be preprocessed to address any missing, inconsistent, or noisy data.

Figure 14

Data Collection Plan for the Climate Dataset

Description of the data collection									
The dataset for this project is being sourced from the NASA Power website in the CSV format. The data pertains to continuous climatic observations for the state of California spanning the period from 2000 to 2020. The dataset comprises various numerical data fields such as latitude, longitude, date, minimum temperature, maximum temperature and so on. This dataset will be integrated with other datasets to create a comprehensive model for analysis									
What will be done with the data once it has been collected?									
	Identify how well the process is meeting **customer** needs			Analyze a **problem**, or identify the causes of variation					
Yes	Obtain **exploratory view** of the process			Test a **hypothesis** about the process output					
	Evaluate the **measurement system**			Test a **hypothesis** about the effects of one or more inputs					
	Check the **stability** of the process (is it in control?)			**Control** a process input or monitor a process output					
	Conduct a **capability** study			**Other reason…**					

Key Variables - A summary of the chosen input variables (Y's) and/or output variables (X's)										
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
What?	**Variable title**	LAT	LON	YEAR	MO	DY	T2M	QV2M	PRECTOTCORR	WS10M
	(X) or output (Y) variable?	X	X	X	X	X	X	X	X	X
	Unit of measurement	Degrees	Degrees	Year	Month	Day	Celsius	g/kg	mm/day	m/s
	Data type	object	object	int64	int64	int64	float	float	float	float
	Collection method	We extracted this data from NASA power website in a CSV format. This dataset contains total of 3068000 records and 20 attributes. Total size of the data is 309 MB								
Historical data	Historical data exist?	This Dataset contains total of 20 attributes from the year 2000 to 2020 with all the climatic data that includes latitude and longitude values.								
	Source of historical data	https://power.larc.nasa.gov/data/								
	lata representative/reliable?	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	Mean	-	-	-	-	-	13.352	5.63	1.05	3.86
	Upper specification limit	-	-	-	-	-	41.73	18.59	128.4	17.97
	Lower specification limit	-	-	-	-	-	-22.32	0.4	0	0.52
	Standard deviation	-	-	-	-	-	8.66	2.61	3.95	2.08
Who?	**Data collector**	**Saranya Gondeli**								
	)perational definition exist?	In this dataset, all the variables are considered as numerical or float values and the values remain unchanged throughout the process except date which is changed to date format. Some of the data like T2M means temperature at 2 meters, QV2M is specific humidity at 2 meters, PRECTOTCORR means total precipitation and WS10M means wind direction at 10 meters. All the remaining columns are self-exploratory.								
When?	**Start date**	31-Mar	31-Mar	31-Mar	31-Mar	31-Mar	31-Mar	31-Mar	31-Mar	31-Mar
	Due date	1-Apr	1-Apr	1-Apr	1-Apr	1-Apr	1-Apr	1-Apr	1-Apr	1-Apr
	Duration (in days)	1	1	1	1	1	1	1	1	1

Figure 15 displays the data collection plan for the drought area percentage dataset, which includes all the necessary details based on the features of the dataset. The collection plan specifies only the variables required for modeling the data, omitting other variables such as the values of columns ‘None’, ‘D1-D4’, ‘D2-D4’, ‘D3-D4’, ‘D4’, and ‘DSCI’. These variables represent the severity of drought, with D0 indicating mild conditions and D4 indicating extreme drought. This dataset contains weekly data from 2000 to 2020 for 51 counties in California. The

dataset has a total of 55K records and the data provides the drought area percentage for each county every week. However, it is not feasible to interpolate the drought area percentage for the entire week, so the next step for this dataset is to populate the data for each day based on the drought percentage for that week. After the daily data is obtained, the dataset will be preprocessed to address any missing, noisy, or inconsistent data. Then, the data will be integrated with other datasets based on the county names and dates available in the dataset.

Figure 15

Data Collection Plan for Drought Area Percentage Dataset

Description of the data collection				
The dataset pertains to the Drought area percentage in California's counties obtained from the drought monitor website covering the time period from 2000 to 2020. The dataset encompasses several numeric data fields, which indicate the intensity of drought area percentages across California. The dataset will be merged with other datasets based on the County name and date.				
What will be done with the data once it has been collected?				
	Identify how well the process is meeting **customer** needs		Analyze a **problem**, or identify the causes of variation
Yes	Obtain **exploratory view** of the process		Test a **hypothesis** about the process output
	Evaluate the **measurement system**		Test a **hypothesis** about the effects of one or more inputs
	Check the **stability** of the process (is it in control?)		**Control** a process input or monitor a process output
	Conduct a **capability** study		**Other reason…**

Key Variables - A summary of the chosen input variables (Y's) and/or output variables (X's)							
		1	**2**	**3**	**4**	**5**	**6**
What?	**Variable title**	County	Date	D0-D4			
	(X) or output (Y) variable?	X	X	Y			
	Unit of measurement	String	Date	Percentage			
	Data type	Object	Datetime	Flaot			
	Collection method	We extracted this data from drought monitor website in a CSV format. This dataset contains total of 55845 records and 9 attributes. Total size of the data is 28 MB					
Historical data	Historical data exist?	This Dataset contains total of 9 attributes from the year 2000 to 2020 with all the drought area percentage data for each county in California					
	Source of historical data	https://droughtmonitor.unl.edu/DmData/DataTables.aspx?state,ca					
	al data representative/reliable?	Yes	Yes	Yes			
	Mean	-	-	56.633357			
	Upper specification limit	-	12/31/22	100			
	Lower specification limit	-	1/1/00	0			
	Standard deviation	-	-	46.15519			
Who?	**Data collector**	**Dharma Teja Kolluri**					
	Operational definition exist?	In this dataset, county names are given in a column, Date has been taken as one column and D0-D4 is the drought area percentage that varies from mild to severe drought conditions.					
When?	**Start date**	2-Apr	2-Apr	2-Apr			
	Due date	3-Apr	3-Apr	3-Apr			
	Duration (in days)	1	1	1			

Figure 16 illustrates the data collection approach for the NDVI dataset, which outlines all the essential information based on the dataset features. The data collection plan specifies the relevant variables necessary for modeling the data such as County, Date, and NDVI values. These variables represent the different bands of the satellite image which are retrieved using

Python API from the Earth Engine Data Catalog. This dataset comprises data from 2000 to 2020 for 51 counties in California, with a total of 220K records providing the NDVI values for each county.

Figure 16

Data Collection Plan for NDVI Dataset

Description of the data collection							
This dataset depicts the NDVI values of California state with Dates from 2000 to 2020. We obtained this dataset from the Earth Engine Data Catalog and this data could be pulled using the python API by directly accessing the data without downloading them and feeding them to the code.							
What will be done with the data once it has been collected?							
	Identify how well the process is meeting **customer** needs			Analyze a **problem**, or identify the causes of variation			
Yes	Obtain **exploratory view** of the process			Test a **hypothesis** about the process output			
	Evaluate the **measurement system**			Test a **hypothesis** about the effects of one or more inputs			
	Check the **stability** of the process (is it in control?)			**Control** a process input or monitor a process output			
	Conduct a **capability** study			**Other reason…**			

Key Variables - A summary of the chosen input variables (Y's) and/or output variables (X's)							
		1	**2**	**3**	**4**	**5**	**6**
What?	**Variable title**	County	Date	NDVI			
	(X) or output (Y) variable?	X	X	X			
	Unit of measurement	String	Date	Float			
	Data type	Object	Datetime	Float			
	Collection method	We extracted this data from Google Earth Engine using an API to get the values in a CSV format. This dataset contains total of 220830 records and 3 attributes. Total size of the data is 17 MB					
Historical data	Historical data exist?	This Dataset contains total of 3 attributes from the year 2000 to 2020 with NDVI data for each county in California.					
	Source of historical data						
	al data representative/reliable?	Yes	Yes	Yes			
	Mean	-	-	0.006333			
	Upper specification limit	-	12/29/20	0.34513			
	Lower specification limit	-	1/1/00	-0.60146			
	Standard deviation	-	-	0.140868			
Who?	**Data collector**	**Ramakrishna Poluru**					
	Operational definition exist?	In this dataset, all the column names are self-exploratory and can be understood from the dataset.					
When?	**Start date**	4-Apr	4-Apr	4-Apr			
	Due date	5-Apr	5-Apr	5-Apr			
	Duration (in days)	1	1	1			

Dataset Samples

Following are the raw dataset samples that have been used in this project. Figure 17 shows the raw climatic dataset sample which consists of descriptions of data columns.

Figure 17

Raw Climatic Dataset Sample

-BEGIN HEADER-
NASA/POWER CERES/MERRA2 Native Resolution Daily Data
Dates (month/day/year): 09/02/2006 through 12/31/2006
Location: Regional
Elevation from MERRA-2: Average for 0.5 x 0.625 degree lat/lon region = na meters
The value for missing source data that cannot be computed or is outside of the sources availability range: -999
Parameter(s):
T2M MERRA-2 Temperature at 2 Meters (C)
T2MDEW MERRA-2 Dew/Frost Point at 2 Meters (C)
TS MERRA-2 Earth Skin Temperature (C)
T2M_RANGE MERRA-2 Temperature at 2 Meters Range (C)
T2M_MAX MERRA-2 Temperature at 2 Meters Maximum (C)
T2M_MIN MERRA-2 Temperature at 2 Meters Minimum (C)
QV2M MERRA-2 Specific Humidity at 2 Meters (g/kg)
RH2M MERRA-2 Relative Humidity at 2 Meters (%)
PRECTOTCORR MERRA-2 Precipitation Corrected (mm/day)
PS MERRA-2 Surface Pressure (kPa)
WS10M MERRA-2 Wind Speed at 10 Meters (m/s)
WS10M_MAX MERRA-2 Wind Speed at 10 Meters Maximum (m/s)
WS10M_MIN MERRA-2 Wind Speed at 10 Meters Minimum (m/s)
WS10M_RANGE MERRA-2 Wind Speed at 10 Meters Range (m/s)
WD10M MERRA-2 Wind Direction at 10 Meters (Degrees)
-END HEADER-

LAT	LON	YEAR	MO	DY	T2M	T2MDEW	TS	T2M_RANGE	T2M_MAX	T2M_MIN	QV2M	RH2M	PRECTOTCOI	PS	WS10M	WS10M_MA	WS10M_MI	WS10M_RAI	WD10M
32.75	-123.75	2006	9	2	17.28	15.57	18.65	0.45	17.47	17.02	10.89	89.56	0.25	101.44	7.73	8.61	7	1.61	331.97
32.75	-123.25	2006	9	2	17.05	15.55	18.24	0.49	17.24	16.75	10.89	90.69	0.3	101.4	8.02	8.98	7.2	1.78	331.47
32.75	-122.75	2006	9	2	16.9	15.57	18.04	0.51	17.1	16.59	10.91	91.68	0.25	101.36	8.4	9.38	7.5	1.88	330.67
32.75	-122.25	2006	9	2	16.78	15.52	17.8	0.49	16.97	16.48	10.88	92.12	0.14	101.31	8.76	9.74	7.82	1.91	329.69
32.75	-121.75	2006	9	2	16.66	15.41	17.5	0.44	16.82	16.37	10.8	92.19	0.03	101.25	9.08	10.02	8.1	1.91	328.38
32.75	-121.25	2006	9	2	16.55	15.39	17.32	0.41	16.69	16.28	10.8	92.72	0.04	101.19	9.34	10.22	8.27	1.95	326.25
32.75	-120.75	2006	9	2	16.66	15.38	17.41	0.42	16.82	16.4	10.8	91.97	0.04	101.13	9.61	10.49	8.38	2.11	323.15
32.75	-120.25	2006	9	2	16.79	15.36	17.43	0.35	16.93	16.59	10.8	91.12	0.02	101.06	9.55	10.4	8.22	2.18	318.59
32.75	-119.75	2006	9	2	17.13	15.49	17.77	0.32	17.3	16.98	10.9	89.85	0.01	100.99	8.98	10.14	7.53	2.61	313.62
32.75	-119.25	2006	9	2	17.73	15.88	18.51	0.43	17.96	17.54	11.19	88.75	0	100.92	7.93	9.63	6.33	3.3	308.16
32.75	-118.75	2006	9	2	18.42	16.66	19.49	0.68	18.76	18.08	11.75	89.38	0	100.88	6.6	8.49	4.86	3.63	300.94
32.75	-118.25	2006	9	2	19.46	17.45	20.28	1.24	20.11	18.87	12.38	88.18	0.03	100.78	4.79	7.39	2.68	4.71	298.41
32.75	-117.75	2006	9	2	21.3	17.24	21.92	3.86	23.44	19.59	12.36	79.66	0.23	100.47	3.3	6.21	1.35	4.86	289.31
32.75	-117.25	2006	9	2	24.85	14.58	25.77	9.14	29.85	20.73	10.86	59.16	0.52	98.32	2.23	4.61	0.61	4.01	270.43
32.75	-116.75	2006	9	2	29.26	10.74	30.8	14.76	37.2	22.45	8.7	34.91	0.73	95.01	1.81	3.2	0.44	2.76	233.55

The raw drought dataset sample has been shown in Figure 18, in which the data consists of County, dates, and severity of drought-affected areas percentage.

Figure 18

Raw Drought Dataset

county	dates	None	D0-D4	D1-D4	D2-D4	D3-D4	D4	DSCI
Mariposa County	1/10/00	28.6	71.4	0	0	0	0	71
Sonoma County	1/10/00	68.2	31.8	0	0	0	0	32
Shasta County	1/10/00	22.29	77.71	0	0	0	0	78
Del Norte County	1/10/00	0	100	0	0	0	0	100
Inyo County	1/10/00	0	100	0	0	0	0	100
San Bernardino Cour	1/10/00	0	100	0	0	0	0	100
Solano County	1/10/00	69.16	30.84	0	0	0	0	31
Imperial County	1/10/00	0	100	0	0	0	0	100
El Dorado County	1/10/00	0	100	0	0	0	0	100
Los Angeles County	1/10/00	19.04	80.96	0	0	0	0	81
Mono County	1/10/00	1.25	98.75	0	0	0	0	99
Orange County	1/10/00	0.3	99.7	0	0	0	0	100
Monterey County	1/10/00	99.03	0.97	0	0	0	0	1
Ventura County	1/10/00	96.01	3.99	0	0	0	0	4
Lassen County	1/10/00	1.99	98.01	0	0	0	0	98
Siskiyou County	1/10/00	100	0	0	0	0	0	0
San Benito County	1/10/00	24.8	75.2	0	0	0	0	75
Santa Barbara Count	1/10/00	0	100	0	0	0	0	100

Figure 19 shows the raw data of NDVI value which is extracted from the Earth Engine Data Catalog in which an API was used to extract the NDVI value and date that image was taken.

Figure 19

Raw NDVI Dataset

County	date	ndvi_value
Alameda County	1/5/00	0.098712023
Alameda County	1/21/00	-0.035884832
Alameda County	2/6/00	0.108218886
Alameda County	2/22/00	0.10464126
Alameda County	3/9/00	0.183525157
Alameda County	3/25/00	0.31148064
Alameda County	4/10/00	0.312395324
Alameda County	4/26/00	0.311211034
Alameda County	5/12/00	0.311181498

Data Pre-Processing

Data preprocessing is a crucial stage in the modeling process that involves transforming raw data into a more suitable format for modeling. The primary goal of data preprocessing is to enhance the quality and accuracy of the data by addressing inconsistencies, missing values, and noisy data to create a more accurate and consistent dataset for modeling. For this study, the raw datasets were analyzed to identify patterns and inconsistencies. A data exploration plan was then developed to extract meaningful insights and guide the application of data cleaning and transformation techniques to the available datasets.

Data Exploration Plan

An essential step in the data engineering process is conducting data analysis to obtain a comprehensive understanding of the data complexity, interpretability, and anomalies. Various time series analyses were performed on the dataset to extract meaningful insights. The outcomes

of these analyses and the evaluation of all datasets will be leveraged to proceed with the Data Cleaning and Data Transformation stages. Below are some of the analyses executed on the raw data:

Figure 20 highlights the presence of outliers in the climatic data, which necessitates appropriate treatment in the data cleaning phase. One approach to addressing this issue is by removing the outliers from the dataset, while another approach involves imputing the data with alternative values.

Figure 20

Box Plot Illustrating Outliers in a Climatic Dataset

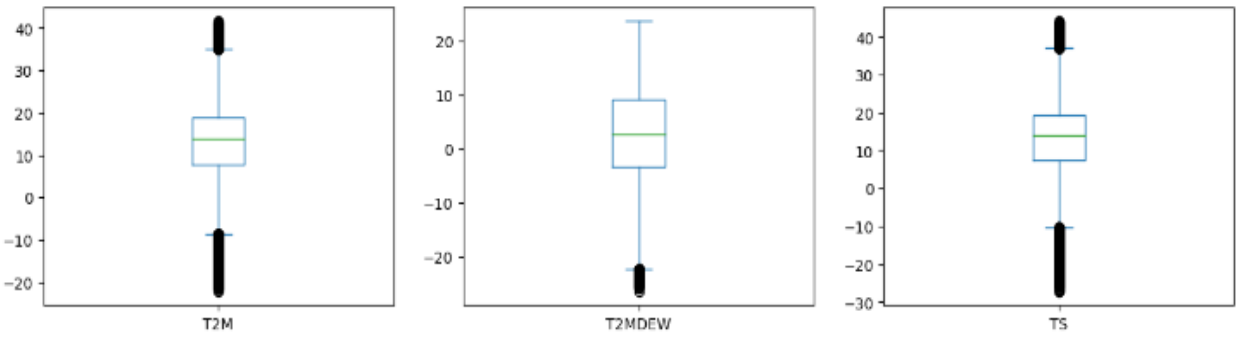


Figure 21 illustrates that the climatic dataset comprises three separate columns that correspond to the year, month and day of the data recorded for each county. When the date is not in the correct format, standard data analysis using Pandas or Matplotlib is not possible. Therefore, it is necessary to merge these three columns into a single column, which is an essential step in cleaning inconsistent data.

Figure 21

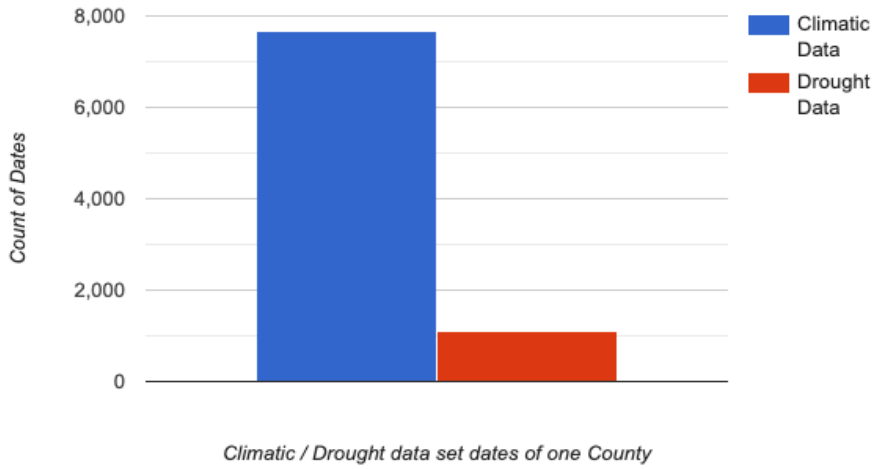
The Figure shows the Year Month Day in a Separate Column

LAT	LON	YEAR	MO	DY	T2M	T2MDEW	TS	...	QV2M	RH2M	PRECTOTCORR	PS	WS10M	WS10M_MAX	WS10
42.25	-116.25	2020	12	31	-2.67	-4.90	-4.07	...	3.14	86.09	0.41	83.57	3.52	6.54	
42.25	-115.75	2020	12	31	-2.23	-4.64	-3.33	...	3.19	84.99	0.38	84.09	3.90	7.04	
42.25	-115.25	2020	12	31	-2.18	-4.50	-2.85	...	3.20	85.39	0.42	84.29	4.05	6.65	
42.25	-114.75	2020	12	31	-2.37	-4.50	-2.74	...	3.19	86.11	0.46	84.40	3.93	6.35	
42.25	-114.25	2020	12	31	-2.77	-4.75	-3.04	...	3.14	86.58	0.48	84.35	3.59	6.34	

Based on Figure 22, the climatic data set encompasses time series data on a daily basis, whereas the drought dataset embodies weekly time series data that portrays the percentage of areas affected by drought. To attain temporal synchronization between the weekly drought data and the daily climatic data, a technique known as imputation must be executed to substitute missing values. Concretely, the weekly drought data values for each date within the week should be imputed to derive the daily percentage of the area affected by drought. Subsequently, merging the datasets with the climatic dataset based on the date and county variables is required.

Figure 22

Bar Chart showing the Count of Date Records



The heatmap was generated using a correlation coefficient computation technique between each pair of variables in the dataset. The resulting correlation values were then represented as a color-coded matrix. The correlation coefficient values can range between -1 and 1, where values closer to -1 indicate a strong negative correlation, values closer to 1 indicate a strong positive correlation, and values closer to 0 indicate no correlation. From Figure 23, the heatmap reveals strong correlations between temperature and humidity variables. On the other hand, precipitation exhibits almost zero correlation with all other variables, indicating its independence from the remaining variables. This information is useful for data transformation during the data reduction step, particularly when using techniques such as PCA or Singular Value Decomposition (SVD).

Figure 23

Heatmap showing Correlations between Variables in Climatic Data

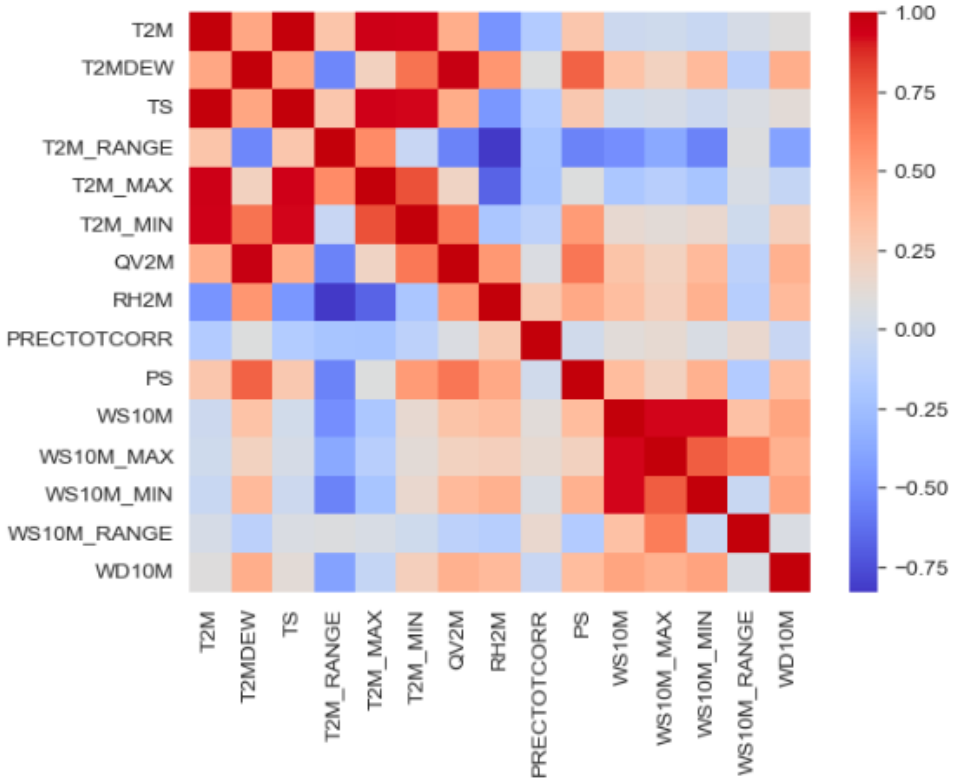


Figure 24 depicts the time-series trend of temperature and the monthly aggregated data are analyzed to comprehend the temperature fluctuations, as it is a crucial predictor feature for drought area percentage. Figure 24 reveals that California experiences high temperatures during 6 to 8 months, corresponding to June to August months, which corroborates with the domain knowledge. This strengthens the validity of the dataset, as it aligns with the real-world scenario in California.

Figure 24

Line Graph representing Temperature Values Yearly and Aggregated Monthly

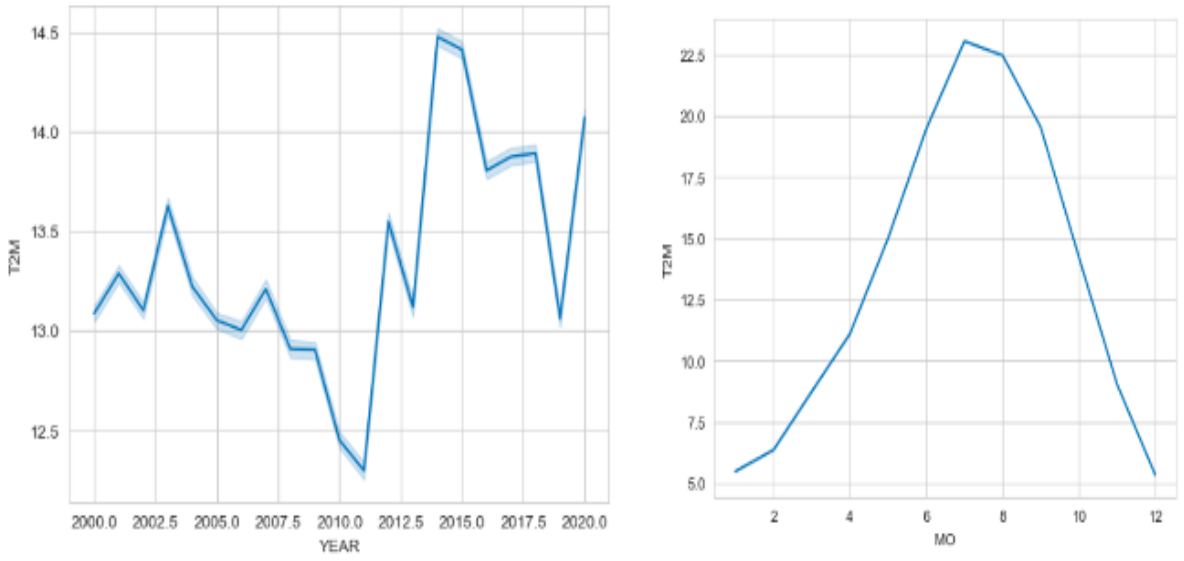


Figure 25 displays the average percentage of areas affected by drought from 2000 to 2020. The data reveals that all counties included in the dataset have experienced a minimum of 50 percent drought during this time period.

Figure 25

Bar Graph representing the Average Drought Percentage Area by County

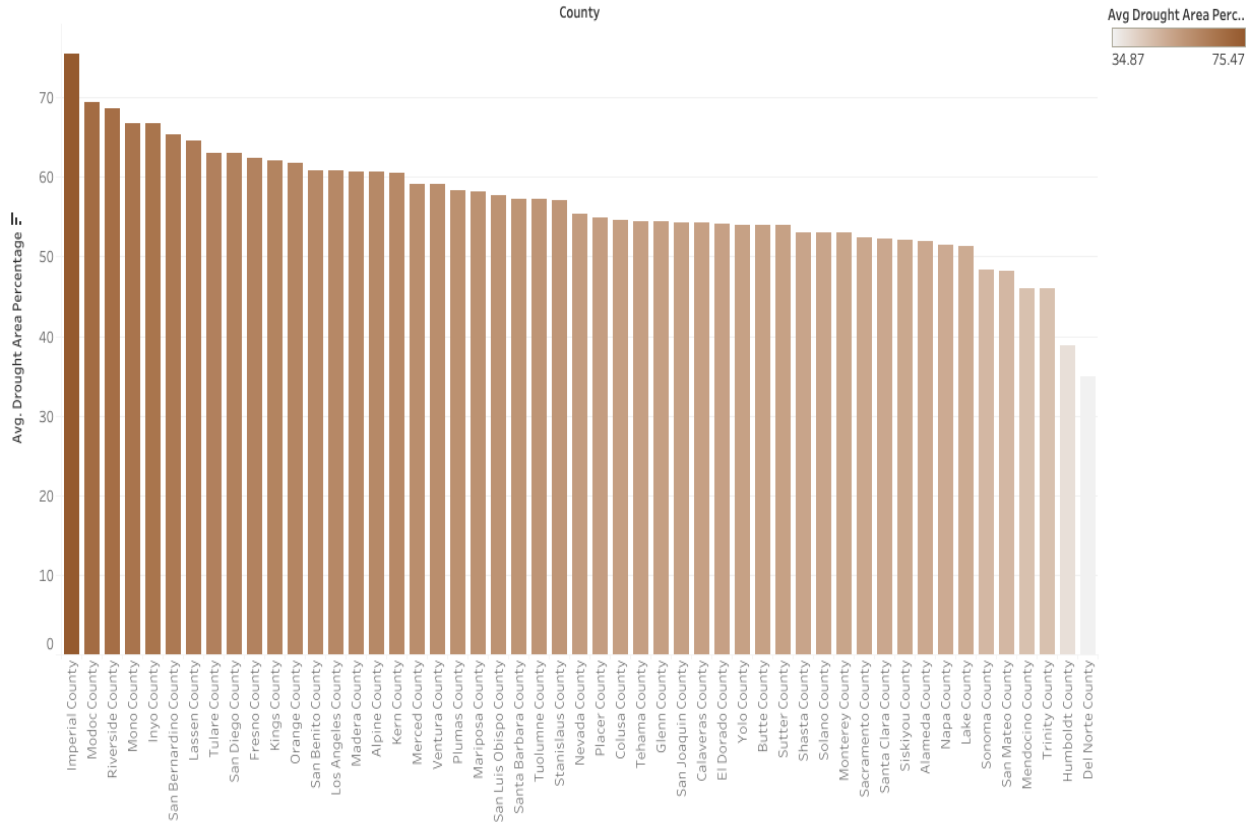
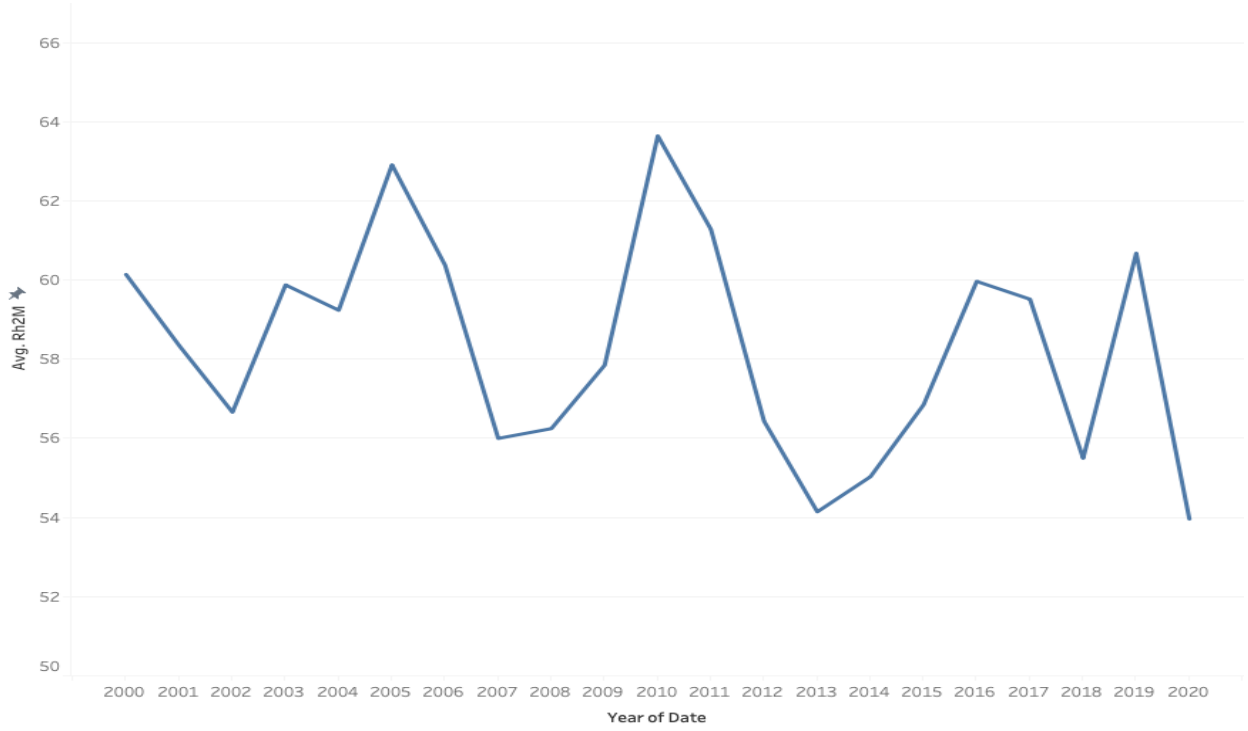


Figure 26 shows that from 2000 to 2020, California experienced lower relative humidity levels. The data was used to calculate the average relative humidity level, and it was found that there were fluctuations over this time frame. From 2011 to 2014, there were consistently low levels of relative humidity, which can contribute to an increase in the percentage of areas experiencing drought conditions across the state. This is because lower relative humidity levels lead to less rainfall, which can result in a lack of water bodies and ultimately lead to more areas experiencing drought conditions. The accompanying image further highlights the relationship between lower relative humidity and higher drought percentages.

Figure 26

Line Graph showing the Trends for Relative Humidity in the Climatic Dataset



Data Cleaning: Handling of Incomplete & Missing Data

The Drought Area percentage dataset provides the data of drought percentages for 51 counties in California on a weekly basis as shown in Figure 27. As the climate dataset values are all on a daily basis, the missing values for each county in the remaining week of the 21 years need to be filled.

Figure 27

Drought Data Available on the Weekly Scale

	Week	None	D0-D4	D1-D4	D2-D4	D3-D4	D4	DSCI	county
29513	2016-12-27	0.0	100.0	100.0	91.31	0.00	0.0	291	Alpine County
29514	2016-12-20	0.0	100.0	100.0	91.31	1.03	0.0	292	Alpine County

To address the issue of missing values, all the dates ranging from Jan 1, 2000, to Dec 31, 2020, were generated and used to fill in the drought areas percentage based on the weekly values and sample code is shown in Appendix A. This approach entailed assigning the same drought area percentage value for the entire week's dates, based on the weekly value. This process was applied to all the counties throughout the years as shown in Figure 28.

Figure 28

Drought Data After Using the Imputation Technique for Daily Drought Data

	county	dates	None	D0-D4	D1-D4	D2-D4	D3-D4	D4	DSCI
316134	Alpine County	2016-12-20	0.0	100.0	100.0	91.31	1.03	0.0	292.0
316186	Alpine County	2016-12-21	0.0	100.0	100.0	91.31	1.03	0.0	292.0
316229	Alpine County	2016-12-22	0.0	100.0	100.0	91.31	1.03	0.0	292.0
316255	Alpine County	2016-12-23	0.0	100.0	100.0	91.31	1.03	0.0	292.0
316333	Alpine County	2016-12-24	0.0	100.0	100.0	91.31	1.03	0.0	292.0
316363	Alpine County	2016-12-25	0.0	100.0	100.0	91.31	1.03	0.0	292.0
316414	Alpine County	2016-12-26	0.0	100.0	100.0	91.31	1.03	0.0	292.0
316481	Alpine County	2016-12-27	0.0	100.0	100.0	91.31	0.00	0.0	291.0
316521	Alpine County	2016-12-28	0.0	100.0	100.0	91.31	0.00	0.0	291.0
316568	Alpine County	2016-12-29	0.0	100.0	100.0	91.31	0.00	0.0	291.0
316648	Alpine County	2016-12-30	0.0	100.0	100.0	91.31	0.00	0.0	291.0

After populating daily basis values in the Drought dataset, missing values were identified as depicted in Figure 29. These missing values occurred due to the dataset's starting week being 4 Jan 2000, rendering all county values before that date as NULL. The data consisted of 153 null values per column.

Figure 29

Total Number of Null Values in Each Column After Imputation

```
Number of null values in each column:
county       0
dates        0
None       153
D0-D4      153
D1-D4      153
D2-D4      153
D3-D4      153
D4         153
DSCI       153
dtype: int64

Percentage of null values in each column:
county     0.000000
dates      0.000000
None       0.039108
D0-D4      0.039108
D1-D4      0.039108
D2-D4      0.039108
D3-D4      0.039108
D4         0.039108
DSCI       0.039108
dtype: float64
```

To address the issue of missing values, the dataset underwent a process of row elimination where the rows containing null values were removed, as depicted in Figure 30-31. These rows were deemed insignificant for the modeling task. Additionally, the climatic data available in the dataset initiated from Jan 4, 2000, coincides with the onset of drought data that is free of any missing values.

Figure 30

Counts of Total Records Before and After Dropping Null Values

```
Number of rows remaining before dropping null values: 391221
Number of rows remaining after dropping null values: 391068
```

Figure 31

Total Number of Null Values in Each Column Removing the Null Values

```
Percentage of null values in each column after dropping null rows:
county    0.0
dates     0.0
None      0.0
D0-D4     0.0
D1-D4     0.0
D2-D4     0.0
D3-D4     0.0
D4        0.0
DSCI      0.0
dtype: float64
```

The NDVI dataset has missing values for many dates between 2000 and 2020. These values are extracted from satellite images for 51 counties during the same time period. The missing values need to be handled before the data can be integrated. Figure 32 shows the missing dates in the dataset for Alameda County.

Figure 32

NDVI Data Having Missing Values for Some Dates

	County	date	ndvi_value
0	Alameda County	2000-01-01	-0.062434
1	Alameda County	2000-01-03	0.023525
2	Alameda County	2000-01-05	0.101889
3	Alameda County	2000-01-06	-0.136883
4	Alameda County	2000-01-08	-0.079211
5	Alameda County	2000-01-10	0.014741

The missing values in the dataset were filled using an interpolation technique. This technique involves estimating the missing values by interpolating between two available data points. After performing the interpolation, the dataset size increased from 220K records to 391K records, as all the data was now available for all the days in the dataset as shown in Figure 33.

Figure 33

NDVI Data After Interpolation Technique is Applied

	County	Date	ndvi_value
0	Alameda County	2000-01-01	-0.062434
1	Alameda County	2000-01-02	-0.019455
2	Alameda County	2000-01-03	0.023525
3	Alameda County	2000-01-04	0.062707
4	Alameda County	2000-01-05	0.101889
5	Alameda County	2000-01-06	-0.136883
6	Alameda County	2000-01-07	-0.108047
7	Alameda County	2000-01-08	-0.079211
8	Alameda County	2000-01-09	-0.032235
9	Alameda County	2000-01-10	0.014741

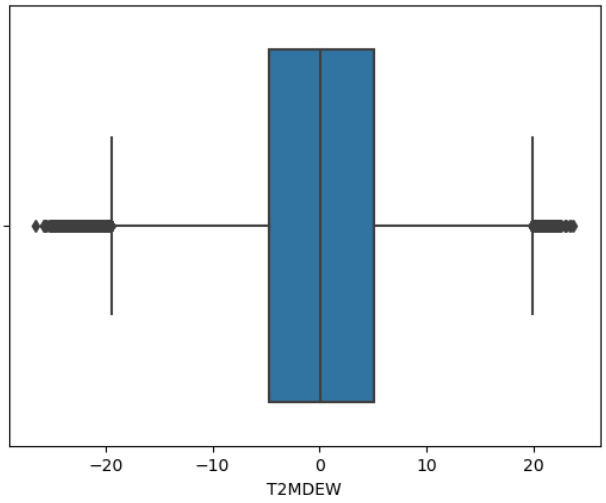
Data Cleaning: Handling of Noisy Data

The climatic dataset manifests anomalous values in certain columns, commonly known as outliers, which deviate significantly from the rest of the dataset. These outliers can lead to inaccurate analysis and modeling. Visual analysis methods such as scatter plots and box plots are frequently employed to detect outliers. In this study, outliers have been identified using box plots, as illustrated in Figure 34, and eliminated from the dataset that falls beyond the Max and Min range of the IQR region.

Anomalies lying beyond the interquartile range for the feature 'T2MDEW' in the climatic dataset were detected.

Figure 34

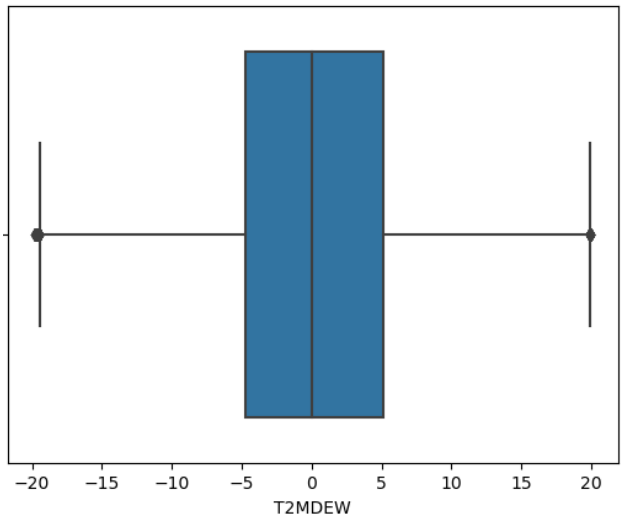
Box Plot to Identify the Outliers of the Data



The identification and removal of outliers are carried out by detecting the values that lie outside the Max and Min range of the IQR region leading to a more consistent dataset that can enhance the performance of modeling as shown in Figure 35.

Figure 35

Box Plot After Handling the Outliers for Climatic Data



Data Cleaning: Handling of Inconsistent Data

Inconsistencies in the data were addressed by employing several strategies, which involved transforming the dataset into the requisite format, generating novel features through the process of feature engineering, and converting the data types into a uniform format to facilitate more effective analysis as shown in Figure 36-37.

Figure 36

Before Combining the Columns Year, Month, and Day

LAT	LON	YEAR	MO	DY	T2M	T2MDEW	TS	T2M_RANGE	...	T2M_MIN	QV2M	RH2M	PRECTOTCORR	PS	WS10M	WS1C
32.75	-124.25	2000	1	1	13.61	9.43	14.13	0.79	...	13.25	7.20	75.81	0.02	102.37	7.93	
32.75	-123.75	2000	1	1	13.50	9.39	14.07	0.71	...	13.18	7.17	76.19	0.03	102.33	8.04	
32.75	-123.25	2000	1	1	13.43	9.38	14.12	0.72	...	13.11	7.17	76.41	0.03	102.30	8.19	
32.75	-122.75	2000	1	1	13.30	9.32	14.03	0.80	...	12.90	7.17	76.79	0.02	102.26	8.29	
32.75	-122.25	2000	1	1	13.14	9.26	13.84	0.84	...	12.70	7.15	77.24	0.01	102.22	8.36	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
42.25	-116.25	2020	12	31	-2.67	-4.90	-4.07	7.46	...	-5.76	3.14	86.09	0.41	83.57	3.52	
42.25	-115.75	2020	12	31	-2.23	-4.64	-3.33	6.80	...	-4.91	3.19	84.99	0.38	84.09	3.90	
42.25	-115.25	2020	12	31	-2.18	-4.50	-2.85	6.55	...	-4.59	3.20	85.39	0.42	84.29	4.05	
42.25	-114.75	2020	12	31	-2.37	-4.50	-2.74	6.60	...	-4.57	3.19	86.11	0.46	84.40	3.93	
42.25	-114.25	2020	12	31	-2.77	-4.75	-3.04	6.78	...	-4.88	3.14	86.58	0.48	84.35	3.59	

Figure 37

After Combining the Columns Year, Month, and Day into One Column Date

IDEW	TS	T2M_RANGE	T2M_MAX	T2M_MIN	QV2M	RH2M	PRECTOTCORR	PS	WS10M	WS10M_MAX	WS10M_MIN	WS10M_RANGE	WD10M	Date
9.43	14.13	0.79	14.03	13.25	7.20	75.81	0.02	102.37	7.93	9.20	6.65	2.55	332.65	01/01/00
9.39	14.07	0.71	13.89	13.18	7.17	76.19	0.03	102.33	8.04	9.36	6.70	2.66	329.88	01/01/00
9.38	14.12	0.72	13.83	13.11	7.17	76.41	0.03	102.30	8.19	9.56	6.95	2.61	327.08	01/01/00
9.32	14.03	0.80	13.71	12.90	7.17	76.79	0.02	102.26	8.29	9.81	7.11	2.71	324.39	01/01/00
9.26	13.84	0.84	13.55	12.70	7.15	77.24	0.01	102.22	8.36	10.11	7.22	2.89	321.39	01/01/00

The snippet highlights how inconsistent data are managed in a climate dataset. The county names were derived from the latitude and longitude coordinates using the geopy.geocoders library, a Python program for geocoding and reverse geocoding. Since the county names could not be established since the lat and lon coordinates were in marine regions

rather than land regions, the County values with None were eliminated from the dataset. The counties that did not belong to California were eliminated, including several Nevada counties, to make sure that the dataset solely contained California county names. The final dataset was reduced to just 51 different counties in California.

This process emphasizes how crucial it is to spot and deal with conflicting data in datasets. While tools like geopy.geocoders can make it easier to extract location-based data from datasets, it is important to take into account the constraints and any problems that can occur. The geopy package was used to extract county names from the latitude and longitude values in the dataset. However, some of these values are located in the ocean and not in California, resulting in the retrieval of null values as shown in Figure 38 to address this issue, the null values were removed from the dataset in the stage of handling inconsistent data, which is illustrated in Figure 39.

Figure 38

Before Removing NaN values in County Column

QV2M	RH2M	PRECTOTCORR	PS	WS10M	WS10M_MAX	WS10M_MIN	WS10M_RANGE	WD10M	Date	county
7.20	75.81	0.02	102.37	7.93	9.20	6.65	2.55	332.65	01/01/00	NaN
7.17	76.19	0.03	102.33	8.04	9.36	6.70	2.66	329.88	01/01/00	NaN
7.17	76.41	0.03	102.30	8.19	9.56	6.95	2.61	327.08	01/01/00	NaN
7.17	76.79	0.02	102.26	8.29	9.81	7.11	2.71	324.39	01/01/00	NaN
7.15	77.24	0.01	102.22	8.36	10.11	7.22	2.89	321.39	01/01/00	NaN

Figure 39

After Removing NaN values in County Column

TS	T2M_RANGE	T2M_MAX	T2M_MIN	QV2M	RH2M	PRECTOTCORR	PS	WS10M	WS10M_MAX	WS10M_MIN	WS10M_RANGE	WD10M	Date	county
.82	1.99	11.95	9.96	6.70	82.53	2.48	99.26	5.32	7.53	4.08	3.44	271.98	01/01/00	San Diego County
.79	3.75	10.66	6.92	6.09	86.21	2.92	95.74	5.37	7.21	3.96	3.25	263.85	01/01/00	San Diego County
.55	10.83	17.27	6.44	5.46	71.68	0.75	99.15	5.00	6.82	3.94	2.88	263.25	01/01/00	Imperial County
.52	12.00	17.97	5.96	5.19	67.88	0.10	100.10	2.94	4.76	1.59	3.17	265.37	01/01/00	Imperial County
.04	3.18	11.19	8.01	6.40	85.76	3.57	97.36	4.88	6.94	3.55	3.40	262.98	01/01/00	San Diego County

Data Transformation

Data transformation is an important step in this project. It involves converting raw data into a format that is suitable for analysis and modeling. Techniques such as normalization, feature engineering, and dimensionality reduction can be applied during data transformation to improve model performance.

Feature Engineering

Feature engineering is a type of data transformation that involves generating new features or altering existing features in a dataset to enhance the efficiency of machine learning models. In this project, A feature is being generated from the available dataset by utilizing the SPEI to demonstrate the drought index, which is calculated based on the precipitation and temperature over different time intervals. SPEI can be calculated using climatic water balance which will again be calculated by considering the difference between precipitation and Potential Evapotranspiration on a monthly time scale. To calculate the SPEI value, it is necessary to aggregate the values into monthly data for all the counties which will make the data size limited

to 12K records. Potential Evapotranspiration can be calculated based on average temperature values and humidity values.

The range of SPEI values can vary from negative to positive values, where negative values indicate drought conditions and positive values indicate wet conditions. The exact range of SPEI value depends on the specific calculation method used and the time scale being analyzed. However, in general, SPEI values can range from approximately -4 to +4, with extreme values outside of this range being rare as shown in Figure 40.

Figure 40

Calculated SPEI Values

county	Year	Month	SPEI_1
Alameda County	2000	1	-0.900588
Alameda County	2000	2	-1.189093
Alameda County	2000	3	0.450381
Alameda County	2000	4	-0.036886
Alameda County	2000	5	-0.945058
Alameda County	2000	6	-1.34502
Alameda County	2000	7	-1.823946
Alameda County	2000	8	-1.624129

Data Integration

Data Integration is one of the techniques used in Data Transformation in which data from multiple sources is combined into a single, consistent, and meaningful format. The process involves identifying the sources of data, resolving any inconsistencies and conflicts, and combining them into a single dataset that can be used for modeling.

The current project involves the utilization of three distinct datasets for the development of a machine-learning model. The initial dataset encompasses climatic data for California counties over a 21-year period, inclusive of latitude and longitude coordinates, and daily dates as shown in Figure 41. The second dataset consists of drought area percentages for California

counties over the same 21-year period, including daily dates. Lastly, the third dataset comprises NDVI data derived from satellite images, containing NDVI values and dates, as shown in Figure 42.

The integration of three datasets was accomplished by leveraging the shared attributes of Date and County names. This allowed for the consolidation of climatic variable data for each county on a specific month across all datasets, serving as a valuable opportunity to create a cohesive dataset with an emphasis on data integrity, consistency, and semantic relevance, as shown in Figure 43. Subsequently, PCA will be implemented to facilitate advanced analysis and modeling of the combined dataset. The utilization of PCA is among the intended next steps.

Figure 41

Climatic Dataset

LAT	LON	T2M	T2MDEW	TS	T2M_RANGE	T2M_MAX	T2M_MIN	QV2M	RH2M	PRECTOTCORR	PS	WS10M	WS10M_MAX	WS10M_MIN
32.75	-124.25	13.61	9.43	14.13	0.79	14.03	13.25	7.20	75.81	0.02	102.37	7.93	9.20	6.65
32.75	-123.75	13.50	9.39	14.07	0.71	13.89	13.18	7.17	76.19	0.03	102.33	8.04	9.36	6.70
32.75	-123.25	13.43	9.38	14.12	0.72	13.83	13.11	7.17	76.41	0.03	102.30	8.19	9.56	6.95
32.75	-122.75	13.30	9.32	14.03	0.80	13.71	12.90	7.17	76.79	0.02	102.26	8.29	9.81	7.11
32.75	-122.25	13.14	9.26	13.84	0.84	13.55	12.70	7.15	77.24	0.01	102.22	8.36	10.11	7.22
32.75	-121.75	13.00	9.21	13.68	0.82	13.40	12.57	7.12	77.68	0.00	102.19	8.42	10.41	7.31
32.75	-121.25	12.95	9.23	13.79	0.83	13.33	12.51	7.14	78.03	0.00	102.15	8.48	10.68	7.41
32.75	-120.75	12.92	9.28	13.95	0.83	13.33	12.50	7.17	78.46	0.01	102.11	8.60	11.04	7.57
32.75	-120.25	12.90	9.28	13.99	0.80	13.29	12.49	7.17	78.60	0.04	102.07	8.65	11.32	7.76
32.75	-119.75	12.83	9.24	13.85	0.76	13.20	12.44	7.15	78.68	0.08	102.03	8.53	11.43	7.66
32.75	-119.25	12.75	9.20	13.64	0.74	13.11	12.37	7.12	78.84	0.14	101.98	8.24	11.35	7.25
32.75	-118.75	12.74	9.25	13.67	0.77	13.09	12.32	7.14	79.22	0.19	101.95	7.88	11.22	6.82
32.75	-118.25	12.86	9.33	14.12	0.71	13.21	12.50	7.19	79.02	0.44	101.85	6.99	10.24	5.87
32.75	-117.75	12.55	9.09	13.87	1.08	13.10	12.02	7.09	79.34	1.44	101.53	6.00	8.76	4.87
32.75	-117.25	10.76	7.87	11.82	1.99	11.95	9.96	6.70	82.53	2.48	99.26	5.32	7.53	4.08
32.75	-116.75	8.18	5.92	8.79	3.75	10.66	6.92	6.09	86.21	2.92	95.74	5.37	7.21	3.96
32.75	-116.25	7.30	4.39	7.56	6.84	11.81	4.98	5.55	83.28	2.20	94.04	6.65	8.38	5.14
32.75	-115.75	10.52	4.86	10.55	10.83	17.27	6.44	5.46	71.68	0.75	99.15	5.00	6.82	3.94

Figure 42

Drought Area Percentage Dataset

county	dates	None	D0-D4	D1-D4	D2-D4	D3-D4	D4	DSCI
San Luis Obispo County	2000-01-04	28.60	71.40	0.0	0.0	0.0	0.0	71.0
Lake County	2000-01-04	68.20	31.80	0.0	0.0	0.0	0.0	32.0
Alpine County	2000-01-04	22.29	77.71	0.0	0.0	0.0	0.0	78.0
Sacramento County	2000-01-04	0.00	100.00	0.0	0.0	0.0	0.0	100.0
Tuolumne County	2000-01-04	0.00	100.00	0.0	0.0	0.0	0.0	100.0
Merced County	2000-01-04	0.00	100.00	0.0	0.0	0.0	0.0	100.0
Placer County	2000-01-04	69.16	30.84	0.0	0.0	0.0	0.0	31.0
Sutter County	2000-01-04	0.00	100.00	0.0	0.0	0.0	0.0	100.0
Calaveras County	2000-01-04	0.00	100.00	0.0	0.0	0.0	0.0	100.0
Mono County	2000-01-04	19.04	80.96	0.0	0.0	0.0	0.0	81.0

Figure 43

NDVI Values Dataset

County	date	ndvi_value
Alameda County	2000-01-01	-0.062434
Alameda County	2000-01-03	0.023525
Alameda County	2000-01-05	0.101889
Alameda County	2000-01-06	-0.136883
Alameda County	2000-01-08	-0.079211
Alameda County	2000-01-10	0.014741
Alameda County	2000-01-12	0.094747
Alameda County	2000-01-17	-0.363047
Alameda County	2000-01-19	-0.003276
Alameda County	2000-01-21	0.017480

Data Regularization

Overfitting is a significant issue that needs to be considered in any Machine Learning model. In overfitting, the ML model tries to fit each data point on the curve because of its learning from the training dataset, which is noisy. Since the model has very little flexibility, it fails to predict the new data points during prediction. Regularization techniques are employed to modify the linear regression model to reduce the adjusted loss function and avoid the problems of overfitting.

In this data regularization process, the first step followed is to split the data into training and testing sets. A linear regression model was then trained using the training set to predict the target variable. The model was evaluated by making predictions on the test set and calculating the mean absolute error and mean squared error.

To improve the performance of the model, ridge regression and lasso regression models were implemented. The Ridge() and Lasso() functions were used to train the models, and the mean squared error was calculated for each. The coefficients of the ridge and lasso models were also obtained using the coef_ attribute. Later L1 and L2 regression is applied from the results obtained from Linear regression.

From Figure 44, the model performs on the training set versus the test set, it is evident that the model is overfitting.

Figure 44

Linear Regression Score

```
Linear Regression-Training set score: 0.83
Linear Regression-Test set score: 0.59
```

From Figure 45, the obtained scores verify that ridge regression decreases the model's intricacy, resulting in a more generalized model that is less prone to overfitting.

Figure 45

Ridge Regression Score

```
Ridge Regression-Training set score:  0.77
Ridge Regression-Test set score: 0.71
```

From Figure 46, the results clearly indicate that the Lasso Regression model is experiencing underfitting.

Figure 46

Lasso Regression Score

```
Lasso Regression-Training set score: 0.30
Lasso Regression-Test set score: 0.21
```

The lasso model's effectiveness is limited as a vast majority of the coefficients have become precisely zero. Figure 47 shows the precise count of features utilized within the model

Figure 47

Count of Features Utilized After Data Regularization

```
Number of features: 9
```

The lasso regression model disregards some of the features in the training set, with only 9 of the 24 being utilized. Data sample after regularization is provided in Figure 48.

Figure 48

Dataset After Data Regularization

e	TS	QV2M	WS10M	WD10M	Month	Year	Lat	Lon	T2M	PRECTOTCORR	SPEI_1	Feature	Coefficient
0	9.116452	6.428226	3.130323	218.273548	1	2000	37.609029	-121.899142	9.723036	4.953214	-0.900588	county	0.081695
0	9.986034	6.871034	3.887759	181.979828	2	2000	37.609029	-121.899142	10.303103	6.548448	-1.189093	Date	0.157642

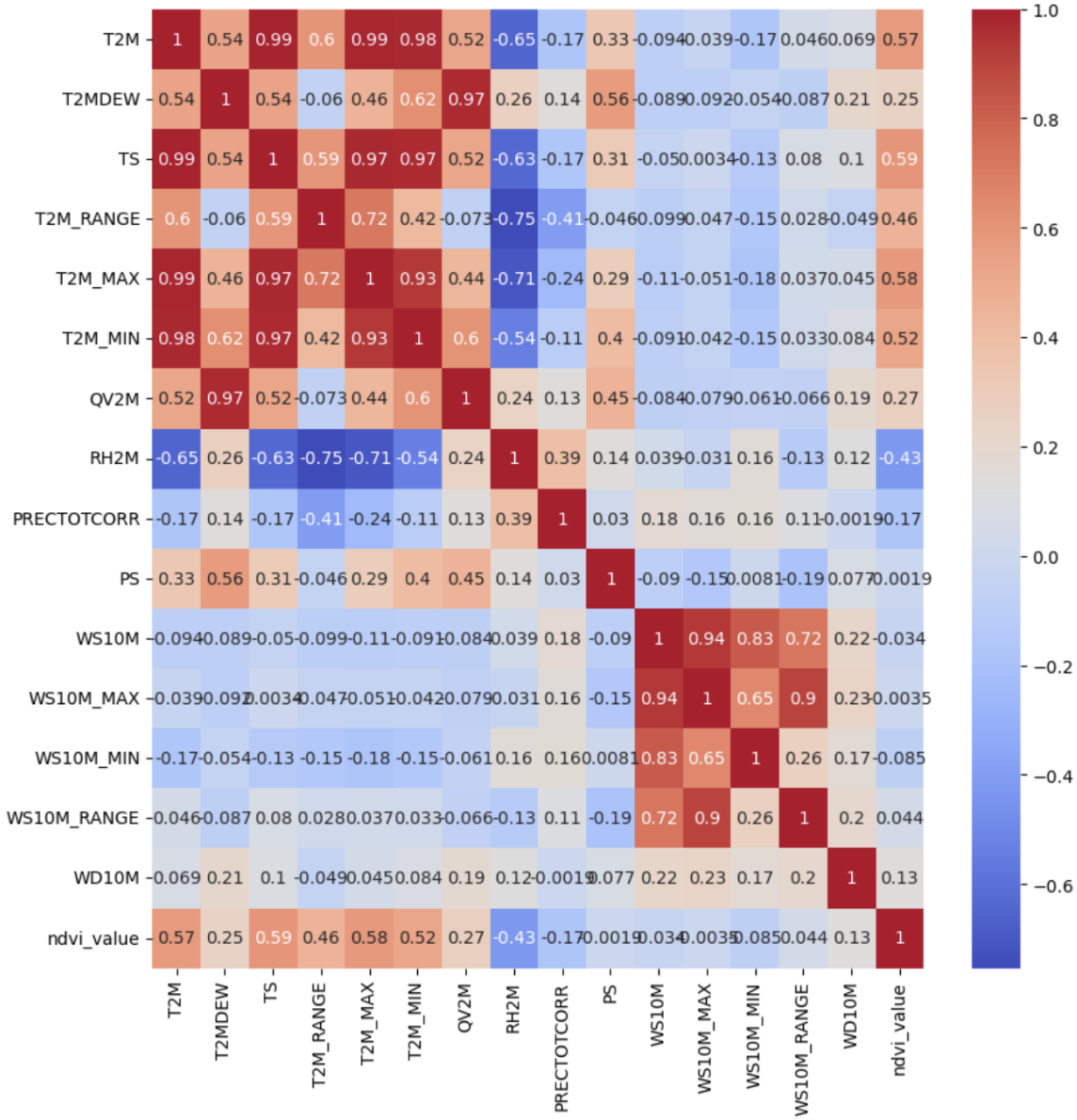
Based on the scores, it is evident that the ridge regression model (L2) is the best fit for the dataset.

Data Transformation: Data Reduction Using PCA

The PCA, or the principal component analysis technique, is often employed in data analysis to identify the most significant elements that contribute to changes in the data. PCA is used in this climate data project to minimize the number of features while making the data simpler to understand and display. Date, latitude, longitude, county, and various climate variables such as temperature, dew point, and precipitation all had been included in the dataset, which included a total of 28 columns. Plotted a heatmap to show the correlation between the numeric features is shown in Figure 49.

Figure 49

Heatmap to Demonstrate the Correlation between Variables After Integrating the Data



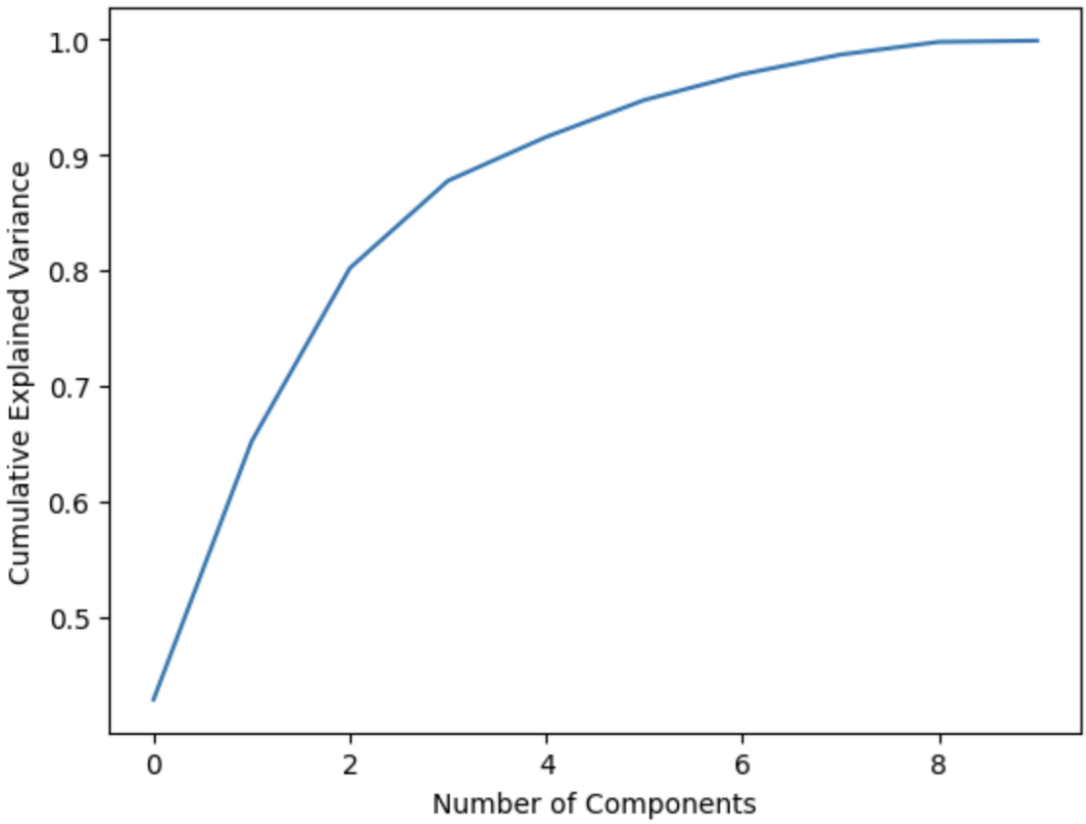
To ensure that all features after PCA receive the same importance, the non-numerical data was removed, then the numerical columns were standardized using scikit-learn's StandardScaler function. Post-data preprocessing, PCA with eight principle components was applied to scaled data. The reduced dimensionality space was created by converting the data with the retrieved

primary components. The generated dataset after PCA contained over two million rows and eight columns.

To determine the appropriate number of principal components to use for further analysis, a cumulative explained variance plot was generated as shown in Figure 50.

Figure 50

Cumulative Explained Variance Plot for PCA on Climatic Data



This is a line graph that displays the cumulative explained variance for each number of principal components, indicating that the first few principal components account for the majority of the variance in the data. More than 95% of the variance in the data is explained by 8 primary components.

After performing PCA on the dataset, the dimensionality of the data was reduced to 8 principal components. These 8 principal components captured the majority of the variance in the data while still maintaining the important information needed for predicting the target variable. Using these 8 principal components, identified the 8 original columns from the dataset that had the highest impact on the target variable.

The columns that were selected are shown in Figure 51. These columns were chosen because the mentioned eight variables had the highest impact on the target variable, and thus were the most important features for training the machine learning model.

Selecting specific columns reduced the dataset's complexity while maintaining high prediction accuracy. This method produced a more efficient and simplified mechanism for accurately anticipating new data.

Figure 51

Dataset after Applying PCA

Date	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	county	Drought_Target
2000-01-31	0.415447	0.553966	-0.071351	0.174321	0.135127	-0.061019	0.040840	-0.042783	Alameda County	27.7875
2000-01-31	0.927041	-0.175768	0.441159	-0.026158	0.153908	-0.184407	0.087667	0.004566	Alpine County	69.4275
2000-01-31	0.764870	0.178987	-0.033970	0.131633	0.232580	0.136561	-0.024457	0.082665	Butte County	71.2675
2000-01-31	0.605074	0.207979	-0.072633	0.169069	0.297221	0.078949	-0.004730	-0.044645	Calaveras County	75.0000
2000-01-31	0.527748	0.396629	-0.063918	0.165236	0.134251	-0.057327	-0.021543	-0.098130	Colusa County	52.7950

Data Preparation

Data preparation is a key component in any data modeling task, following data cleaning, regularization, and PCA. One important aspect of data preparation is dividing the data into training, validation, and testing sets, which is an essential step in modeling.

The training set is the largest subset of the data and is used to train the model and all the models are built using this training data. The validation set is a smaller subset of the data that is used to evaluate the performance of the model during the training process. The model is not trained on this data, but it is used to assess how well the model is generalizing to new data. The

validation set is used to tune the hyperparameters of the model to improve its performance. The test set is a separate subset of the data that is used to evaluate the final performance of the model. The model is not trained or tuned on this data, and it is only used to assess how well the model is able to generalize to new or unseen data.

Given that the dataset for this project is a time series data type spanning from 2000 to 2020, it was partitioned into training, validation, and test sets, with an approximate ratio of 70, 20, and 10 percent, respectively. The splitting was carried out based on the years, where the data from 2000 to 2018 was utilized for training, the data from 2019 was employed as validation data, and the data from 2020 was used as test data.

Splitting the data into the above mentioned ratio for training, validation, and testing datasets respectively provides a reliable and efficient method for maintaining the temporal dependencies between the time series data by evaluating the performance of machine learning models and improving their accuracy and effectiveness. The dataset's train, validation, and test sizes, which will be used to implement models for predicting the percentage of drought area, are illustrated in Figure 52.

Figure 52

Size of Train, Validation, and Test Datasets

```
Train set shape: (11628, 11)
Validation set shape: (612, 11)
Test set shape: (612, 11)
```

Data Statistics

Table 15 provides a detailed account of the different stages involved in processing three datasets that are climatic, drought, and NDVI.

For the climatic data, the raw dataset contains 3,068,000 rows and 20 columns, whereas the pre-processed dataset has 2,254,354 rows and 20 columns. The second dataset pertains to drought data, which is used as a dependent variable in the modeling process. The raw dataset contains 51,408 rows and 9 columns, while the pre-processed dataset has 39,117 rows and 9 columns. This is because the drought data was cleaned using the data imputation technique to match the timescale of climatic data. The third dataset pertains to NDVI data, which is used as an independent variable in the modeling process. The raw dataset contains 220,830 rows and 3 columns and then after performing an interpolation technique to fill the missing values, the total records for the NDVI dataset contain 391,170 rows and 3 columns.

The transformation stage involves data integration, feature selection, and feature extraction using PCA. Data integration involves combining the three different datasets into a single dataset. Feature selection involves aggregating and calculating the SPEI value for a monthly timescale which resulted in changing the size of the dataset from 2 Million records to 12K records. Feature extraction using PCA involves reducing the dimensionality of the dataset by identifying the 8 most important components.

The preparation stage involves dividing the dataset into training, validation, and testing subsets. The training set contains 9,180 rows and 9 columns, while the validation and testing set each have 1,836 rows and 9 columns. This splitting allows for the development and evaluation of models to predict the percentage of drought areas accurately. Figure 53 shows the statistics of each feature after the data preparation stage.

Table 15

Statistics of Each Dataset during Various Stages

Stage	Dataset	Methods	Statistics (Rows x Columns)
Raw	Climatic Data		3068000 x 20
	Drought Data		55845 x 9
	NDVI Data		220830 x 3
Pre-processing	Climatic Data		2254354 x 20
	Drought Data		390915 x 9
	NDVI Data		391170 x 3
Transformation	Merged Dataset	Data Integration Feature Selection Feature extraction (PCA)	12852 x 11
Preparation	Train Data		11628 x 11
	Validation Data		612 x 11
	Test Data		612 x 11

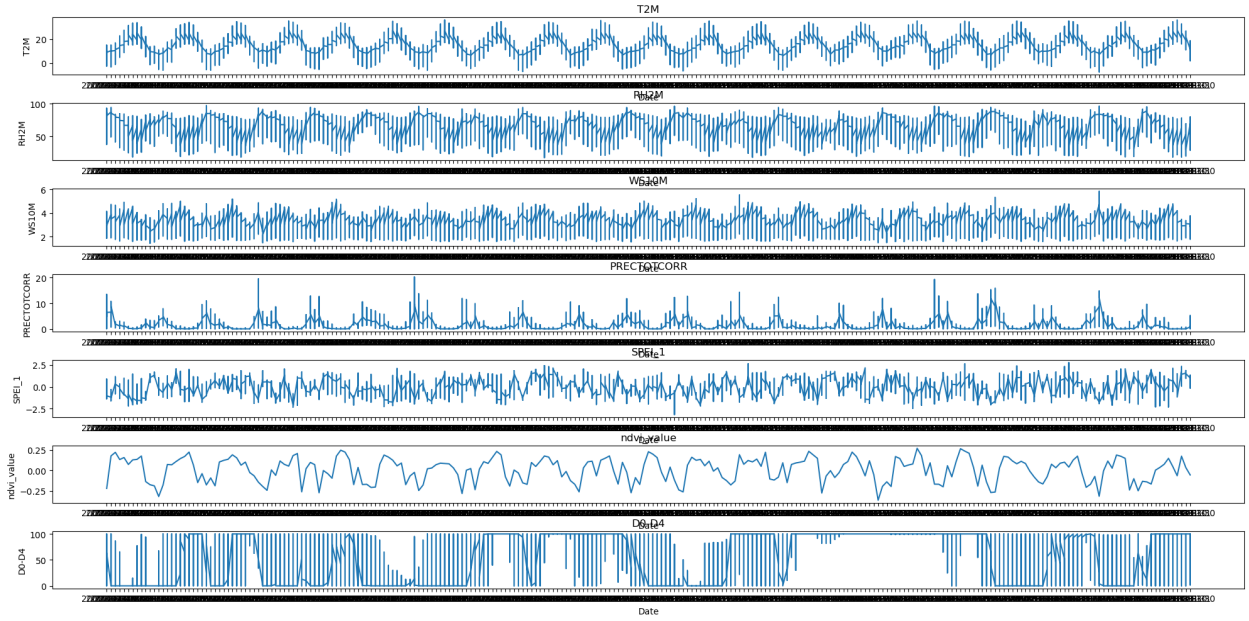
Figure 53

Statistics of each Feature after Data Preparation

	T2M	T2M_MAX	T2M_MIN	RH2M	WS10M	PRECTOTCORR	SPEI_1	ndvi_value	D0-D4
count	12852.000000	12852.000000	12852.000000	12852.000000	12852.000000	12852.000000	12852.000000	12852.000000	12852.000000
mean	14.121648	21.219258	8.220796	58.302470	2.991417	1.595577	-0.015828	0.006654	56.463748
std	7.525318	8.951401	6.230787	18.412878	0.581102	2.281025	0.969499	0.144943	46.168086
min	-7.367262	-2.346548	-13.125833	17.281462	1.445444	0.000000	-3.175957	-0.365494	0.000000
25%	8.672262	14.377270	4.021667	43.048667	2.578000	0.109651	-0.764087	-0.119338	0.000000
50%	13.558172	20.489785	8.135444	59.600000	2.932000	0.615000	0.012674	0.039242	86.183103
75%	20.079785	28.535833	12.915161	74.074839	3.370667	2.211290	0.747791	0.117690	100.000000
max	35.759839	44.602742	28.908387	97.127379	5.821429	20.265081	2.766556	0.269017	100.000000

Figure 54

Line chart for the Features throughout the Years'



Data Analytics Results

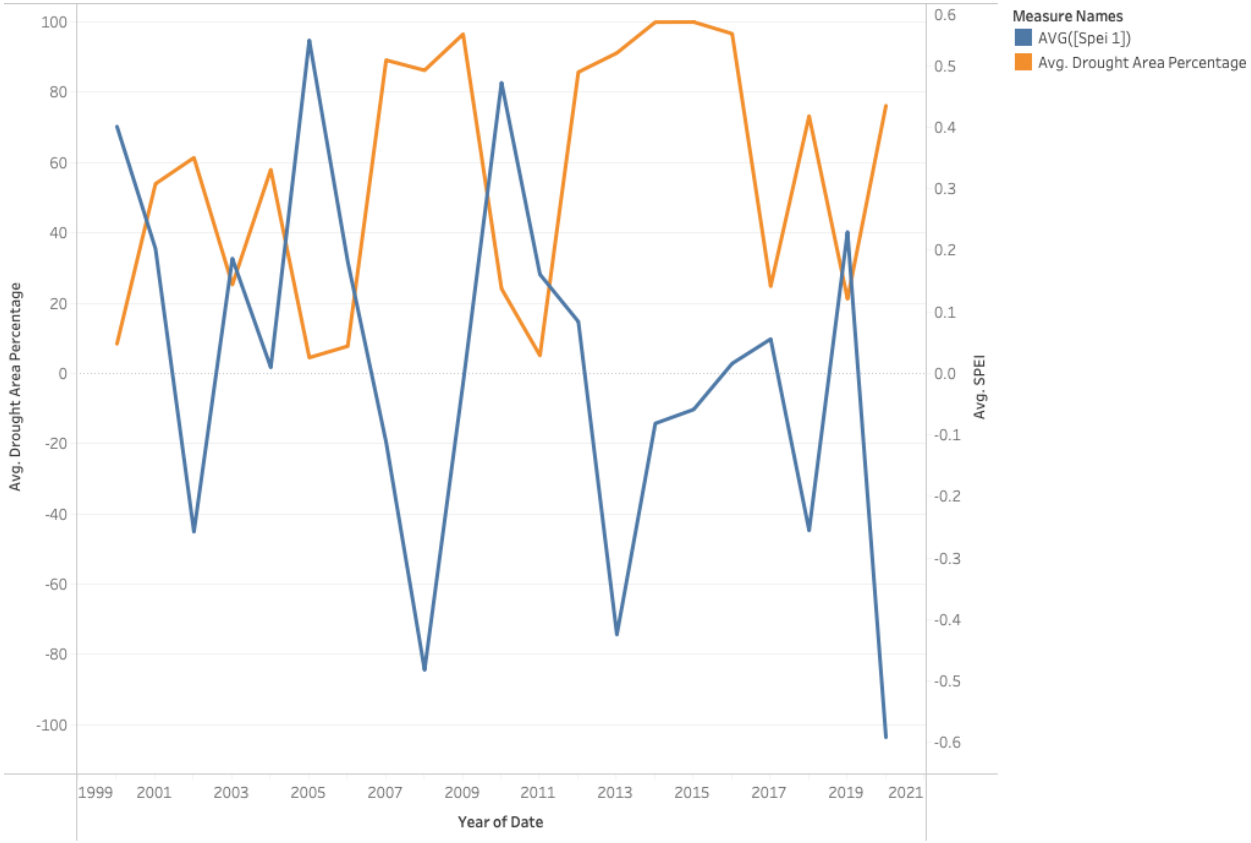
The process of exploring data to identify patterns and gain insights is known as exploratory data analysis. This involves visualizing the data, calculating statistics, and identifying trends. Important insights have been derived from the dataset through comparisons

among variables, correlations, and heatmaps that provide a more proactive way of visualizing the findings.

Figure 55 displays a line plot that shows the average drought area percentage and average SPEI value for all California counties from 2000 to 2020. The line plots have similar patterns with inverse proportionate, indicating that the calculated SPEI value is nearly equal to the drought area percentages obtained from the California drought monitor website. This suggests the reliability of the datasets used for the study.

Figure 55

Line Plot Comparison for Average Drought Percentage and Average SPEI Over the Years



The box plot in Figure 56 displays the annual temperature distribution for all California counties from 2000 to 2020. It is plotted to track the temperature trends over the years, which is one of the key factors in predicting the percentage of drought-affected areas.

Figure 56

Box plot for Temperature throughout the Years

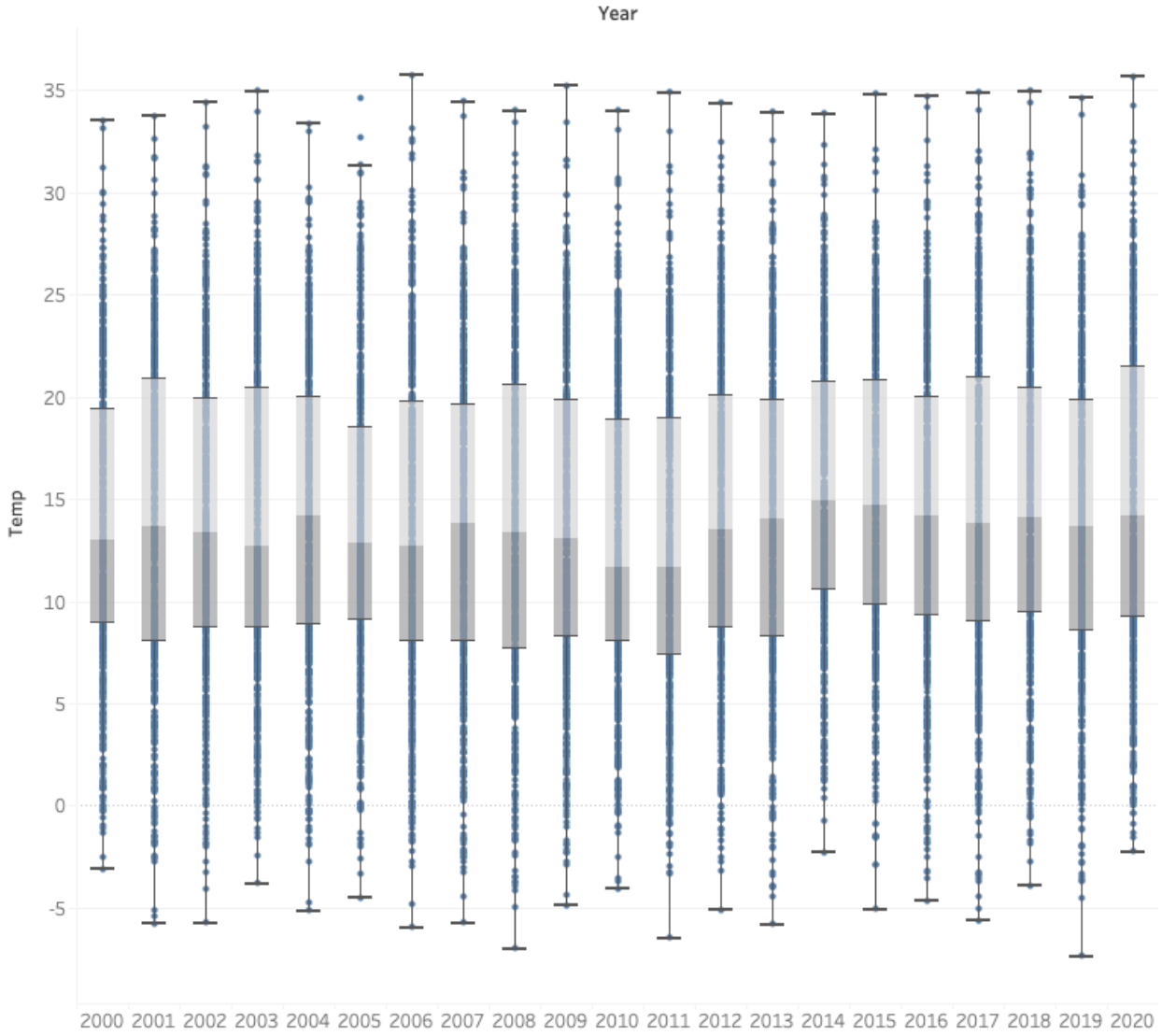
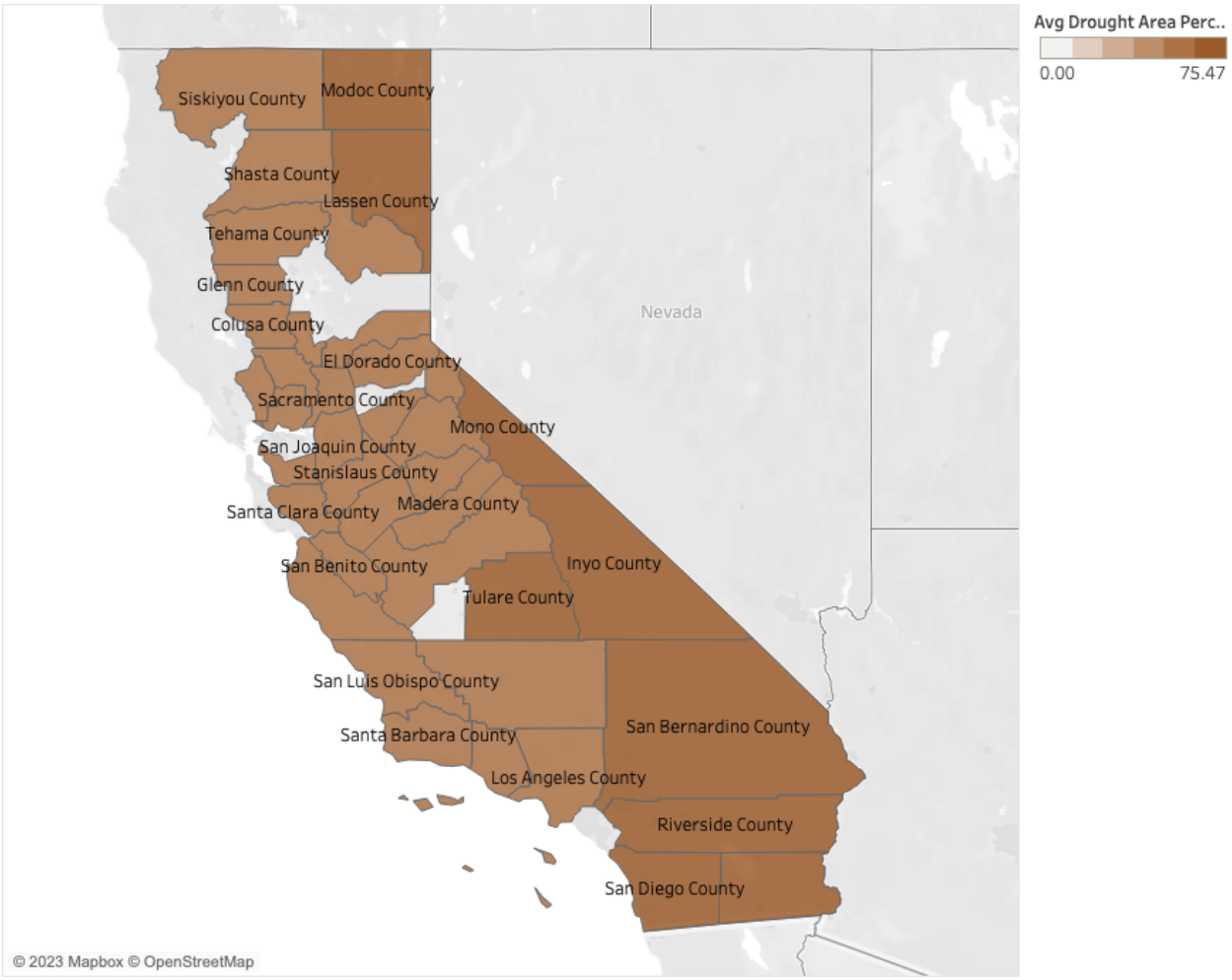


Figure 57 displays a heatmap presenting California counties having an average drought area percentage of more than 50 percent from 2000 to 2020. Counties shaded brown indicate an average drought area percentage of more than 50 percent, whereas non-shaded counties have an

average drought area percentage of less than 50%. This heatmap provides valuable insights into California's drought situation over the last 23 years, highlighting that a significant portion of the state is prone to drought.

Figure 57

Heatmap that shows Average Drought Area Percentage greater than 50 Percent

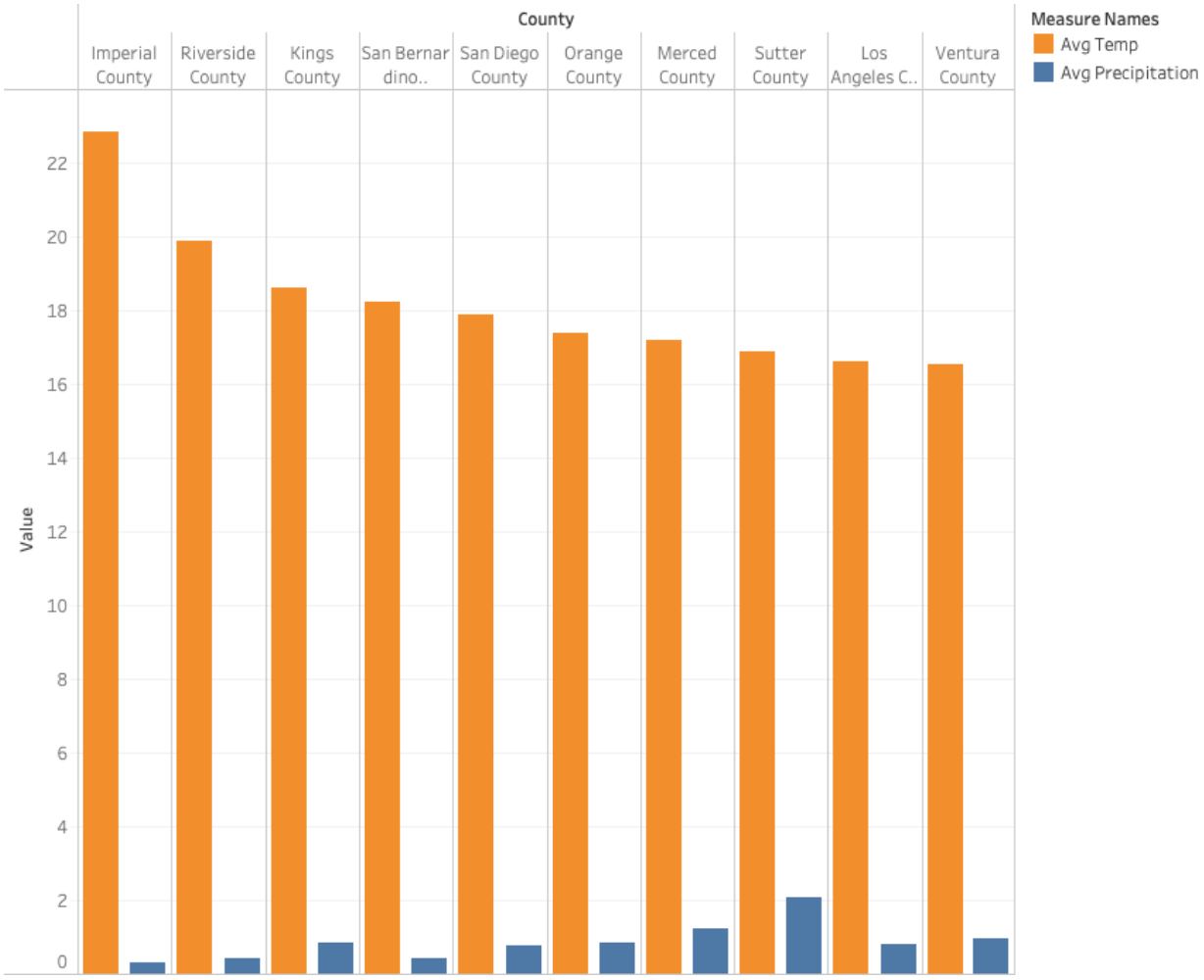


The bar chart in Figure 58 compares the average temperatures and precipitation of the top 10 counties with the highest recorded temperature over the years. This chart aims to investigate the correlation between temperature and precipitation in these counties. Typically, the values of these variables are inversely proportional to each other, which is demonstrated in the graph

below for each county. Furthermore, this relationship can be linked to the direct and inverse relationships between the drought area percentage and temperature and precipitation, respectively.

Figure 58

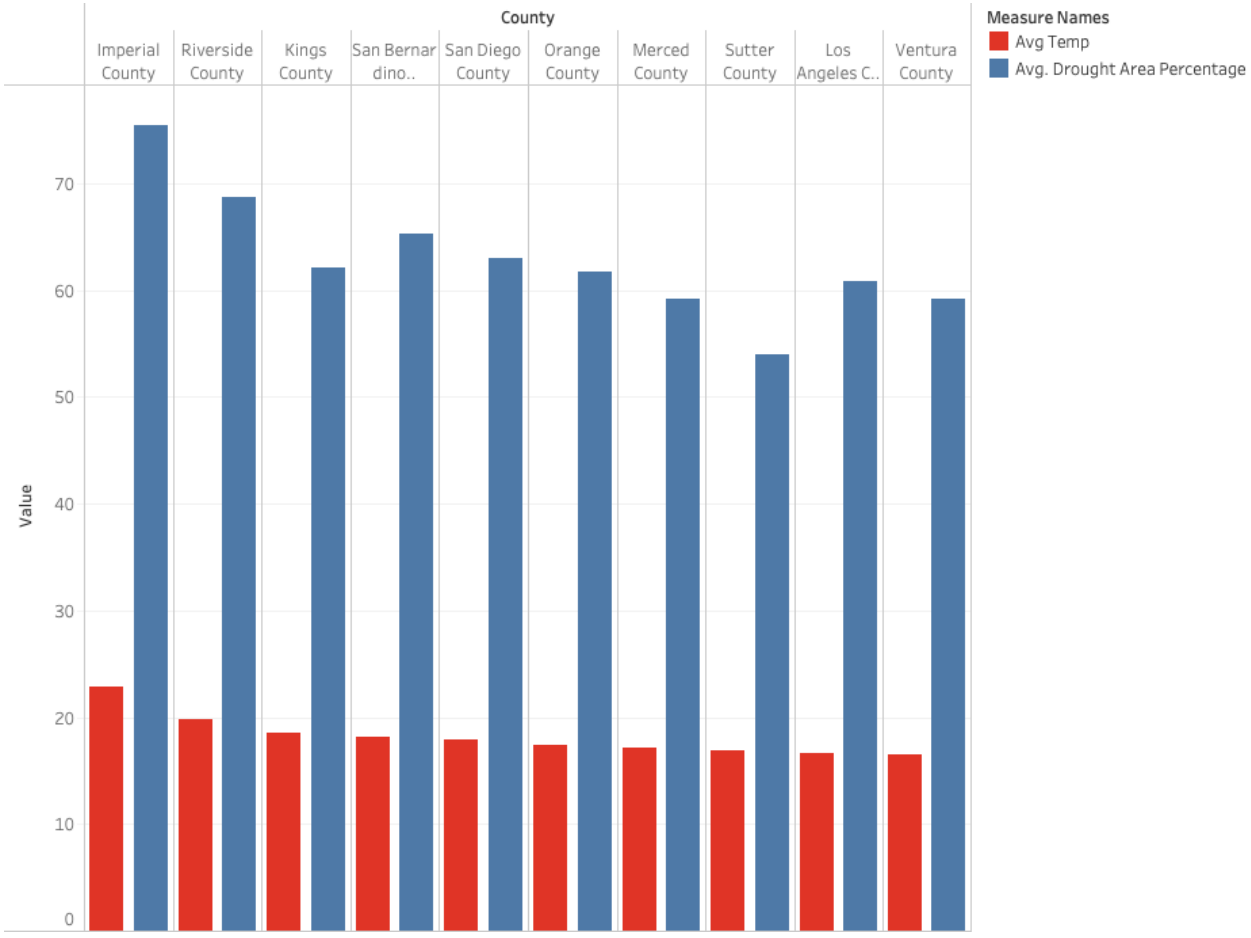
Bar Chart Plotted for the Top 10 Average Temperature and Precipitation



The relationship between average temperatures and drought area percentages among the top 10 counties is illustrated in Figure 59. This visualization provides crucial insights into how temperature affects the drought area percentage. The box plot clearly demonstrates that a rise in temperature has a significant impact on the drought percentage for almost all cases.

Figure 59

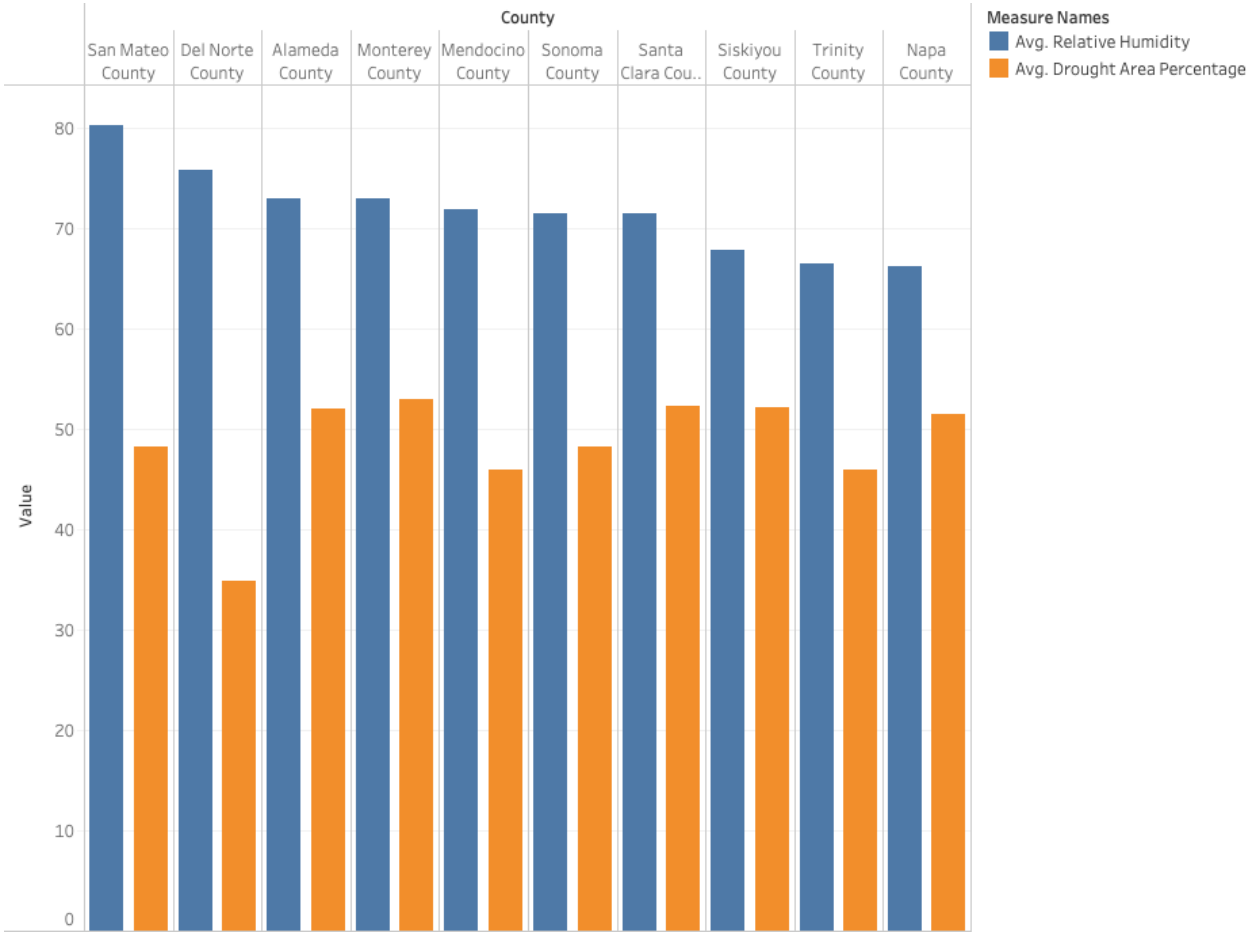
Bar Graph for the Top 10 Counties Average Temperatures and Average Drought Percentage



The bar graph in Figure 60 illustrates the relationship between the average relative humidity and drought area percentage for the top 10 counties over the years. The aim of the graph is to display the correlation between these two variables. Despite some inconsistencies, it is notable that the relative humidity has a significant impact on predicting the SPEI value and has an inverse relationship with the drought percentage. This finding is a crucial insight that highlights the link between relative humidity and drought percentage.

Figure 60

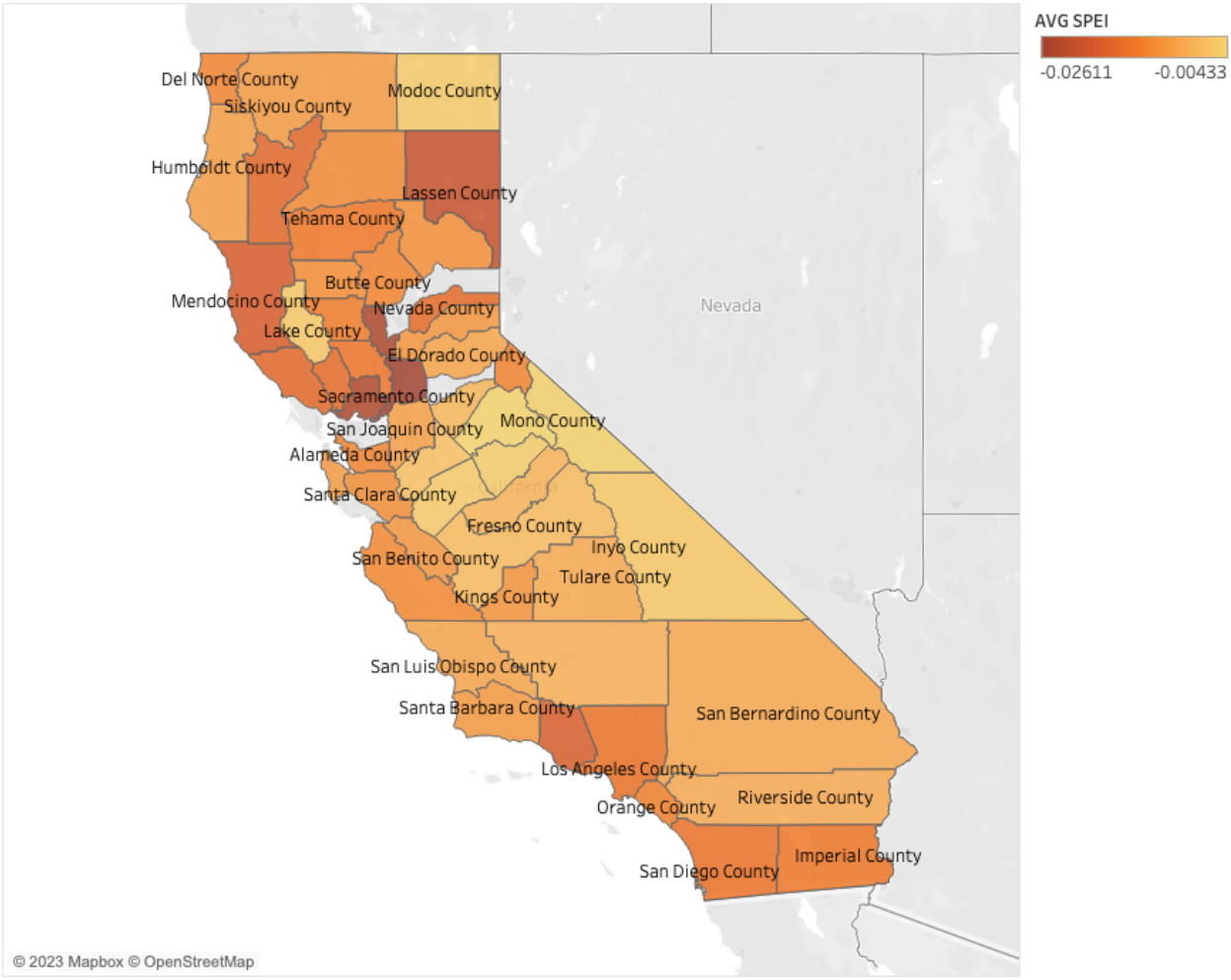
Bar Graph for Top 10 Average Relative Humidity and Average Drought Area Percentage



A heatmap in Figure 61 displays the average SPEI values of each county in California for the past 23 years. Negative values on the heatmap indicate that California has been prone to drought during this period. The middle region of California has been highly susceptible to droughts, while regions closer to Nevada have been less prone to droughts.

Figure 61

Heatmap Showing Average SPEI Values of Each County Throughout the Years



Model Development

Model Proposal

Predicting drought area percentage from time series data involves dealing with complex relationships between the climatic input variables and the target variable. To accurately model these relationships, it is important to choose a machine learning or deep learning model that can effectively capture non-linear dependencies and scale well to large datasets. In this research, Decision Tree Regressor, Random Forest, ANN, and LSTM were chosen as the models.

Decision Tree

Addressing complex relationships between climatic variables being used and the resultant target variable is necessary for accurately estimating the proportion of drought-affected areas using time series data. The choice of an appropriate machine learning or deep learning model becomes essential to accurately capture the non-linear relationships and manage massive datasets. For estimating the proportion of a region that is experiencing drought, many models like neural networks, decision trees, and support vector machines have been used. The Decision Tree model was especially chosen for this research because it can handle non-linear connections and scales effectively to fit large datasets. By analyzing feature importance and comprehending the decision-making process, the Decision Tree model offers interpretability. In order to identify drought patterns and their causes, it recursively separates data using a set of input variables. The model is appropriate for data with errors because of its resilience in managing missing values and outliers. Its potential for drought prediction is increased by its capability to handle categorical and numerical information. The goal of this research is to utilize the Decision Tree Regressor model to obtain an understanding of the variables affecting the likelihood of drought area percentage in California.

Random Forest

Random Forest creates an ensemble of decision trees, each using a random subset of input variables and observations, and aggregates their outputs to produce a final prediction. This technique reduces overfitting and improves generalization performance, which is important when dealing with complex relationships in climatic and SPEI data.

In the study by Dash et al. (2022), Random Forest was identified as a highly effective model for handling heterogeneous data. Unlike other models, Random Forest does not require faster GPU servers and is known for its computational efficiency. In time series modeling of climatic data, overfitting can be a major challenge due to the high dimensionality and nonlinearity of the data. According to Liu et al. (2016), they introduced a self-adjusting mechanism to the Random Forest model to prevent overfitting. This mechanism modifies the number of trees in the forest to achieve a balance between model complexity and prediction accuracy. They applied this approach to time series modeling of climatic data and compared it with conventional Random Forest models.

Artificial Neural Network

ANN has the ability to handle complex, non-linear correlations between input features and target variables. Furthermore, ANNs can generalize effectively to previously unseen data, which is critical in this research project because the model must properly predict dry areas for future periods. ANNs are also quite adaptable, as they can be utilized for regression as well as classification problems, making them a good fit for this research. Finally, ANNs have been implemented successfully to forecast drought conditions in previous research studies, indicating their potential utility in the present study.

In a study conducted by Bodri and Čermàk (2000), the authors used a collection of historical precipitation data from two meteorological stations in Moravia to create a neural network model to predict extreme precipitation events. Because radial basis function neural networks and backpropagation neural networks have proved effective in previous studies on precipitation forecasting, the authors trained those neural network models. The first model achieved an accuracy of 85% and was able to predict the precipitation events for up to ten days ahead. The neural network, according to the researchers, is an appropriate model for this research because of its non-linearity, because of which complex relationships between precipitation and variables like temperature and humidity can be captured. These conclusions are essential to this research since a nonlinear model, like a neural network, could be efficient at capturing the complex relationships among drought area percentages and other variables such as precipitation, temperature, and vegetation index.

Long Short Term Memory

The climatic data, drought data, and Normalized Difference Vegetation Index data are data sequences that extend over long timeframes used to predict future drought conditions. LSTM is effective in modeling sequences with long-term dependencies because it has memory cells that can selectively include or exclude information over time, which enables it to maintain critical information from earlier time steps and discard irrelevant information. This is particularly important when dealing with time-series data, where the previous states can have a significant impact on the current state.

In their study, Poornima and Pushpalatha (2019) compared the performance of LSTM and ARIMA for time series prediction. The authors found that LSTM outperformed ARIMA for longer time scales and when additional parameters with positive correlation were added to

Standard Precipitation Index and SPEI. However, they noted that LSTM was more resource-intensive than ARIMA. Despite this, the authors highlighted the strengths of LSTM, including its ability to overcome vanishing gradients, learn from past experiences, classify processes, and make predictions for time series.

The author's focus was to develop an LSTM model that achieved high accuracy and learning rate while minimizing the number of epochs required. Their use of univariate and multivariate approaches found that scaling up the number of layers improved model accuracy. In the multivariate approach with two layers of LSTM, they were able to reduce the number of epochs required by half compared to using only one layer of LSTM. The authors also addressed the constraint of increasing layers, as this increases time complexity since hidden layers undergo recursion. Despite the computational expense of LSTM, the authors observed very high accuracy and less Root Mean Square Error (RMSE) scores for different timescales and parameter combinations.

Model Supports

Hardware and Software Requirements

When working with machine learning or deep learning models, it is critical to consider the hardware requirements. A strong hardware setup can help accelerate the training and evaluation of models, improving the system's effectiveness and accuracy. For this research, a MacBook Air using the M1 chip was utilized. The M1 chip has an 8-core CPU for a balanced performance and power efficiency. The M1 processor also includes 8GB of RAM. This serves to minimize latency while improving overall model performance. Furthermore, the MacBook Air M1 includes an 8-core GPU designed to expedite machine learning operations such as ANN

model training and testing. The GPU is tailored for Apple's Core ML framework, allowing for rapid and efficient machine learning models execution on macOS.

Table 16 contains thorough information about the research's specific hardware needs.

Table 16

Hardware Specifications Used

Hardware	Configuration
CPU Model Name	Apple M1 chip
CPU Frequency	3.2 GHz
No.of CPU Cores	8
GPU	Integrated 8-core GPU
GPU Memory	8 GB
Available RAM Size	8 GB
Disk Space	256 GB

Software requirements also play a vital role in the development of the models. The operating system that was used in this research was Mac OS, which is extremely compatible with the applications needed for model development. The model was built with Jupyter Notebook, a free, open-source web tool that enables interactive coding in a variety of programming languages. Table 17 shows the software specifications that were used in this research.

Table 17

Software Specifications Used

Software	Configuration
Operating System	macOS Ventura
IDE	Jupyter Notebook

Tool and Libraries

In this research a variety of Python libraries such as Tensorflow, Scikit-learn, NumPy, Pandas, Keras, and Matplotlib were utilized to develop the models and maximize their performances. Detailed usage of each library is shown in Table 18.

Table 18

Libraries and Methods Used

	Library	Method	Usage
Scikit-Learn	sklearn.model_selection	Train_test_split, GridSearchCV	Used for splitting and hyperparameter tuning
	sklearn.preprocessing	StandardScaler, MinMaxScaler, LabelEncoder	Used for feature scaling.
	sklearn.metrics	Mean_absolute_error, mean_square_error, r2_score	Used for performance evaluation
	sklearn.tree	DecisionTreeRegressor	Decision tree-based regression modeling.
	sklearn.ensemble	RandomForestRegressor	Hyperparamaters tuning of Random Forest Regressor.

	Library	Method	Usage
Scikit-Learn	sklearn.impute	SimpleImputer	Generating the data using imputation techniques.
Numpy	numpy.ndarray	std,min,mean,abs,sqrt,sum,max	Used for Numerical computations
	numpy.random	randint	
Seaborn		histplot, boxplot, displot, lineplot, heatmap	Used for data visualizations
Matplotlib	pyplot		Used for data visualizations
Pandas	pandas.DataFrame	head,corr,info,describe,shape,to_datetime,drop,isnull,merge_asof,groupby,Grouper	Used for data manipulations
Keras	keras.optimizers	SGD, Adam	Used for building ANN and LSTM model and implementing hyper parameter tuning
	keras.layers	LSTM,Dropout, Dense	
	keras.models	Sequential	

Model Architecture and Data flow

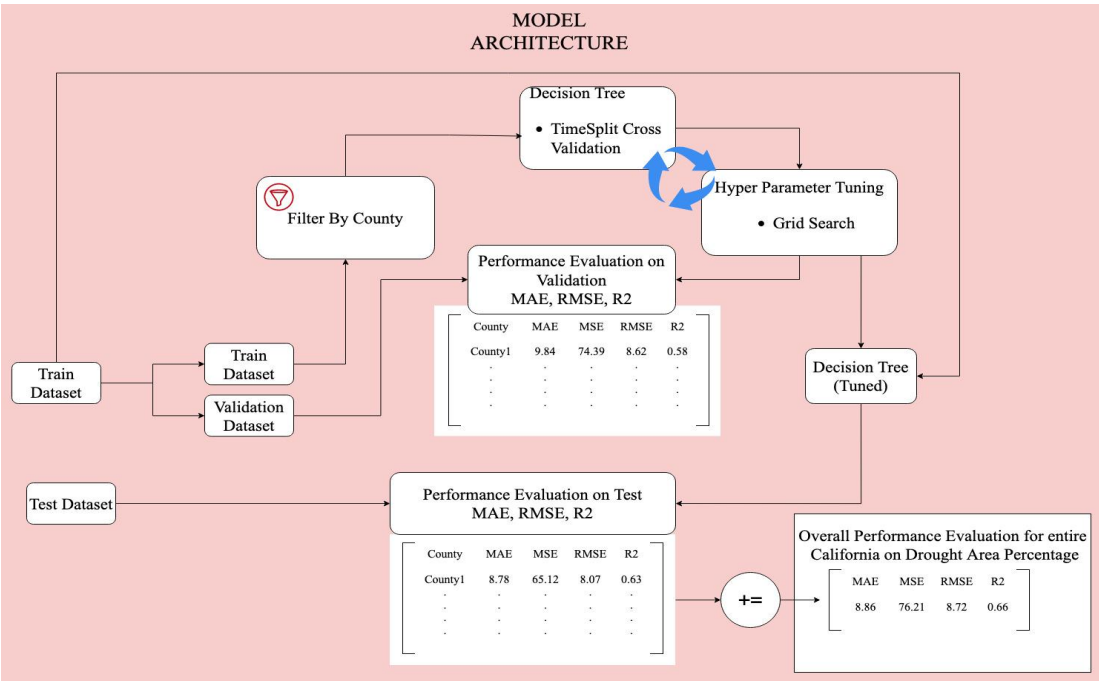
Decision Tree

The model was trained, tested, and evaluated using a dataset of historical weather, NDVI, and drought data for various counties to predict the Area percentage of drought. The training set covered a period of 20 years, from January 2000 to December 2019, and the test set consists of

the last year of available data, from January 2020 to December 2020. For training, the model utilized a feature set consisting of Relative_Humidity, Min_Temperature, Max_Temperature, Wind_Speed, Temperature, Precipitation, SPEI_1, and NDVI. The training set consists of 12240 records. A TimeSeriesSplit cross-validation technique with 4 splits was employed to value the model's effect on the training data. Each split involved a training set with a maximum of 2448 records and a validation set with 612 records and performed hyperparameter tuning using Gridsearch. After cross-validation, the model was fit into the entire training dataset, to evaluate its performance on a test set of 612 records. The model is created to train on the data from each county separately in order to forecast drought and analyzes evaluation metrics such as MAE, RMSE, and R2 for each county and record them in a list. The individual county metrics are aggregated to determine the overall drought area percentage and metrics for the entire state of California. Figure 62 shows the architecture of the Decision tree model used in the research.

Figure 62

Architecture of Decision Tree Model Used in Research



The climatic, NDVI, and drought data were collected from official websites, followed by a rigorous data cleaning process to handle missing values, null entries, and inconsistencies. Following the integration of the three datasets, data transformation operations such scaling, regularization, and dimensionality reduction (PCA) were performed.

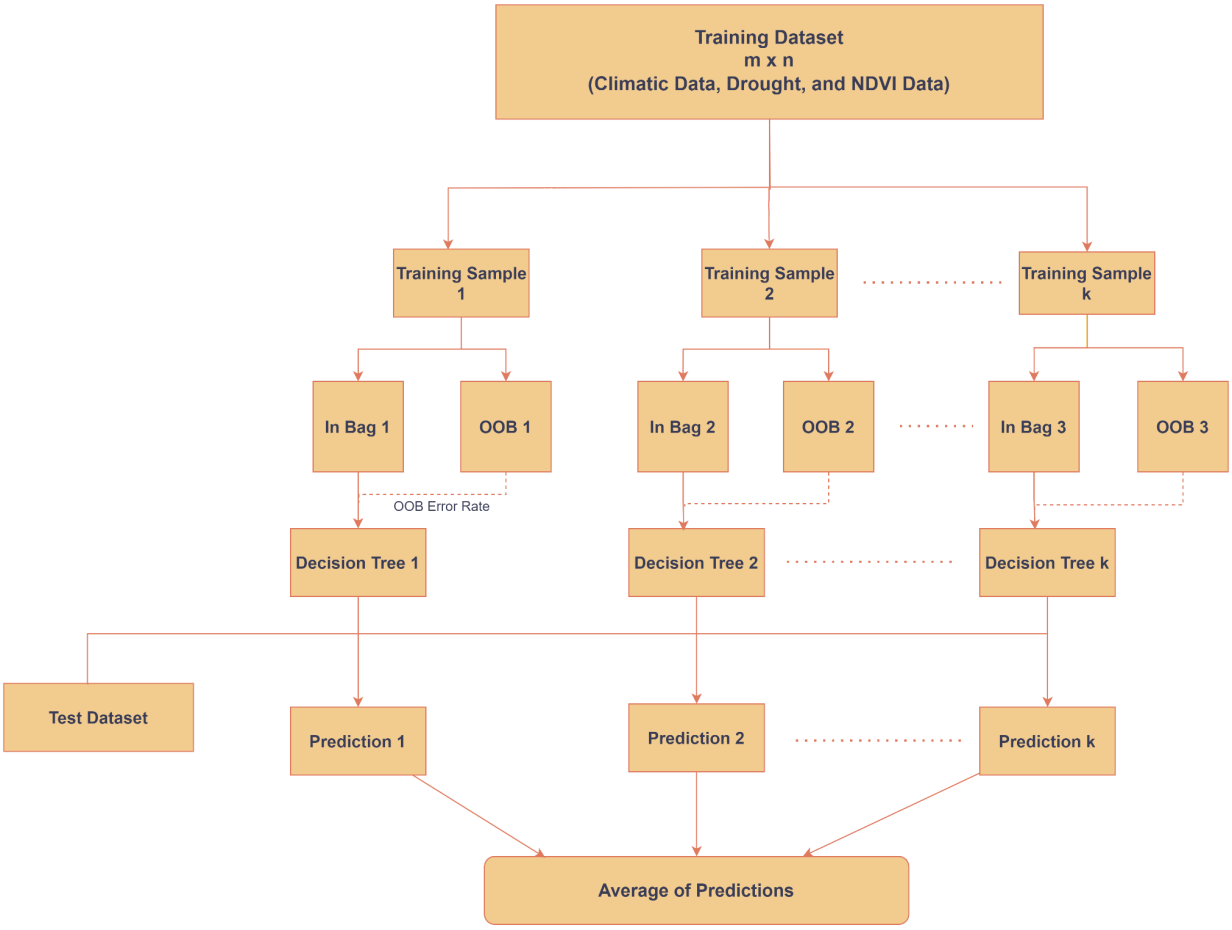
The data was divided into a training set covering 19 years. i.e. from 2000 to 2019 and a test set covering one year (2020) in order to prepare it for modeling. The model was tuned and optimized using a time-split cross-validation approach after being trained independently for each county. The model was then retrained using the whole training dataset, and its performance was assessed using the test dataset. In order to generate overall metrics, the evaluation metrics for each county were aggregated, giving information on the drought area percentage of California.

Random Forest

The training dataset which is being split into k number of random samples with a replacement where the size of the random sample should be less than the training data. This approach is known as bagging. OOB is a validation technique that uses the samples that were not included in the training of a particular decision tree to evaluate its performance. This ensures that each sample is used in the training or evaluation of at least one tree. OOB error rate in the decision tree provides a way to estimate the performance of the model during training and helps to avoid overfitting. And then the predicted data will be aggregated from all the predictions shown by the Decision Trees. Figure 63 shows the architecture of Random Forest model

Figure 63

Architecture of Random Forest Model Used in this Research



For each county, the data was trained and validated, and the evaluation metrics were computed at the county level. These metrics were stored in a list, and in the end, they were aggregated to obtain the overall metric value for the entire dataset, which can be considered for the entire state of California. This approach allowed for an evaluation of the model's performance at both the county and state levels, providing insight into its effectiveness in predicting drought in different regions.

After the validation of the baseline model, the training dataset from 2000 to 2018 was then again validated on the 2019 data using hyperparameter tuning. During this process,

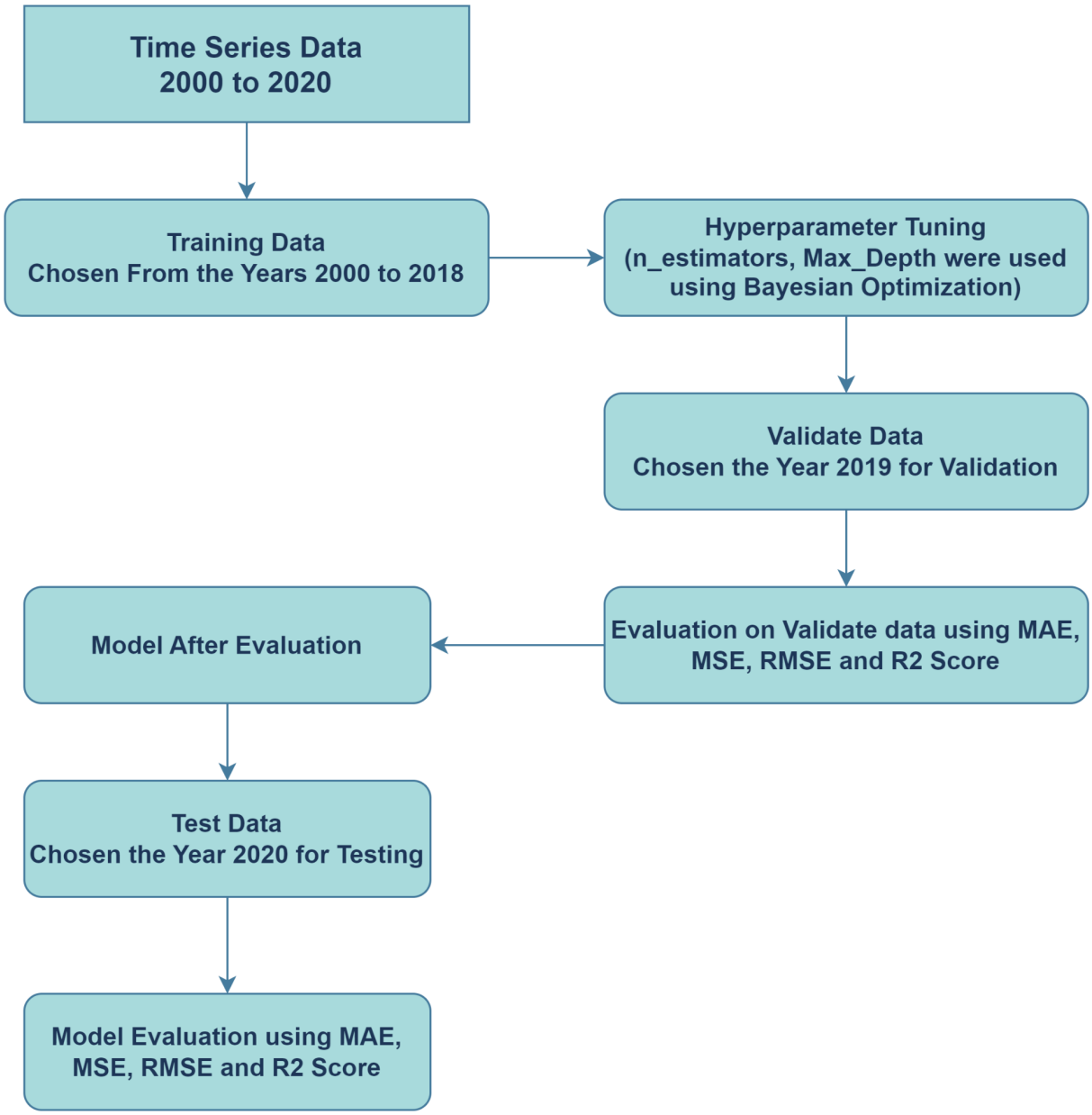
hyperparameter tuning using Bayesian Optimization was applied to the model in which a probabilistic model for the regression model's validation loss is constructed. The search space for this optimization technique is specified by passing the values of n_estimators ranging from 50 to 500, max_depth ranging from 2 to 20, and min_samples_split ranging from 2 to 20, from which the best possible combination of hyperparameters was chosen for the model. This optimization technique performs 32 iterations to determine the optimal hyperparameter combination. The resulting model showed better performance than earlier runs without any hyperparameters.

Using the hyperparameters identified during tuning, the model was trained on the entire dataset from 2000 to 2019 and then tested on the 2020 dataset. In comparison, another version of the model was also trained on the dataset from 2000 to 2019 but without passing any hyperparameters, and tested on the same 2020 dataset. The model trained with hyperparameters consistently outperformed the baseline model, as shown by the evaluation metrics.

Figure 64 shows the data flow of the model on how train, validate and test split has been done and the flow of evaluating and applying hyperparameter tuning on the data.

Figure 64

Data Flow of the Model

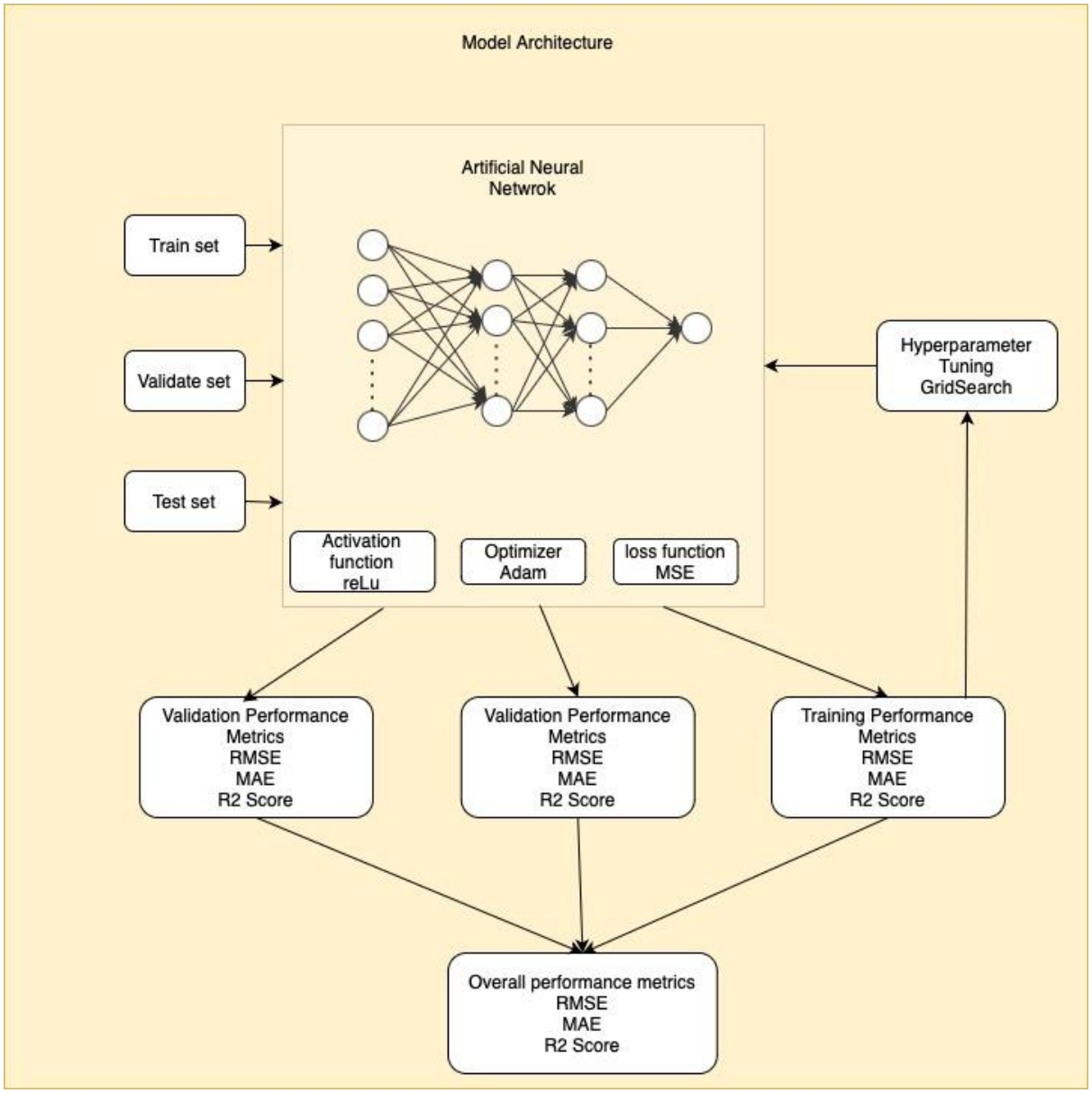


ANN

A total of four layers were employed to build the model: a single input layer, two hidden layers, plus one output layer. The two hidden layers have 100 and 50 nodes, respectively. Each node in the model was given random weights and biases at the start.

Figure 65

Model Architecture



The input variables were then propagated from the input layer to the output layer via hidden layers where the activation function mathematically transforms the input variables. The activation function that was used in this research was the Rectified Linear Unit (reLu) function, which is provided by (1). After obtaining the output data, the error is calculated using the loss

function MSE. During the backpropagation phase, a gradient g for each neuron is calculated. The weights and biases are then changed depending on that value to achieve model convergence. The optimizer function, such as Adam, is responsible for this.

$$z = max(0, v) \quad (1)$$

Following the development of the model, it was trained on the training dataset and tested on the validation set. Based on the results, hyperparameter tuning is done using grid search to select the optimal set of hyperparameters that can improve the performance of the model. Finally, predictions were made using the model on the testing set. The model was then evaluated using metrics such as MAE, MSE, RMSE, and R2 for each county and recorded in a list. Individual county indicators are aggregated to get the overall drought percentage and metrics for California.

LSTM

The transformed time series dataset is then divided based on the time period, the training data spans from the year 2000 to 2018, while the validation data is for the year 2019. The model is trained at the county level, where the data for each county train with the model separately. Models performance is then evaluated at the county level using various evaluation metrics. Finally, the county-level results are aggregated and rolled up to the state level for the state of California.

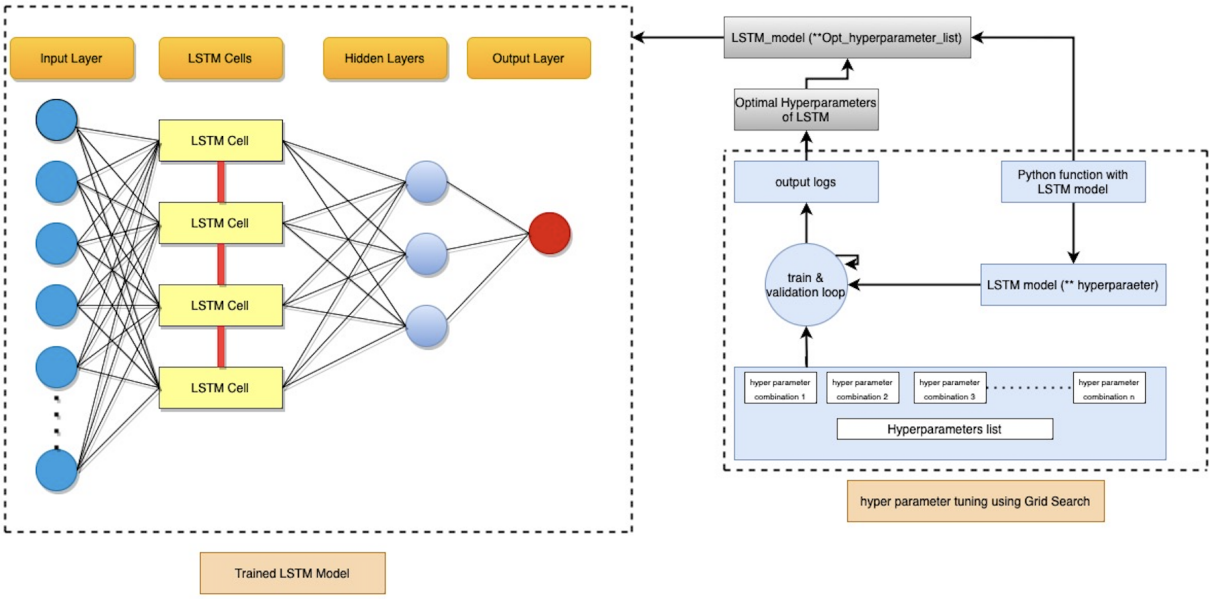
After the validation of the model using the 2019 dataset, it was found that the model's accuracy needed improvement. Therefore, to enhance the model's performance, hyperparameter tuning was carried out using Grid Search, and the model was retrained using the data from 2000 to 2018. The hyperparameters that were tuned were n_lstm_units, learning_rate, and dropout rate. By adjusting these hyperparameters, the model's performance was significantly improved, as the accuracy of the model increased. With this improvement, the model was further retrained

using the data from 2000 to 2019 and tested on the dataset from 2020 to evaluate its generalization capability.

The data flow within the different components of the LSTM model, the input data is first loaded into the LSTM layer, which is designed to handle sequential data. The LSTM layer performs a series of computations on the input data to generate a sequence of hidden states. These hidden states contain information about the input sequence, and are updated and passed along to the next time step in the sequence. This allows the LSTM layer to record long-term dependencies in the input data, making it apt for tasks such as time series forecasting. Figure 65 shows the architecture of the LSTM model

Figure 65

Architecture of the LSTM Model with Hyperparameter Tuning



The sequence of hidden states output by the LSTM layer is then passed on to the dense layer with the ReLu activation function applied. The dense layer is responsible for learning a non-linear relationship between the output of the LSTM layer and the target variable. This is

achieved by applying a series of matrix operations to the input data, followed by the application of the ReLu activation function. The ReLu activation function is commonly used in neural networks as it is simple, computationally efficient, and has been shown to be effective in preventing the vanishing gradient problem.

To prevent overfitting, a dropout layer is incorporated after the dense layer. The dropout layer arbitrarily drops some of the neurons in the course of the training process, thereby preventing the model from overfitting to the training dataset. The dropout rate is set to a value of self.dropout, which determines the fraction of neurons that are randomly dropped during each training iteration. This helps to ensure that the model is generalizing well to new data and not simply memorizing the training set.

Figure 66

LSTM Model with Various Layers in the Model

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 50)                11800

 dense (Dense)               (None, 32)                1632

 dropout (Dropout)           (None, 32)                0

 dense_1 (Dense)             (None, 1)                 33

=================================================================
Total params: 13,465
Trainable params: 13,465
Non-trainable params: 0
_________________________________________________________________
Summary of the ANN_Model :  None
```

Finally, the output from the dropout layer is passed through the final dense layer, which contains a single neuron that produces the predicted value. This output represents the model's prediction for the target variable at the next time step in the sequence.

Model Evaluation

In any machine learning or deep learning study, it is crucial to evaluate the performance of the models used to ensure that they provide reliable and accurate results. In this particular research, the target value is a continuous numerical value, for which the Random Forest regression model is one of the appropriate choices. To measure the accuracy of the predicted values compared to the actual values, various evaluation metrics are used. Some common evaluation metrics used in regression problems include MAE, MSE, RMSE, and R^2 score. These metrics provide different ways of quantifying the performance of the models, with some prioritizing accuracy over precision and vice versa. Careful consideration must be given to the choice of evaluation metric, as it can impact the interpretation and usefulness of the model's results.

Mean Absolute Error

The MAE value is commonly utilized in evaluating the effectiveness of a regression model. It determines the average absolute difference between the actual and predicted values of a dataset. The L1 norm, commonly known as the MAE, is calculated by averaging the absolute differences between predicted and actual outcomes. The advantage of using this metric is that it provides information on the extent of the error in the predictions, irrespective of the direction of the error. It is also less influenced by outlier values, unlike other metrics used for evaluation. The MAE formula is seen (2).

$$MAE = (1/n) * \sum_{i}^{n} (y_i - \widehat{y}_i) \quad (2)$$

where n is the number of samples, and y_i, $\widehat{y}_i$ are the actual and predicted values, respectively.

Mean Squared Error

Another frequently used metric for evaluation in problems related to regression is a MSE. The average of the squared differences between the actual and predicted values is how it is calculated. Lower numbers signify greater model performance, and the MSE value varies between 0 to infinity. By averaging the squared differences between the expected and actual values, the MSE formula is calculated. The formula is seen in (3).

$$MSE = (1/n) * \sum_{i}^{n} (y_i - \widehat{y}_i)^2 \quad (3)$$

where n is the number of samples, and y_i, $\widehat{y}_i$ are the actual and predicted values, respectively.

Root Mean Squared Error

Another often-used evaluation metric for regression models is RMSE. The root mean square error, or RMSE, calculates the average of the squared deviations between predicted and actual values. It is a more sensitive metric to outliers as it squares the errors. RMSE can help identify how close the predictions are to the actual values on average. Better model performance is associated with a lower RMSE. The formula is seen in (4).

$$\mathrm{R}MSE = \sqrt{(1/n) * \sum_{i}^{n} (y_i - \widehat{y}_i)^2} \quad (4)$$

where n is the number of samples, and y_i, $\widehat{y}_i$ are the actual and predicted values, respectively.

R-Squared Score

The R^2 value is a statistical metric used to measure how effectively the regression model fits the data, which is also known as the coefficient of determination. Its range is 0 to 1, with 1

signifying the most optimal model fit to the data. By comparing the variances of predicted and actual values, the R^2 value is determined. A negative R^2 number means the model performs worse than predicting the mean value of the target variable, whereas a value of 0 indicates poor model fit. This value is a useful tool for comparing the performance of different regression models. One drawback of the R^2 value is that it does not indicate whether the model is overfitting or underfitting the data. To calculate this value, the formula used is seen in (4).

$$R^2 = 1 - \left(\sum_i^n (y_i - \hat{y}_i)^2 \Big/ \sum_i^n (y_i - \bar{y})^2\right) \quad (5)$$

where n is the number of samples in the collection, y_i, $\hat{y}_i$ are the actual and predicted values respectively and $\bar{y}$ is the mean value of a dependent variable across all observations.

The research used the four evaluation metrics mentioned above to assess the model's performance. In conclusion, the evaluation metrics are crucial in determining the dependability and precision of regression models, making them essential for any machine learning or deep learning research that employs regression techniques.

Model Validation and Evaluation

In this study, four different models were employed to forecast the percentage of drought-affected areas. Prior to prediction, preprocessing, transformation, and splitting procedures were conducted. The models underwent initial validation using 2019 data, wherein each model was trained on data spanning from 2000 to 2018. Baseline models were trained on the training data and subsequently evaluated using the validation data. Following this, hyperparameter optimization was performed on each model using various techniques. Upon evaluating the validation data results, noticeable improvements in the performance of all models were observed. Utilizing the identified hyperparameters, the training data was extended to

include data from 2000 to 2019, while the test data consisted of 2020 data. In comparison, an alternative version of the model was trained on the dataset from 2000 to 2019 without applying any hyperparameters and then tested on the same 2020 dataset.

Decision Tree Base Model

Using the Decision Tree Base model, the prediction results yielded an MAE value of 10.21, an MSE value of 87.79, an RMSE value of 9.37, and an R2 score of 0.62. It is important to note that no hyperparameter tuning was performed on the data during the predictions.

Decision Tree Hypertuned Model

The Gridsearch Optimization technique was utilized to perform hyperparameter tuning and the model's performance was marginally improved by selecting the optimal hyperparameters of max_tree_depth as 13, min_samples_split as 4, and min_samples_leaf as 6, criterion as MSE, CCP as 0.07, as evidenced by the results. The predictions produced an MAE score of 8.86, an MSE score of 76.03, an RMSE score of 8.72, and an R^2 score of 0.66. Gridsearch Optimization took a considerable amount of time to generate the results because the search space was extensive due to the iterations in Gridsearch, which resulted in a longer execution time.

Random Forest Base Model

Using the Random Forest Base model, the prediction results yielded an MAE value of 7.61, an MSE value of 43.28, an RMSE value of 6.57, and an R2 score of 0.70. It is important to note that no hyperparameter tuning was performed on the data during the predictions.

Random Forest Hypertuned Model

The Bayesian Optimization technique was utilized to perform hyperparameter tuning and the model's performance was marginally improved by selecting the optimal hyperparameters of n_estimators as 423, max_depth as 18, and min_samples_split as 5, as evidenced by the results.

The predictions produced an MAE score of 6.81, an MSE score of 39.73, an RMSE score of 6.3, and an R^2 score of 0.74. Bayesian Optimization took a considerable amount of time to generate the results because the search space was extensive, and the number of iterations was set to 32, which resulted in a longer execution time.

ANN Baseline Model

The original ANN base model generated prediction results with an R2 score of 0.67, an MAE of 8.91, an MSE of 84.20, an RMSE of 9.17, and an RMSE of 84.20. It's crucial to point out that the dataset was never tuned for the hyperparameters prior to obtaining these results.

ANN Hypertuned Model

The grid search technique is utilized, in which a range of values for each hyperparameter is established, including the activation function, hidden layer count, and neurons count, and then the algorithm is trained on the model on each combination of hyperparameters in the grid. After doing the hyperparameter tuning, the algorithm provided the optimal combination of parameters as the two hidden layers, 100 neurons in the first hidden layer and 50 neurons in the second, the activation method being reLu, and the optimizer being the Adam function. Running the model with these hyperparameters yielded results of 7.43 for MAE, 52.67 for MSE, 7.25 for RMSE, and 0.70 for R2.

LSTM Base Model

A common practice in time series analysis is to ensure that the model is accurately predicting future values based on the patterns it has learned from past data. So the data of the years prior to 2020 is used for training. The results of the evaluation metrics for the model are as follows: an MAE value of 12.86, an MSE value of 168.52, an RMSE value of 12.98, and an R2 score of 0.54.

Post Parameter Tuned LSTM Model

The results of the baseline model were not satisfactory with very less accuracy rates. To boost the models performance Grid Search model is employed to perform hyperparameter tuning. Hyperparameter tuning involves searching for the best combination of hyperparameters that will result in the highest performance of a model. This procedure helps to optimize the model's effectiveness for a particular dataset.

The dictionary param_grid was used to specify the values for each hyperparameter. For the n_lstm_units hyperparameter 32, 64, and 128 values are used to determine the optimal number of LSTM units. The lr hyperparameter was tuned using values of 0.1, 0.01, and 0.001 to determine the optimal learning rate. And we tuned the dropout hyperparameter using values of 0.1, 0.2, and 0.5 to determine the optimal dropout rate. The grid search explored all possible combinations of these hyperparameter values to find the best combination that minimizes the MSE. Though the model took more time than the baseline model, it showed improved performance with the following hyperparameters n_lstm_units: 32, lr: 0.01, dropout: 0.2. The predictions produced an MAE score of 11.20, an MSE score of 126.31, an RMSE score of 11.23, and an R2 score of 0.56.

Comparison of Models

Comparing evaluation metrics of Decision Tree, Random Forest, ANN, and LSTM models, Table 19 demonstrates the impact of hyperparameter tuning on their performance. Notably, all models exhibited improved results after the optimization process, underscoring the significance of hyperparameter tuning for achieving better performance.

Table 19

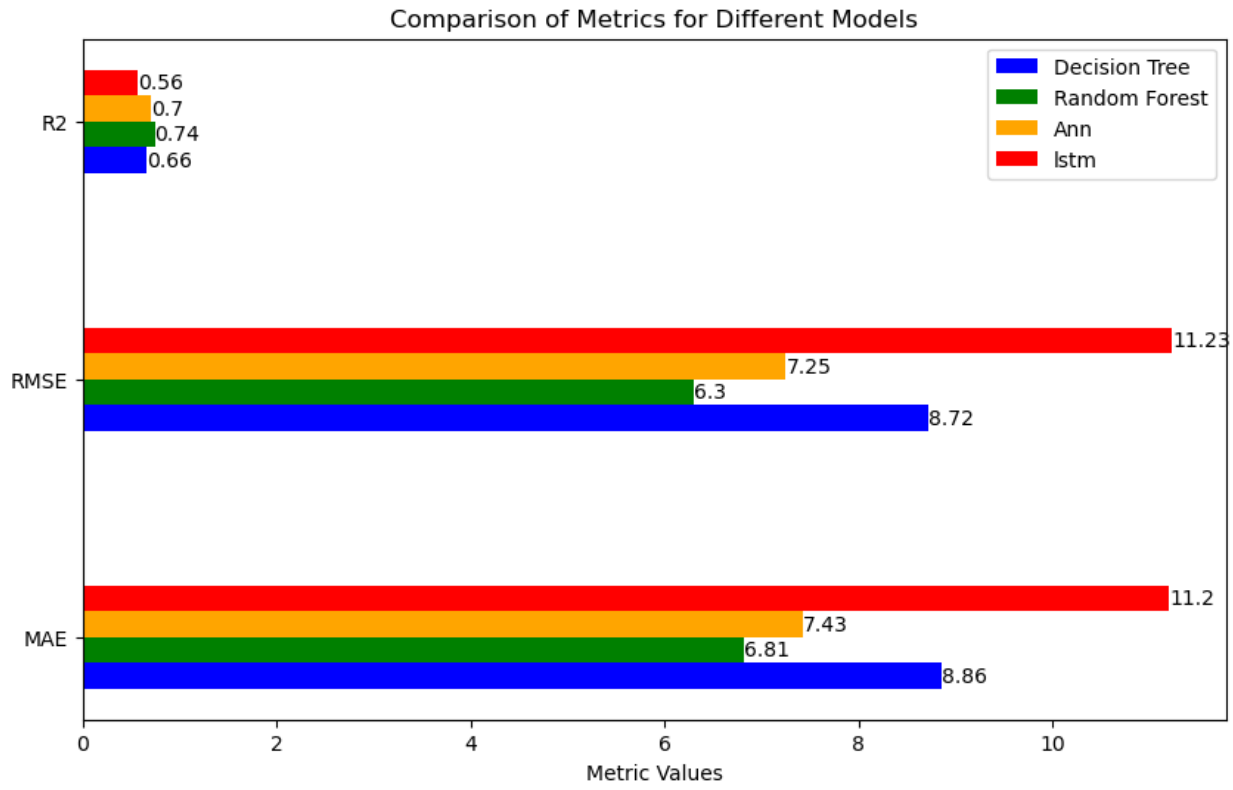
Comparison of Models with Evaluation Metrics

Model		MAE	MSE	RMSE	R^2
Decision Tree	Baseline model	10.21	87.79	9.37	0.62
	Hypertuned model	8.86	76.03	8.72	0.66
Random Forest	Baseline model	7.61	43.28	6.57	0.7
	Hypertuned model	6.81	39.73	6.3	0.74
ANN	Baseline model	8.91	84.20	9.17	0.67
	Hypertuned model	7.43	52.67	7.25	0.70
LSTM Model	Baseline model	12.86	168.52	12.98	0.54
	Hypertuned model	11.20	126.31	11.23	0.56

Figure 67 shows the comparison of bar plot for all the models that have been used in this research.

Figure 67

Comparison of Models in Bar Chart



Conclusions

From the results obtained, Random Forest demonstrated superior performance compared to other models that were used in the research after performing hyperparameter tuning on the model with an R^2 value of 0.74. When compared to ANN and LSTM, Random Forest outperformed them due to its capability to identify non-linear relationships, resistance to overfitting, and generalization ability. On the other hand, LSTM was the least effective model among the ones employed in the research, yielding an R2 value of 0.56. Among the Neural Networks algorithms, ANN performed better than LSTM, ranking second in terms of performance, just after Random Forest. The primary factor for this sub-par performance of Neural Networks algorithms is their dependency on a significant amount of data to learn patterns

and avoid overfitting which is one of the main reasons Random Forest performed better than all other models.

Limitations

The primary limitation of the study is that the model was trained at the county level with a limited dataset and predicted target values in the same county. This resulted in a shortage of data points to detect temporal dependencies and a significant number of irregularities in the test dataset, leading to a suboptimal performance from all the models. Basically, Decision Tree struggles with handling continuous and complex relationships between variables, which limits its ability to capture the complex relations of drought. Similarly, the Random Forest model is limited to predicting the values within the range of the training data, and it may struggle to make accurate predictions when given new data points outside of the training dataset. Both Neural Networks models face a significant limitation as they need huge data to predict accurate results which is lacking in this research. As a result, these models were prone to overfitting leading to suboptimal performance. In addition to the limited number of data points in the training dataset, the model lacked more features that contribute to the effects of drought and have an optimal correlation with the target variable.

Future Scope

In the future, to enhance the accuracy of the drought prediction model used in this research, the dataset could be expanded by incorporating additional features, such as underground water balances, soil moisture index, and average rainfall, among other things. By including these features, the dataset would be more complete, and the models would be able to identify more patterns in the data, resulting in better accuracy. Also, the models can be further refined by gathering more detailed data at the city level, enabling government officials to make

more accurate predictions and take appropriate action to mitigate the effects of drought in vulnerable regions.

In the future, if the data is more complex and varied, a hybrid approach involving models like Random Forest and XG Boost can be used to achieve more precise results. Bagging and boosting techniques can be applied to the ANN model by mixing numerous neural networks trained with various initial weights or structures. For LSTM, various time series analysis techniques can be applied to capture the patterns and trends over time such as autoregression or moving averages.

References

Agana, N. A., & Homaifar, A. (2017). A deep learning based approach for long-term drought prediction. *SoutheastCon 2017*. https://doi.org/10.1109/secon.2017.7925314

Alireza, N. S., Banafsheh, Z., & Mohsen, N. (2012). SEASONAL METEOROLOGICAL DROUGHT PREDICTION USING SUPPORT VECTOR MACHINE. *Journal of Water and Wastewater; Ab Va Fazilab (in Persian)*, *23*(282), 72–84. http://www.wwjournal.ir/article_1663_f5a9abfba84e1010776de62ba4f50896.pdf

Barua, S., Ng, A. K., & Perera, B. (2012). Artificial Neural Network–Based Drought Forecasting Using a Nonlinear Aggregated Drought Index. *Journal of Hydrologic Engineering*, *17*(12), 1408–1413. https://doi.org/10.1061/(asce)he.1943-5584.0000574

Belayneh, A., Adamowski, J., & Khalil, B. (2016). Short-term SPI drought forecasting in the Awash River Basin in Ethiopia using wavelet transforms and machine learning methods. *Sustainable Water Resources Management*, *2*(1), 87–101. https://doi.org/10.1007/s40899-015-0040-5

Bodri, L., & Čermàk, V. (2000). Prediction of extreme precipitation using a neural network: application to summer flood occurrence in Moravia. *Advances in Engineering Software*, *31*(5), 311–321. https://doi.org/10.1016/s0965-9978(99)00063-0

Breiman, L. (2001). Random Forests. Machine Learning. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/a:1010933404324

Chopda, J. V., Raveshiya, H., Nakum, S., & Nakrani, V. (2018). Cotton Crop Disease Detection using Decision Tree Classifier. *2018 International Conference on Smart City and Emerging Technology (ICSCET)*. https://doi.org/10.1109/icscet.2018.8537336

Dash, A., Jetley, S., Rege, A., Chopra, S., & Sawant, R. (2022). Drought Prediction and Water Quality Estimation using Satellite Images and Machine Learning. *2022 7th International Conference on Communication and Electronics Systems (ICCES)*. https://doi.org/10.1109/icces54183.2022.9835727

Dhyani, Y., & Pandya, R. J. (2021). Deep Learning Oriented Satellite Remote Sensing for Drought and Prediction in Agriculture. In *2021 IEEE 18th India Council International Conference (INDICON)*. https://doi.org/10.1109/indicon52576.2021.9691608

Ganguli, P., & Reddy, M. J. (2014). Ensemble prediction of regional droughts using climate inputs and the SVM-copula approach. *Hydrological Processes*, *28*(19), 4989–5009. https://doi.org/10.1002/hyp.9966

Granata, F. (2019). Evapotranspiration evaluation models based on machine learning algorithms—A comparative study. *Agricultural Water Management*, *217*, 303–315. https://doi.org/10.1016/j.agwat.2019.03.015

Hernández-Espinosa, N., Mondal, S., Autrique, E., González-Santoyo, H., Crossa, J., Huerta-Espino, J., Singh, R. P., & Guzmán, C. A. (2018). Milling, processing and end-use quality traits of CIMMYT spring bread wheat germplasm under drought and heat stress. *Field Crops Research*, *215*, 104–112. https://doi.org/10.1016/j.fcr.2017.10.003

Khan, N., Shahid, S., Shahid, S., Ahmed, K., Shiru, M. S., & Ahmed, K. (2020). Prediction of droughts over Pakistan using machine learning algorithms. *Advances in Water Resources*, *139*, 103562. https://doi.org/10.1016/j.advwatres.2020.103562

Kiem, A. S., Sharma, A., Westra, S., Van Dijk, A., Evans, J. P., O'Donnell, A., Rouillard, A., Barr, C., Tyler, J. J., Thyer, M., Jakob, D., Woldemeskel, F. M., Singh, V. P., & Mehrotra,

R. N. (2016). Natural hazards in Australia: droughts. *Climatic Change*, *139*(1), 37–54. https://doi.org/10.1007/s10584-016-1798-7

Kim, Y. H., Lee, K., & Kim, G. (2020). Forecast of drought index using decision tree based methods. *Journal of the Korean Data & Information Science Society*. https://doi.org/10.7465/jkdi.2020.31.2.273

Liu, H., Fu, M., Jin, X., Shang, Y., Shindell, D., Faluvegi, G., Shindell, C., & He, K. (2016). Health and climate impacts of ocean-going vessels in East Asia. *Nature Climate Change*, *6*(11), 1037–1041. https://doi.org/10.1038/nclimate3083

Liu, S., Kakani, V. G., Yang, X., & Yin, J. (2022). COVID-19 positive cases prediction based on LSTM algorithm and its variants. *2022 Asia Conference on Algorithms, Computing and Machine Learning (CACML)*. https://doi.org/10.1109/cacml55074.2022.00052

Lotfirad, M., Esmaeili-Gisavandani, H., & Adib, A. (2021). Drought monitoring and prediction using SPI, SPEI, and random forest model in various climates of Iran. *Journal of Water and Climate Change*, *13*(2), 383–406. https://doi.org/10.2166/wcc.2021.287

Masinde, M. (2014). Artificial neural networks models for predicting effective drought index: Factoring effects of rainfall variability. *Mitigation and Adaptation Strategies for Global Change*, *19*(8), 1139–1162. https://doi.org/10.1007/s11027-013-9464-0

Masroor, Rehman, S., Sajjad, H., Rahaman, M. H., Sajjad, H., Ahmed, R., & Singh, R. (2021). Assessing the impact of drought conditions on groundwater potential in Godavari Middle Sub-Basin, India using analytical hierarchy process and random forest machine learning algorithm. *Groundwater for Sustainable Development*, *13*, 100554. https://doi.org/10.1016/j.gsd.2021.100554

Mokhtar, A., Jalali, M., He, H., Al-Ansari, N., Elbeltagi, A., Alsafadi, K., Abdo, H. G., Sammen, S. S., Gyasi-Agyei, Y., & Rodrigo-Comino, J. (2021). Estimation of SPEI Meteorological Drought Using Machine Learning Algorithms. *IEEE Access*, *9*, 65503–65523. https://doi.org/10.1109/access.2021.3074305

Morid, S., Smakhtin, V., & Bagherzadeh, K. (2007). Drought forecasting using artificial neural networks and time series of drought indices. *International Journal of Climatology*, *27*(15), 2103–2111. https://doi.org/10.1002/joc.1498

Nabipour, N., Dehghani, M., Mosavi, A., & Shamshirband, S. (2020). Short-Term Hydrological Drought Forecasting Based on Different Nature-Inspired Optimization Algorithms Hybridized With Artificial Neural Networks. *IEEE Access*, *8*, 15210–15222. https://doi.org/10.1109/access.2020.2964584

Nourani, V., Baghanam, A. H., Adamowski, J., & Kisi, O. (2014). Applications of hybrid wavelet–Artificial Intelligence models in hydrology: A review. *Journal of Hydrology*, *514*, 358–377. https://doi.org/10.1016/j.jhydrol.2014.03.057

Poornima, S., & Pushpalatha, M. (2019). Drought prediction based on SPI and SPEI with varying timescales using LSTM recurrent neural network. *Soft Computing*, *23*(18), 8399–8412. https://doi.org/10.1007/s00500-019-04120-1

Prasad, N. E., Kumar, P. S., & Naidu, M. U. R. (2013). An Approach to Prediction of Precipitation Using Gini Index in SLIQ Decision Tree. *International Conference on Intelligent Systems, Modelling and Simulation*. https://doi.org/10.1109/isms.2013.27

Saltz, J. S. (2021). CRISP-DM for Data Science: Strengths, Weaknesses and Potential Next Steps. *2021 IEEE International Conference on Big Data (Big Data)*. https://doi.org/10.1109/bigdata52589.2021.9671634

Santos, J. a. C., Portela, M. M., & Pulido-Calvo, I. (2014). Spring drought prediction based on winter NAO and global SST in Portugal. *Hydrological Processes*, *28*(3), 1009–1024. https://doi.org/10.1002/hyp.9641

Sardar, V. S., Chaudhari, S., Anchalia, A., Kakati, A., Paudel, A., & N, B. B. (2022). Ensemble Learning with CNN and BMO for Drought Prediction. *2022 IEEE 3rd Global Conference for Advancement in Technology (GCAT)*. https://doi.org/10.1109/gcat55367.2022.9971991

Schäfer, F., Zeiselmair, C., Becker, J. N., & Otten, H. (2018). Synthesizing CRISP-DM and Quality Management: A Data Mining Approach for Production Processes. *2018 IEEE International Conference on Technology Management, Operations and Decisions (ICTMOD)*. https://doi.org/10.1109/itmc.2018.8691266

Senanayake, S., Pradhan, B., Al-Amri, A., & Park, H. (2022). A new application of deep neural network (LSTM) and RUSLE models in soil erosion prediction. *Science of the Total Environment*, *845*, 157220. https://doi.org/10.1016/j.scitotenv.2022.157220

Song, Y., & Lu, Y. (2015). Decision tree methods: applications for classification and prediction. *Shanghai Archives of Psychiatry*, *27*(2), 130–135. https://doi.org/10.11919/j.issn.1002-0829.215044

Swain, D. L., Langenbrunner, B., Neelin, J. D., & Hall, A. (2018). Increasing precipitation volatility in twenty-first-century California. *Nature Climate Change*, *8*(5), 427–433. https://doi.org/10.1038/s41558-018-0140-y

Tian, W., Wu, J., Cui, H., & Hu, T. (2021). Drought Prediction Based on Feature-Based Transfer Learning and Time Series Imaging. *IEEE Access*, *9*, 101454–101468. https://doi.org/10.1109/access.2021.3097353

Vicente-Serrano, S. M., Van Der Schrier, G., Beguería, S., Azorin-Molina, C., & López-Moreno, J. I. (2015). Contribution of precipitation and reference evapotranspiration to drought indices under different climates. *Journal of Hydrology*, *526*, 42–54. https://doi.org/10.1016/j.jhydrol.2014.11.025

Wang, W., Sun, L., Pei, Z. Y., Chen, Y., & Zhang, X. (2019). Analysis of Temporal and Spatial Variation of Growing Season Drought in Jiling Province Based on Standardized Precipitation Evapotranspiration Index. *International Conference on Agro-Geoinformatics*. https://doi.org/10.1109/agro-geoinformatics.2019.8820436

Wilhite, D. A., Svoboda, M., & Hayes, M. J. (2007). Understanding the complex impacts of drought: A key to enhancing drought mitigation and preparedness. *Water Resources Management*, *21*(5), 763–774. https://doi.org/10.1007/s11269-006-9076-5

Xu, L., Chen, N., Zhang, X., & Chen, Z. (2018). An evaluation of statistical, NMME and hybrid models for drought prediction in China. *Journal of Hydrology*, *566*, 235–249. https://doi.org/10.1016/j.jhydrol.2018.09.020f

Appendix A

Gantt Chart

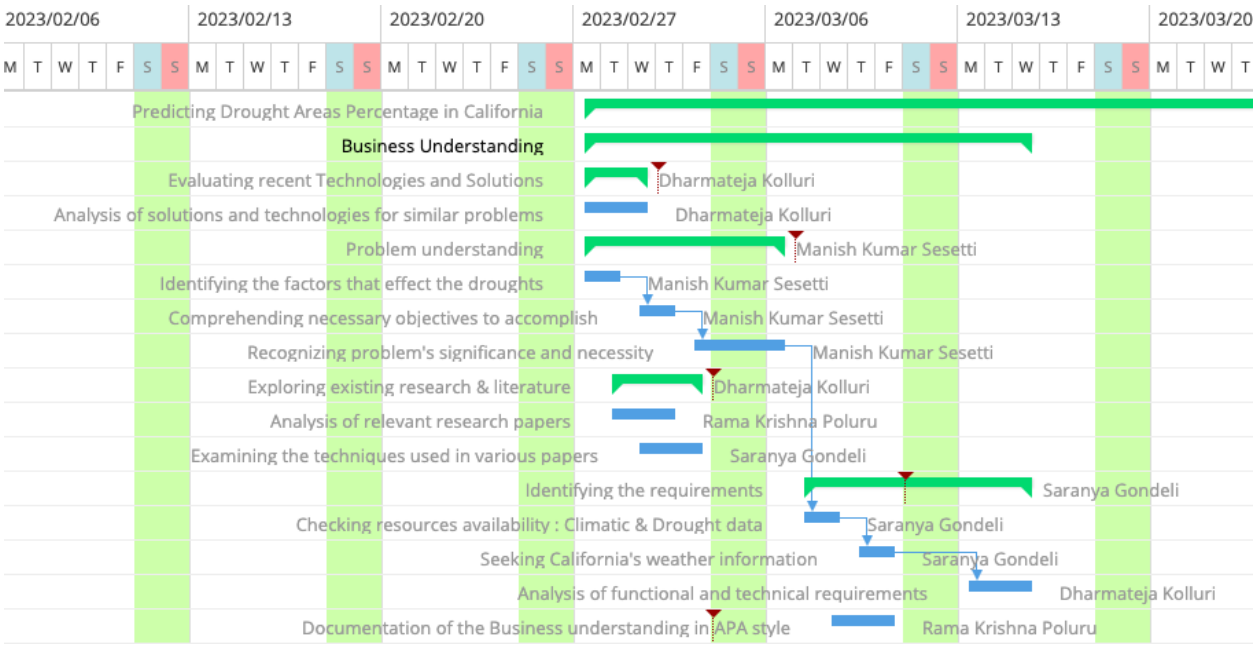


Figure A1. Gantt Chart - Business Understanding Phase

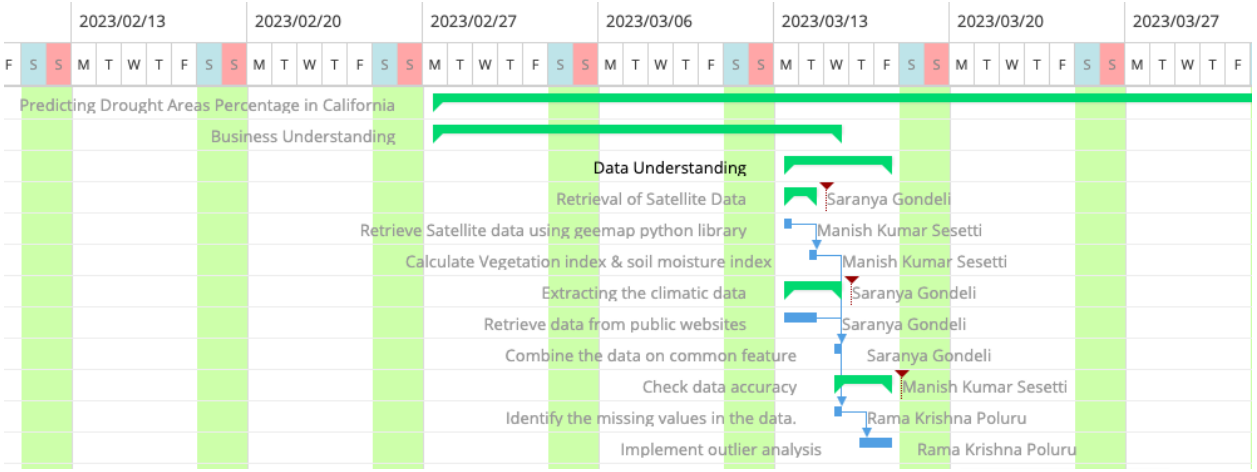


Figure A2. Gantt Chart - Data Understanding Phase

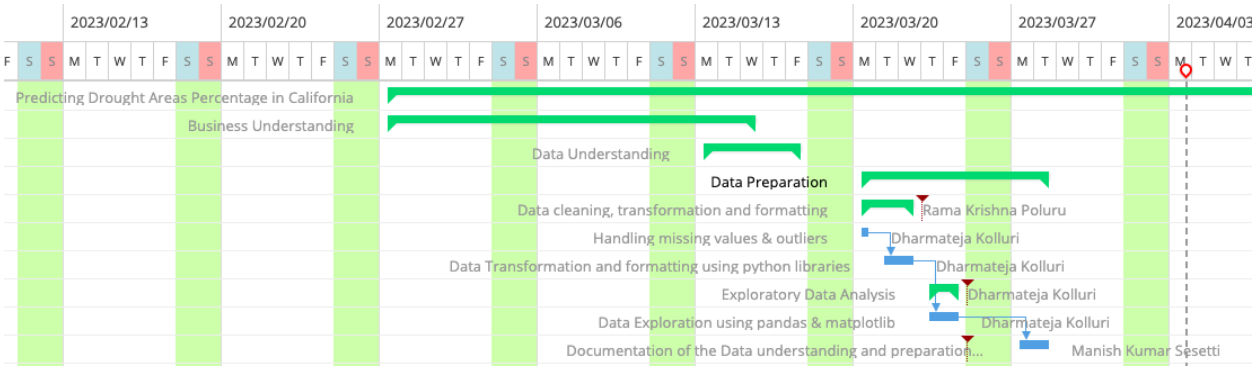


Figure A3. Gantt Chart - Data Preparation Phase

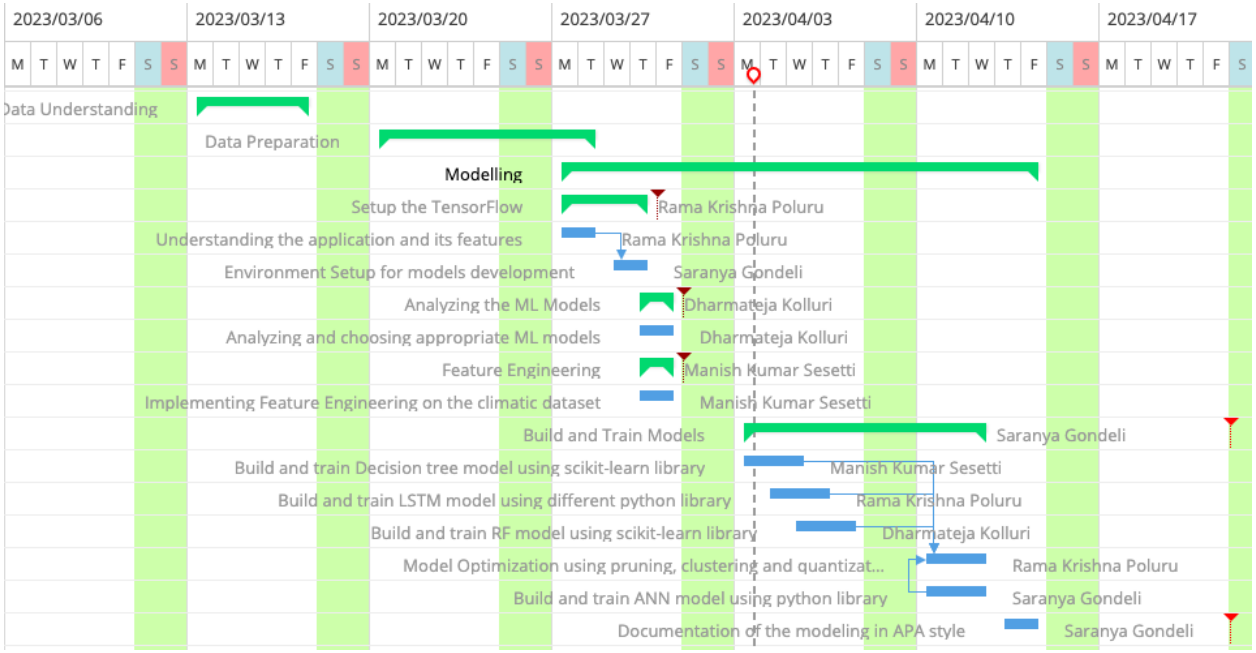


Figure A4. Gantt Chart - Modeling Phase

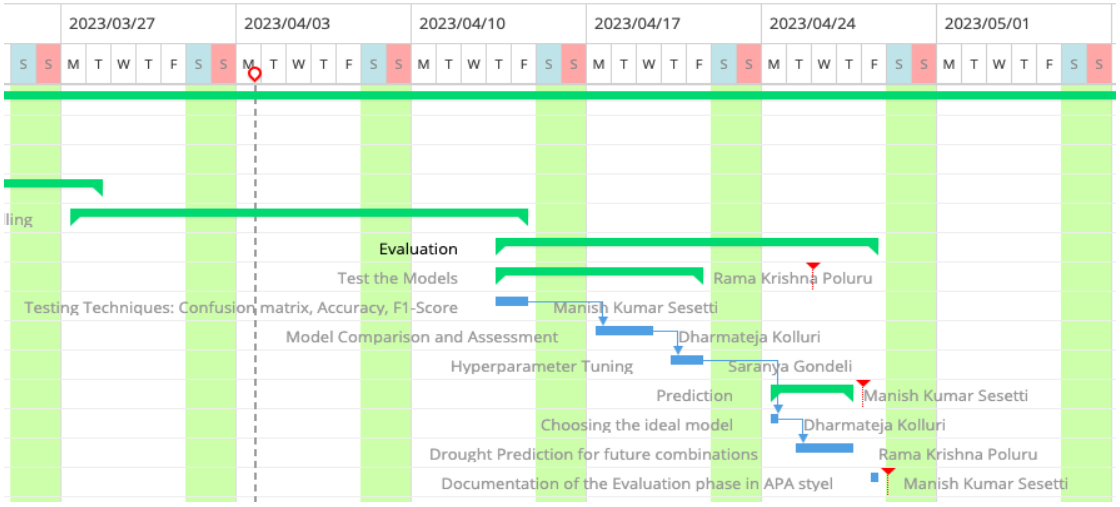


Figure A5. Gantt Chart - Evaluation Phase

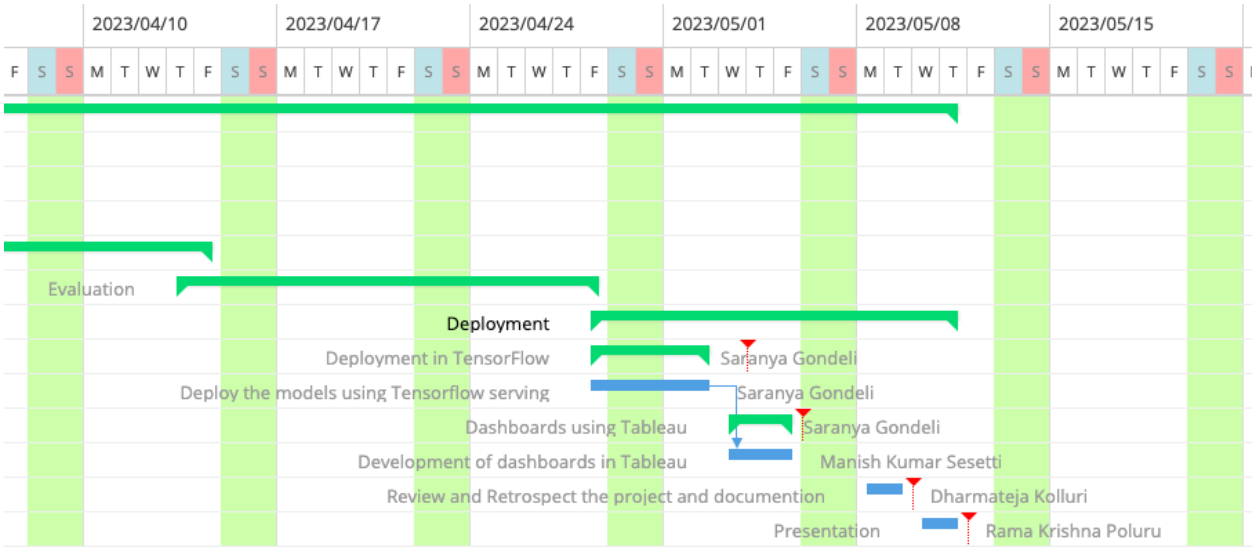


Figure A6. Gantt Chart - Deployment Phase

Appendix B

Python code snippets to perform modeling

```
import pandas as pd
import numpy as np
import itertools

# Create a date range from '2000-01-01' to '2020-12-31'
date_range = pd.date_range(start='2000-01-01', end='2020-12-31', freq='D')

# Create a DataFrame with a 'date' column containing the date range
df = pd.DataFrame({'dates': date_range})
# Print the DataFrame
df.head(2)
```

Figure B1. Code to implement data imputation by creating date column for drought data

```
county_list=['Alameda County','Alpine County','Butte County']
# use itertools.product to generate all combinations of list and DataFrame
Dates_And_County = itertools.product(county_list, df['dates'])
# create new DataFrame with all combinations
new_df = pd.DataFrame(Dates_And_County, columns=['county', 'dates'])
```

Figure B2. Code to combine dates and county values

```
import os
import pandas as pd

# specify the folder path where the CSV files are located
folder_path = '/Users/rkcabhi/Downloads/GWAR Drought Data/'

# get a list of all CSV files in the folder
csv_files = [file for file in os.listdir(folder_path) if file.endswith('.csv')]

# create an empty list to hold the DataFrames
dfs = []

# loop through each CSV file and append its DataFrame to the list
for file in csv_files:
    file_path = os.path.join(folder_path, file)
    dataf = pd.read_csv(file_path)
    # add a new column for the CSV file name without extension
    dataf['county'] = os.path.splitext(file)[0]
    dfs.append(dataf)

# concatenate all DataFrames into one
droughtdata = pd.concat(dfs, ignore_index=True)
droughtdata=droughtdata[droughtdata['Week']<='2020-12-31']
droughtdata.head(3)
```

Figure B3. Code for data integration combining all datasets

```
# convert 'dates' and 'Week' columns to datetime objects
new_df['dates'] = pd.to_datetime(new_df['dates'])
droughtdata['Week'] = pd.to_datetime(droughtdata['Week'])

# sort both dataframes by the join column
new_df = new_df.sort_values(by=['dates', 'county'])
droughtdata = droughtdata.sort_values(by=['Week', 'county'])

# merge the two dataframes
merged_df = pd.merge_asof(new_df.set_index('dates'),
                          droughtdata.set_index('Week'),
                          left_index=True, right_index=True,
                          direction='backward')

# reset index and fill missing values using the last occurred metric value
merged_df = merged_df.reset_index().fillna(method='ffill')

# rename the 'index' column to 'dates'
merged_df = merged_df.rename(columns={'index': 'dates'})

merged_df
```

Figure B4. Code for handling missing values

```
from keras.models import Sequential
from keras.layers import LSTM, Dense, Dropout
from keras.optimizers import Adam
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.model_selection import GridSearchCV
import numpy as np
from keras.callbacks import EarlyStopping
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt


class LstmModel:

    def __init__(self, n_lstm_units=50, n_dense_units=32, n_epochs=50, batch_size=32, dropout=0.1, lr=0.001):
        self.n_lstm_units = n_lstm_units
        self.n_dense_units = n_dense_units
        self.n_epochs = n_epochs
        self.batch_size = batch_size
        self.dropout = dropout
        self.lr = lr
        self.model = None

    def build_model(self,X_train):
        self.model = Sequential()
        self.model.add(LSTM(units=self.n_lstm_units, input_shape=(1,X_train.shape[2])))
        self.model.add(Dense(units=self.n_dense_units, activation='relu'))
        self.model.add(Dropout(self.dropout))
        self.model.add(Dense(units=1))
        self.model.compile(loss='mean_squared_error', optimizer=Adam(lr=self.lr))

    def summary_the_model(self):
        return self.model.summary()

    def fit(self, X_train, y_train,X_val,Y_val):
        self.model.fit(X_train, y_train,validation_data=(X_val,Y_val), epochs=self.n_epochs, batch_size=self.batch_size)

    def predict(self, X_test):
        y_pred = self.model.predict(X_test)
        return y_pred.flatten()

    def evaluate(self, y_true, y_pred):
        mae = mean_absolute_error(y_true, y_pred)
        mse = mean_squared_error(y_true, y_pred)
        rmse = mean_squared_error(y_true, y_pred, squared=False)
        return [mae,mse,rmse]

    def return_the_model(self):
        return self.model

    def hyperparameter_tuning(self, X_train, y_train):
        # Define the hyperparameter grid
        param_grid = {'n_lstm_units': [32, 64, 128], 'lr': [0.1, 0.01, 0.001], 'dropout': [0.1, 0.2, 0.5]}

        # Initialize GridSearchCV with the model, parameter grid, and scoring metric
        grid_search = GridSearchCV(estimator=self.model, param_grid=param_grid, scoring='neg_mean_squared_error', cv=3)

        # Fit GridSearchCV on your training data
        grid_search.fit(X_train, y_train)

        # Retrieve the best hyperparameters and use them to build the final model
        best_params = grid_search.best_params_
        self.n_lstm_units = best_params['n_lstm_units']
        self.lr = best_params['lr']
        self.dropout = best_params['dropout']
        self.build_model(X_train)
```

Figure B5. Code for implementing the LSTM model

```
from keras.models import Sequential
from keras.layers import Dense, Dropout
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.callbacks import EarlyStopping
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.model_selection import GridSearchCV
import numpy as np
from keras.callbacks import EarlyStopping

class ANN:

    def __init__(self, input_dim, output_dim, hidden_layers=2, hidden_units=64, dropout_rate=0.2, epochs=50, batch_size=32):
        self.input_dim = input_dim
        self.output_dim = output_dim
        self.hidden_layers = hidden_layers
        self.hidden_units = hidden_units
        self.dropout_rate = dropout_rate
        self.epochs = epochs
        self.batch_size = batch_size
        self.model = None

    def build_model(self):
        model = Sequential()
        model.add(Dense(self.hidden_units, activation='relu', input_dim=self.input_dim))
        model.add(Dropout(self.dropout_rate))
        for i in range(self.hidden_layers - 1):
            model.add(Dense(self.hidden_units, activation='relu'))
            model.add(Dropout(self.dropout_rate))
        model.add(Dense(self.output_dim))
        model.compile(optimizer='adam', loss='mean_squared_error')
        self.model = model

    def summary_the_model(self):
        return self.model.summary()

    def fit(self, X_train, y_train):
        if not self.model:
            self.build_model()
        es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=10)
        self.model.fit(X_train, y_train, epochs=self.epochs, batch_size=self.batch_size, validation_split=0.2, callbacks=[es])

    def predict(self, X_test):
        y_pred = self.model.predict(X_test)
        return y_pred

    def evaluate(self, y_true, y_pred):
        mae = mean_absolute_error(y_true, y_pred)
        mse = mean_squared_error(y_true, y_pred)
        rmse = mean_squared_error(y_true, y_pred, squared=False)
        #rmsle = mean_squared_error(np.log(y_true), np.log(y_pred), squared=False)
        #r2 = r2_score(y_true, y_pred)
        return [mae,mse,rmse]


    def return_the_model(self):
        return self.model
```

Figure B6. Code for implementing the ANN model

```
from sklearn.ensemble import RandomForestRegressor

import numpy as np
import pandas as pd
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor

class RandomForest:

    def __init__(self, n_estimators=100, criterion='friedman_mse', max_depth=None, min_samples_split=2, min_samples_leaf=1):
        self.n_estimators = n_estimators
        self.criterion = criterion
        self.max_depth = max_depth
        self.min_samples_split = min_samples_split
        self.min_samples_leaf = min_samples_leaf
        self.model = RandomForestRegressor(n_estimators=n_estimators, criterion=criterion, max_depth=max_depth,
                                           min_samples_split=min_samples_split, min_samples_leaf=min_samples_leaf)

    def fit(self, X_train, y_train):
        self.model.fit(X_train, y_train)

    def predict(self, X_test):
        y_pred = self.model.predict(X_test)
        return y_pred

    def evaluate(self, y_true, y_pred):
        mae = mean_absolute_error(y_true, y_pred)
        mse = mean_squared_error(y_true, y_pred)
        rmse = mean_squared_error(y_true, y_pred, squared=False)
        return [mae,mse,rmse]

    def hyperparameter_tuning(self, X_train, y_train, param_grid):
        grid_search = GridSearchCV(estimator=self.model, param_grid=param_grid, cv=5, n_jobs=-1)
        grid_search.fit(X_train, y_train)
        print("Best parameters: ", grid_search.best_params_)
        self.model = grid_search.best_estimator_

    def return_the_model(self):
        return self.model
```

Figure B7. Code for implementing the RF model

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.preprocessing import StandardScaler, MinMaxScaler, LabelEncoder


class FeatureUnderstanding:

    def __init__(self, data):
        self.data = data

    def correlation_matrix(self, figsize=(10, 10)):
        corr_matrix = self.data.corr()
        plt.figure(figsize=figsize)
        sns.heatmap(corr_matrix, cmap='YlGnBu', annot=True)
        plt.show()

    def scatterplot(self, x, y, figsize=(10, 8)):
        plt.figure(figsize=figsize)
        sns.scatterplot(data=self.data, x=x, y=y)
        plt.title("Scatter plot {} vs Target: {}".format(x,y))
        plt.show()

    def boxplot(self, x, y, figsize=(10, 8)):
        plt.figure(figsize=figsize)
        sns.boxplot(data=self.data, x=x, y=y)
        plt.show()
```

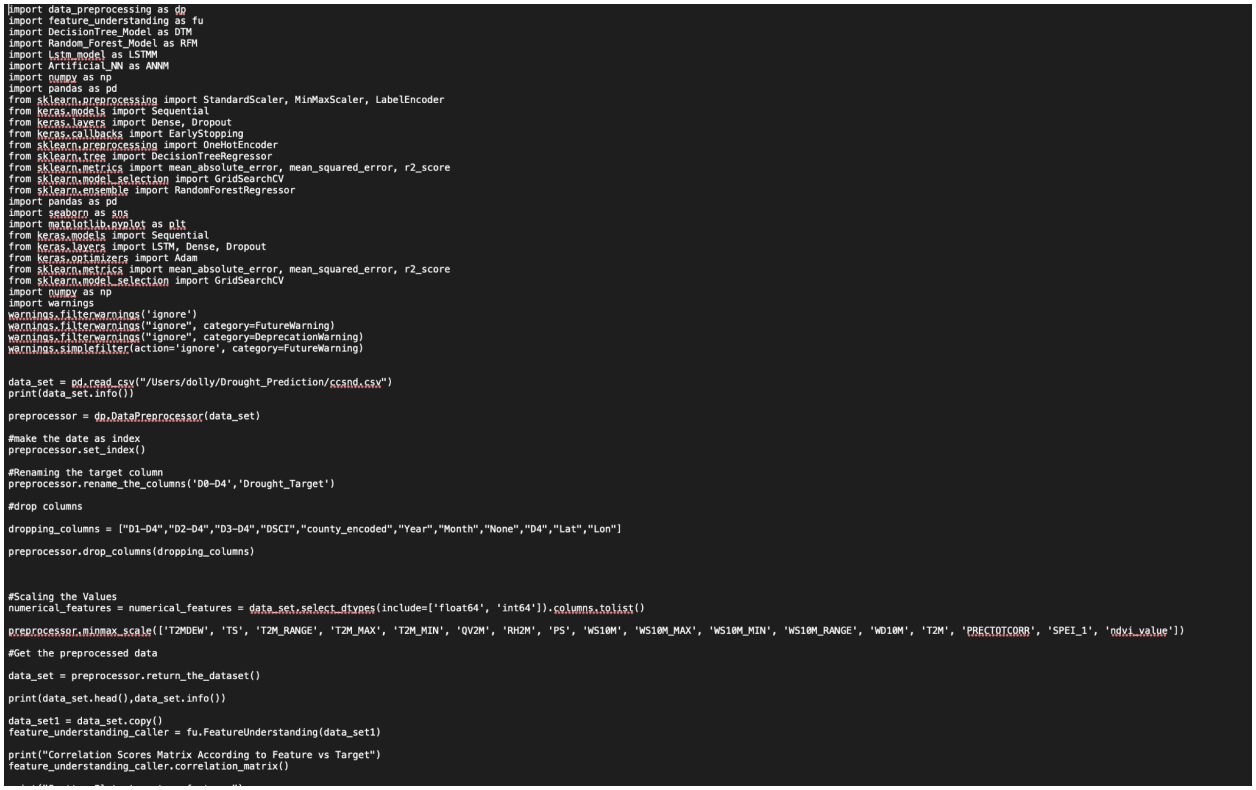
Figure B8. Code for implementing feature understanding

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.model_selection import GridSearchCV

import numpy as np
import pandas as pd

from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.model_selection import GridSearchCV
class DecisionTree:

    def __init__(self, criterion='mse', max_depth=None, min_samples_split=2, min_samples_leaf=1):
        self.criterion = criterion
        self.max_depth = max_depth
        self.min_samples_split = min_samples_split
        self.min_samples_leaf = min_samples_leaf
        self.model = DecisionTreeRegressor(criterion=criterion, max_depth=max_depth,
                                           min_samples_split=min_samples_split, min_samples_leaf=min_samples_leaf)

    def fit(self, X_train, y_train):
        self.model.fit(X_train, y_train)

    def predict(self, X_test):
        y_pred = self.model.predict(X_test)
        return y_pred

    def evaluate(self, y_true, y_pred):
        mae = mean_absolute_error(y_true, y_pred)
        mse = mean_squared_error(y_true, y_pred)
        rmse = mean_squared_error(y_true, y_pred, squared=False)
        #rmsle = mean_squared_error(np.log(y_true), np.log(y_pred), squared=False)
        return [mae,mse,rmse]

    def hyperparameter_tuning(self, X_train, y_train, param_grid):
        grid_search = GridSearchCV(estimator=self.model, param_grid=param_grid, cv=5, n_jobs=-1)
        grid_search.fit(X_train, y_train)
        print("Best parameters: ", grid_search.best_params_)
        self.model = grid_search.best_estimator_

    def return_the_model(self):
        return self.model
```

Figure B9. Code for implementing Decision Tree model

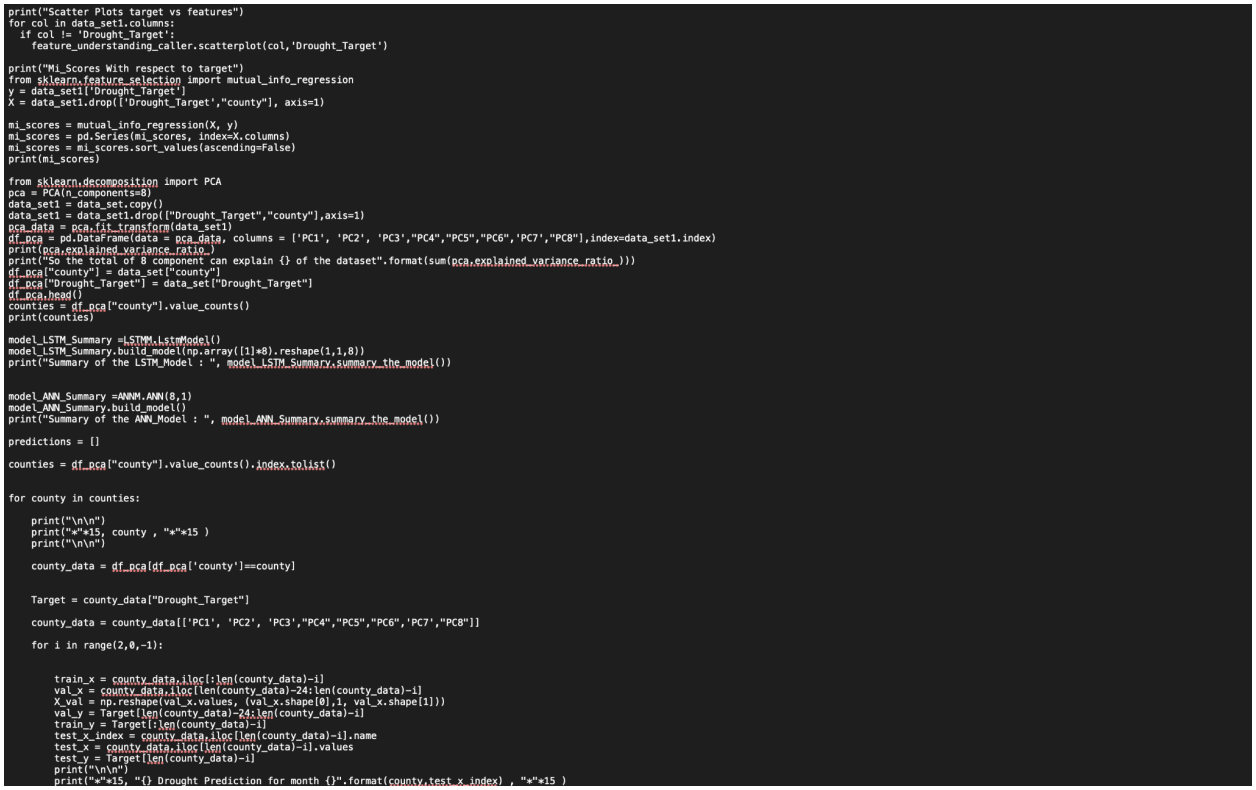
```
import pandas as pd
from sklearn.preprocessing import StandardScaler, MinMaxScaler, LabelEncoder
import numpy as np
from sklearn.preprocessing import OneHotEncoder
import data_preprocessing as dp
import feature_understanding as fu
import DecisionTree_Model as DTM
import Random_Forest_Model as RFM
import Lstm_model as LSTMM
import Artificial_NN as ANNM
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler, MinMaxScaler, LabelEncoder
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.callbacks import EarlyStopping
from sklearn.preprocessing import OneHotEncoder
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import LSTM, Dense, Dropout
from keras.optimizers import Adam
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.model_selection import GridSearchCV
import numpy as np
class DataPreprocessor:
    def __init__(self, data):
        self.data = data

    def drop_columns(self, columns):
        self.data = self.data.drop(columns=columns)

    def standard_scale(self, columns):
        scaler = StandardScaler()
        self.data[columns] = scaler.fit_transform(self.data[columns])

    def minmax_scale(self, columns):
        scaler = MinMaxScaler()
        self.data[columns] = scaler.fit_transform(self.data[columns])

    def replace_missing_values(self, method='mean'):
        if method == 'mean':
            self.data = self.data.fillna(self.data.mean())
        elif method == 'median':
            self.data = self.data.fillna(self.data.median())
        elif method == 'mode':
            self.data = self.data.fillna(self.data.mode().iloc[0])

    def datetime_to_features(self, column):
        self.data[column] = pd.to_datetime(self.data[column])
        self.data['Month'] = self.data[column].dt.month
        self.data['Year'] = self.data[column].dt.year
        self.data = self.data.drop(columns=[column])

    def set_index(self):
        self.data = self.data.set_index('Date')

    def rename_the_columns(self, column_name, rename_name):
        self.data = self.data.rename(columns={column_name:rename_name})
    def return_the_dataset(self):
        return self.data
```

Figure B10. Code for implementing data pre-processing Part A



```
import data_preprocessing as dp
import feature_understanding as fu
import DecisionTree_Model as DTM
import Random_Forest_Model as RFM
import Lstm_model as LSTMM
import Artificial_NN as ANNM
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler, MinMaxScaler, LabelEncoder
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.callbacks import EarlyStopping
from sklearn.preprocessing import OneHotEncoder
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import LSTM, Dense, Dropout
from keras.optimizers import Adam
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.model_selection import GridSearchCV
import numpy as np
import warnings
warnings.filterwarnings('ignore')
warnings.filterwarnings("ignore", category=FutureWarning)
warnings.filterwarnings("ignore", category=DeprecationWarning)
warnings.simplefilter(action='ignore', category=FutureWarning)


data_set = pd.read_csv("/Users/dolly/Drought_Prediction/ccsnd.csv")
print(data_set.info())

preprocessor = dp.DataPreprocessor(data_set)

#make the date as index
preprocessor.set_index()

#Renaming the target column
preprocessor.rename_the_columns('D0-D4','Drought_Target')

#drop columns

dropping_columns = ["D1-D4","D2-D4","D3-D4","DSCI","county_encoded","Year","Month","None","D4","Lat","Lon"]

preprocessor.drop_columns(dropping_columns)



#Scaling the Values
numerical_features = numerical_features = data_set.select_dtypes(include=['float64', 'int64']).columns.tolist()

preprocessor.minmax_scale(['T2MDEW', 'TS', 'T2M_RANGE', 'T2M_MAX', 'T2M_MIN', 'QV2M', 'RH2M', 'PS', 'WS10M', 'WS10M_MAX', 'WS10M_MIN', 'WS10M_RANGE', 'WD10M', 'T2M', 'PRECTOTCORR', 'SPEI_1', 'ndvi_value'])

#Get the preprocessed data

data_set = preprocessor.return_the_dataset()

print(data_set.head(),data_set.info())

data_set1 = data_set.copy()
feature_understanding_caller = fu.FeatureUnderstanding(data_set1)

print("Correlation Scores Matrix According to Feature vs Target")
feature_understanding_caller.correlation_matrix()
```

Figure B11. Code for implementing data preprocessing Part B



```
print("Scatter Plots target vs features")
for col in data_set1.columns:
  if col != 'Drought_Target':
    feature_understanding_caller.scatterplot(col,'Drought_Target')

print("Mi_Scores With respect to target")
from sklearn.feature_selection import mutual_info_regression
y = data_set1['Drought_Target']
X = data_set1.drop(['Drought_Target',"county"], axis=1)

mi_scores = mutual_info_regression(X, y)
mi_scores = pd.Series(mi_scores, index=X.columns)
mi_scores = mi_scores.sort_values(ascending=False)
print(mi_scores)

from sklearn.decomposition import PCA
pca = PCA(n_components=8)
data_set1 = data_set.copy()
data_set1 = data_set1.drop(["Drought_Target","county"],axis=1)
pca_data = pca.fit_transform(data_set1)
df_pca = pd.DataFrame(data = pca_data, columns = ['PC1', 'PC2', 'PC3',"PC4","PC5","PC6",'PC7',"PC8"],index=data_set1.index)
print(pca.explained_variance_ratio_)
print("So the total of 8 component can explain {} of the dataset".format(sum(pca.explained_variance_ratio_)))
df_pca["county"] = data_set["county"]
df_pca["Drought_Target"] = data_set["Drought_Target"]
df_pca.head()
counties = df_pca["county"].value_counts()
print(counties)

model_LSTM_Summary =LSTMM.LstmModel()
model_LSTM_Summary.build_model(np.array([1]*8).reshape(1,1,8))
print("Summary of the LSTM_Model : ", model_LSTM_Summary.summary_the_model())


model_ANN_Summary =ANNM.ANN(8,1)
model_ANN_Summary.build_model()
print("Summary of the ANN_Model : ", model_ANN_Summary.summary_the_model())

predictions = []

counties = df_pca["county"].value_counts().index.tolist()


for county in counties:

    print("\n\n")
    print("*"*15, county , "*"*15 )
    print("\n\n")

    county_data = df_pca[df_pca['county']==county]


    Target = county_data["Drought_Target"]

    county_data = county_data[['PC1', 'PC2', 'PC3',"PC4","PC5","PC6",'PC7',"PC8"]]

    for i in range(2,0,-1):


        train_x = county_data.iloc[:len(county_data)-i]
        val_x = county_data.iloc[len(county_data)-24:len(county_data)-i]
        X_val = np.reshape(val_x.values, (val_x.shape[0],1, val_x.shape[1]))
        val_y = Target[len(county_data)-24:len(county_data)-i]
        train_y = Target[:len(county_data)-i]
        test_x_index = county_data.iloc[len(county_data)-i].name
        test_x = county_data.iloc[len(county_data)-i].values
        test_y = Target[len(county_data)-i]
        print("\n\n")
        print("*"*15, "{} Drought Prediction for month {}".format(county,test_x_index) , "*"*15 )
```

Figure B12. Code for implementing PCA

```
        print("*"*15, "{} Drought Prediction for month {}".format(county,test_x_index) , "*"*15 )
        print("\n\n")

        DT = DTM.DecisionTree()
        DT_param_grid = {
    'criterion': [ 'friedman_mse'],
    'max_depth': [5, 10, 20],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2,],
    'max_features': ['auto']
     }


        DT.hyperparameter_tuning(train_x,train_y,DT_param_grid)
        DT_y_pred = DT.predict(test_x.reshape(1,-1))
        print(DT_y_pred,test_y)
        DT_eval_results = DT.evaluate(DT_y_pred,[test_y])
        predictions.append({"County": county, "Algorithm": "Decision Tree", "Month": test_x_index,
                            "Actual": test_y, "Predicted": DT_y_pred, "MAE": DT_eval_results[0],
                            "MSE": DT_eval_results[1], "RMSE": DT_eval_results[2]})
        #Random Forest Model
        RF = RFM.RandomForest()
        RT_param_grid = {
          'n_estimators': [50, 100],
          'max_features': ['auto', 'sqrt', 'log2'],
          'max_depth': [5, 10],
          'min_samples_split': [2, 5],
          'min_samples_leaf': [1, 2],
        }
        RF.hyperparameter_tuning(train_x,train_y,RT_param_grid)
        RF_y_pred = RF.predict(test_x.reshape(1,-1))
        RF_eval_results = RF.evaluate(RF_y_pred, [test_y])

        predictions.append({"County": county, "Algorithm": "Random Forest","Month": test_x_index, "Actual": test_y,
"Predicted":RF_y_pred ,"MAE" :RF_eval_results[0],"MSE":RF_eval_results[1],"RMSE":RF_eval_results[2]})

        #LSTM Model
        train_x = np.reshape(train_x.values, (train_x.shape[0],1, train_x.shape[1]))
        test_x = np.reshape(test_x,(1,1,8))
        val_x = county_data.iloc[len(county_data)-24:len(county_data)-i]
        X_val = np.reshape(val_x.values, (val_x.shape[0],1, val_x.shape[1]))

        LSTM_obj = LSTMM.LstmModel()
        LSTM_obj.build_model(train_x)
        LSTM_obj.fit(train_x,train_y,X_val,val_y)
        LSTM_y_pred = LSTM_obj.predict(test_x)
        LSTM_eval_results = LSTM_obj.evaluate([test_y],LSTM_y_pred)

        predictions.append({"County": county, "Algorithm":"LSTM","Month": test_x_index, "Actual": test_y, "Predicted": LSTM_y_pred,"MAE" :LSTM_eval_results[0],"MSE":LSTM_eval_results[1],"RMSE":LSTM_eval_results[2]})
        #ANN Model
        train_x = county_data.iloc[:len(county_data)-i]
        val_x = county_data.iloc[len(county_data)-24:len(county_data)-i]
        val_y = Target[len(county_data)-24:len(county_data)-i]
        train_y = Target[:len(county_data)-i]
        test_x_index = county_data.iloc[len(county_data)-i].name
        test_x = county_data.iloc[len(county_data)-i].values.reshape(1,8)
        test_y = Target[len(county_data)-i]

        ANN_Model = ANNM.ANN(input_dim=train_x.shape[1],output_dim = 1)
        ANN_Model.build_model()
        ANN_Model.fit(train_x,train_y)
        ANN_y_pred = ANN_Model.predict(test_x)
        ANN_eval_results  = ANN_Model.evaluate([test_y],ANN_y_pred)

        predictions.append({"County": county, "Algorithm":"ANN","Month": test_x_index, "Actual": test_y, "Predicted": ANN_y_pred,"MAE" :ANN_eval_results[0],"MSE":ANN_eval_results[1],"RMSE":ANN_eval_results[2] })
df = pd.DataFrame(predictions)

# Save DataFrame as CSV file
df.to_csv('result_data.csv', index=False)
```

Figure B13. Code for training and testing the models

Alameda

In [10]:
```
import ee
ee.Initialize()

# Define the study area
# study_area = ee.FeatureCollection("TIGER/2018/States").filterMetadata('NAME', 'equals', 'California')

alamedaBounds = ee.Geometry.Rectangle([-122.40, 37.40, -121.50, 38.00])
counties = ee.FeatureCollection('TIGER/2018/Counties') \
            .filter(ee.Filter.eq('NAME', 'Alameda')) \
            .filterBounds(alamedaBounds)

# Define the NDVI function
def addNDVI(image):
    # Select bands and set projection
    bands = ['B4', 'B5']
    proj = image.select('B4').projection()
    image = image.select(bands).reproject(proj)

    # Reduce resolution of image to 60 meters
    image = image.reduceResolution(
        reducer=ee.Reducer.mean(),
        maxPixels=65535
    ).reproject(
        crs=image.projection(),
        scale=60
    )

    ndvi = image.normalizedDifference(['B5', 'B4']).rename('NDVI')
    return image.addBands(ndvi)

# Load the Landsat collection and filter by date and study area
collection = ee.ImageCollection('LANDSAT/LE07/C02/T1')\
              .filterBounds(counties)\
              .filterDate('2000-01-01', '2020-12-31')\
              .map(addNDVI)

# Define a function to get the date and NDVI value
def get_ndvi(image):
    return ee.Feature(None, {'date': image.date().format('YYYY-MM-dd'), 'NDVI': image.select('NDVI').reduceReg

# Map the function over the collection
ndvi_collection = collection.map(get_ndvi)
# Convert the collection to a list and print the results
ndvi_list = ndvi_collection.getInfo()['features']
for item in ndvi_list:
    ee_dict = ee.Dictionary(item['properties'])
    has_ndvi = ee.Algorithms.If(ee_dict.contains('NDVI'), True, False)
    if has_ndvi.getInfo():
        df_ndvi.loc[len(df_ndvi.index)] = [item['properties']['date'], item['properties']['NDVI']]
#         print(item['properties']['date'], item['properties']['NDVI'])

print(df_ndvi)
df_ndvi.to_csv('alameda_NDVI.csv', index=False)
```

Figure B14. Code for retrieving the NDVI values for counties