California Wildfire Prediction: Machine Learning

Kaylee Pham, David Shin, Saulo Rubio, Sergio Ramirez
Tyler Poplawski, David Macoto Ward, Lior Zlotikman*
{kaylee.pham.704,david.shin.904,saulo.rubio.491,sergio.ramirez.754}@my.csun.edu
{tyler.poplawski.456,david.ward.761,lior.zlotikman.183}@my.csun.edu
Department of Computer Science
California State University, Northridge
Northridge, California

Figure 1: A firefighter works during the Creek Fire in Madera County, California on Sep 7th, 2020. Photographer: Josh Edelson/AFP via Getty Images

ABSTRACT

The occurrence of wildfires in California are frequent and cause a vast range of damages to the land and society. This project strives to mitigate the damages from wildfires in California by implementing machine learning techniques. Previous researches found on the topic have low accuracy scores, focus on small portions of California, or use various other features to predict wildfires. In project, we predict the conditions across California that would be most likely to cause more wildfire by using the Land Surface Temperature, Normalized Vegetation Index, Thermal Anomalies, and Fire Incident Record data as input for our models. As a result, the accuracy score of the each model is as follows: Artificial Neural Network 81%, Support Vector Machine 93%, K-Nearest Neighbors 87%, Gaussian Naive Bayes 85%, and Logistic Regression 84%. We found the Support Vector Machine had the accuracy score of 93% in predicting the conditions of wildfires across California.

KEYWORDS

Neural Networks, Support Vector Machine, K-Nearest Neighbors, Gaussian Naive Bayes, Logistic Regression, Remote Sensing Data, Spatio-temporal data, Wildfire, Prediction, MODIS, Land Surface Temperature, Normalized Vegetation Index, Thermal Anomalies, California

1 INTRODUCTION

While much research has been conducted on how to predict wildfires with great accuracy, wildfires are still a major ongoing issue due to climate change. Between January and October 2020 wildfires burned 3.75 million acres, killed 29 people, and destroyed 8,169 structures in California [1]. Thus, a better model is still needed to predict the creation and spread of wildfires. The goal of this project is to create a reliable and accurate wildfire prediction model to help our community in solving this major problem. The project design begins with preprocessing and inputting the data to the Neural Network (NN), Support Vector Machines (SVM), K-Nearest Neighbors, Gaussian Naive Bayes (GNB) and Logistic Regression (LR) models. These models in turn predict the conditions in which wildfire are likely to occur in California. The final part of the design measures the quality and accuracy of the prediction.

*All authors contributed equally to this research.

COMP490 Senior Design Project, California State University, Northridge,

The rest of this paper is organized as following: Section 2 discusses related work; Section 3 presents the framework of the project; Section 4 discusses the implementation details of the machine learning models; Section 5 presents the experimental results; and Section 6 concludes the project and discusses future work.

2 RELATED WORK

There has been a sizable amount of work done on the topic already, with previous groups used different combinations of machine learning models as well as datasets with varying success.

2.1 Datasets

Three groups had a large influence on the data chosen, Sayad et al. [12] among others [10] and [14], as the same combination of data that they used was also chosen for the project: past fire records, satellite data, and meteorological data. The Sayad et al. paper was especially influential in choosing features correlated to fires given the very accurate results and the overlap of their three chosen features being common among many other groups who also achieved very accurate results. The features selected in the Sayad et al. group were: normalized difference vegetation index, land surface temperature, and thermal anomalies.

A number of groups used some combination of the same datasets mentioned above. Zhang et al. [15] used a conjunction of satellite images and past fire record data. Storer and Green [13] and Liang et al. [7] both used a combination of past fire record databases and meteorological data to train their models. Lall and Mathibela [6] showed promising results using vegetation and environmental features to train their highly accurate artificial NN. Perumal and Zyl's [9] focused on satellite data, specifically the Visible Infrared Imaging Radiometer Suite (VIIRS) satellite instrument for the area of South Africa. The authors indicated for future work that MODIS data would likely improve upon their work, which also played a role in the decision to use such data.

2.2 Support Vector Machine

Researchers use a combination of meteorological data and the US forest fire database. The meteorological data consists of temperature data as it pertains to weather, humidity, rain and snowfall levels among other data. The US forest fire database included data such as geographical coordinates, area affected by fire, and severity, all in comma separated value (csv) format. The two datasets were chosen due to their reliability. Researchers demonstrate differences between some of the most popular machine learning models, and identifies SVM as having the greatest accuracy in this domain. Additionally, in this comparison, data fusion and binary and multiclass classification were used [8].

Researchers use data from Moderate Resolution Imaging Spectroradiometer (MODIS) satellite sensors. We haven't determined the exact features we will select for our database but we will likely choose features related to what they used: normalized difference vegetation index (NDVI or health of crops), land surface temperatures (LST), and thermal anomalies. This paper also compares Neural Networks against SVM, and demonstrated that in their test, a NN has 98.32% accuracy while an SVM has 97.48% [12].

2.3 Neural Network

In a research paper by Liang et al., three different neural network models were tested against each other, those being a BPNN model, a Recurrent NN model, and the LSTM model mentioned earlier. The goal was to determine which of these methods is best to build a prediction model in order to help firefighters and emergency personnel assess the risk and spread of a fire before it grows too large. The results of this paper conclude that the LSTM model produced the best predictions of the three, with an accuracy of 90.9% [7]. Contrary to this discovery, a conference paper by Lall and Mathibela used a slight modification of BPNN called the Resilient Back-Propagation algorithm (RPROP), which resulted in the system's overall performance having an accuracy of 97%, as well as 87% precision and 88% recall [6].

Both papers proved to have created successful wildfire risk prediction systems, however they also faced similar drawbacks such as ensuring the model was not overfitting the data. To avoid this, Liang et al. performed a multi-collinearity test on the data to remove factors that were proven to skew the model's interpretation rather than benefit it. They also made note that an overall limitation of the system was due to the modeling data coming from a single area [7].

The authors of this paper, [13], propose the Particle Swarm Optimization (PSO) algorithm to train a Neural Network. For this, they use the RMSE(Root Mean Squared Error) to compare the results of the PSO algorithm with the results of a Backpropagation algorithm.

This paper overlaps with our project in using satellite images and past fire record data along with CNNs [15]. They created a model where satellite images of past fire data and a 2D CNN, with 2 convolutional layers with 32 and 64 nodes using Sigmoid and ReLU activation functions respectively.

3 DESIGN

This project is composed of three major components: preparation, model building, and quality measurement Fig. 2.

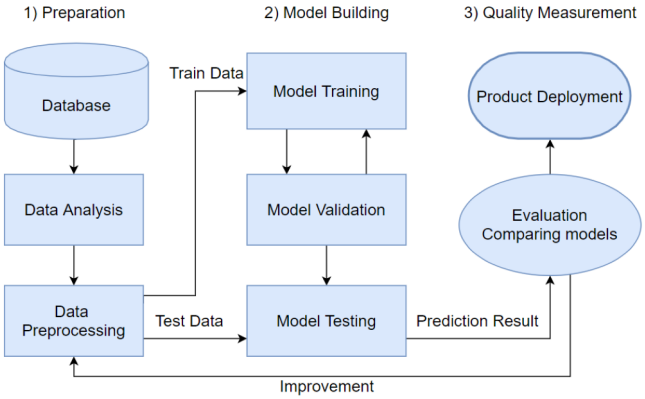


Figure 2: Design Overview

3.1 Data Preparation

The first part of preparation is to gather satellite images from the Land Processes Distributed Active Archive Center (LP DAAC). The

products used for this project are MOD13A1 500m 16 days for NDVI, MOD11A1 1 km for Land Surface Temperature, VPN14A1 1km for Thermal Anomalies, and MCD64A1 500m for Burn Area. The next step is to obtain a California forest fire records list from the California Department of Forestry and Fire Protection (CAL FIRE). The years analyzed on this project are 2019 and 2020. The final step of preparation is exploring and cleaning the satellite images and extracting the quality values from them. Then, apply feature engineering to the California fire record dataset, and preprocess all the collected data into a new dataset to be used in the next component.

3.2 Model Building

For model building, the first step is splitting the data into two subsets for training and testing. Next, feed the data into training model, validate the model, and test the model with the testing data subset. Finally, output the prediction result.

3.2.1 Neural Network. The neural network model used follows a feed forward network style and makes use of the backpropagation algorithm. "Feed forward" refers to a non-cyclical style of neural network, in which the data is passed through multiple layers of neurons strictly from input to output without looping back. Backpropagation is used to help identify which neurons contribute to errors in the output from the network. When errors are encountered, the weights between connections are modified in an attempt to guide the network to a more accurate result. [5]

Referring to Fig. 3, the following steps of the neural network flow can be traced. First, the training set of data is passed into input neurons and initial weights are chosen. This data is fed through the network and a linear combination of the inputs and weights is calculated, which becomes the input for the hidden layers. From this layer the data passes through a Rectified Linear Unit (ReLU) activation function and is compared against a threshold value to determine if that neuron's output is passed onto the next layer. This method is repeated between each of the hidden layers. A final linear combination is again calculated before arriving at the output layer. Here is where error checking takes place and weights are modified as needed. Lastly, this entire training process is repeated until the best fitting weights have been discovered and the model's accuracy is maximally improved.

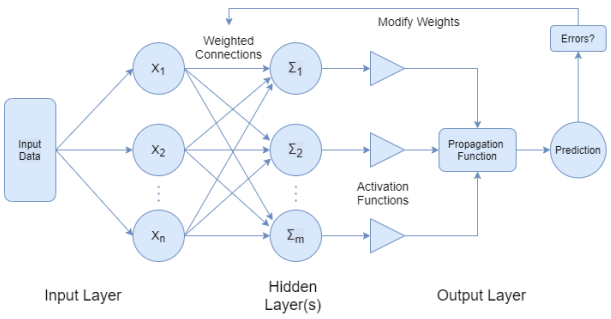


Figure 3: Neural network workflow

3.2.2 Support Vector Machine. Support Vector Machines (SVM) are commonly used for classification and regression problems. The main objective of an SVM is to find the optimal hyperplane for the classification of two classes. An easy way to explain how an SVM works is by using a 2D space representation of two different classes linearly separable. The optimal hyperplane is the line passing between these two classes, creating a maximum margin that serves as the host of the support vectors. Unfortunately, not all datasets can be linearly separable; in a multidimensional space, Kernel functions are used to shape the hyperplane. To find the optimal hyperplane in a multidimensional space, a Support Vector Machine(SVM) needs the following components:

- **Feature Selection**: helps create new features so that the Kernel function can transform and find boundaries in the dataset.
- **SVM Kernel**: functions used to shape the hyperplane in a multidimensional space.
- **Classifier (SVM)**: finds the optimal hyperplane and the support vectors that define the maximum margin.

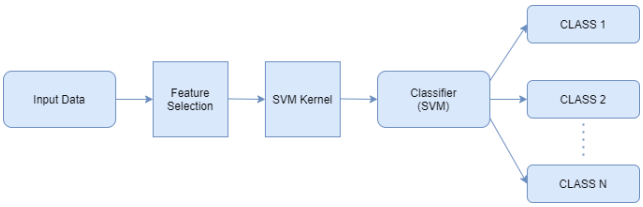


Figure 4: Support Vector Machine

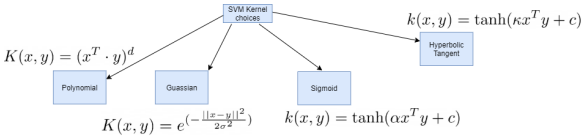


Figure 5: SVM Kernels

3.2.3 SVM Kernel. For the SVM algorithm we choose a kernel, a map from often single dimensional space to n-dimensional space. This is done in the hopes that a higher dimensional space will allow the data to be classified more efficiently. And as previously mentioned in the section above, being able to find a plane that will linearly separate the data is an extremely integral part of the SVM model. For our data and design we found that the following 4 kernels are potential candidates: polynomial, Gaussian, Sigmoid, and Hyperbolic Tangent. In figure 4 above we can see that for each given kernel, we have an associated function. And interestingly, choosing any such kernel will allow us to make predictions based only on our original feature selections.

We found that the polynomial kernel is often associated with having high measure of accuracy in the realm of of image processing. The Gaussian kernel is looked at as being more general purpose. That is to say that it primarily operates without being given prior

knowledge of input data. Moreover, the remaining two kernels, Sigmoid and Hyerbolic Tangent, form an excellent pair with neural networks. What we hope to accomplish by selecting a kernel model, is to choose one such model that will help us deliver the most accurate and reliable results.

3.2.4 Hyper-parameters and tuning. It would be to our detriment to run the SVM model out of the box and disregard its configuration. Moreover, in the last section we mentioned the notion of kernels, but we left out other key aspects. Those key ideas are hyper-parameters and the process of tuning them. Hyper-parameters are strictly kernel parameters that focus on either the soft margin cost or the cost of miss-classification, C, the curvature or influence of a single training example extends, γ, and the degree, poly. By default Scikit-Learn (Sklearn) uses the values of: C = 1, γ = scale, and poly = 3 for each appropriate kernel. In our case we mainly want to look at the values of C and γ. Further, for each of these parameters we would like to find the best combination such that our model has high accuracy while also maintaining an optimal variance and bias. As such, after finding an acceptable kernel, we then need to perform some kind of parameter tuning. One of the easiest yet brute force ways to do this is via a technique known as a grid search. In a grid search we take a grid or list of parameters and test them with a given kernel. The combination that provides the best results will ultimately be returned as the optimal parameter set. And so with this in mind we look to find the best combinations of parameters for our chosen kernel.

3.2.5 K-Nearest Neighbors. K-Nearest Neighbors is known as one of the simplest non-parametric classifiers, and a popular algorithm for binary classification in high dimensional problems[11]. K is the number of nearest neighbors and is the core deciding factor. K is generally an odd number if the number of classes is two. To find the closest similar points, the key is finding the distance between points using distance measures such as Euclidean distance, Hamming distance, Manhattan distance and Minkowski distance. Fig. 6 shows how the algorithm works [3]. In this project, KNN is being used from Scikit-learn (default setting) as a starting point, and will find an optimal k value based on the size and other factors of the data.

3.2.6 Logistic Regression. Logistic Regression is another simple statistical method that is commonly used in machine learning for binary classification. Its basic fundamental concepts are also constructive in deep learning. It uses a log of odds as the dependent variable. Logistic Regression predicts the probability of occurrence of a binary event utilizing a logit function.

Properties of Logistic Regression:

- The dependent variable in logistic regression follows Bernoulli Distribution.
- Estimation is done through maximum likelihood.
- No R Square, Model fitness is calculated through Concordance, KS-Statistics.

Refering to Fig. 7 the sigmoid function, also called logistic function gives an 'S' shaped curve that can take any real-valued number and map it into a value between 0 and 1. If the curve goes to positive infinity, y predicted will become 1, and if the curve goes to negative infinity, y predicted will become 0. If the output of the sigmoid function is more than 0.5, we can classify the outcome as 1 or YES,

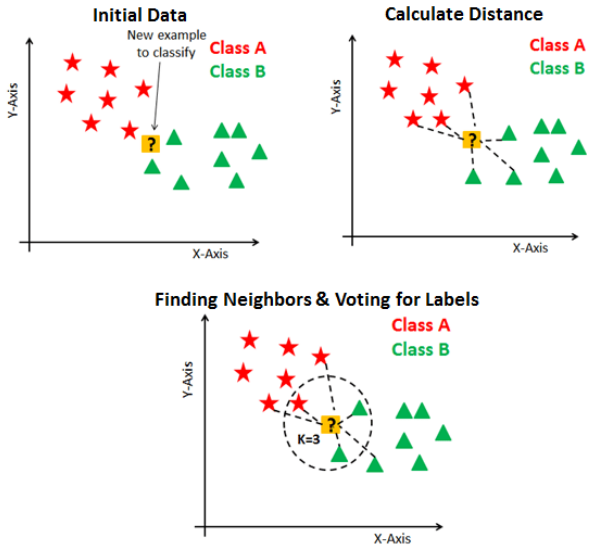


Figure 6: KNN Classification

and if it is less than 0.5, we can classify it as 0 or NO [4]. In this project, LR is being used from Scikit-learn (default setting) as a starting point, and will apply hyper-parameter tuning to improve the model.

$$f(x) = 1/1 + e^{-x}$$

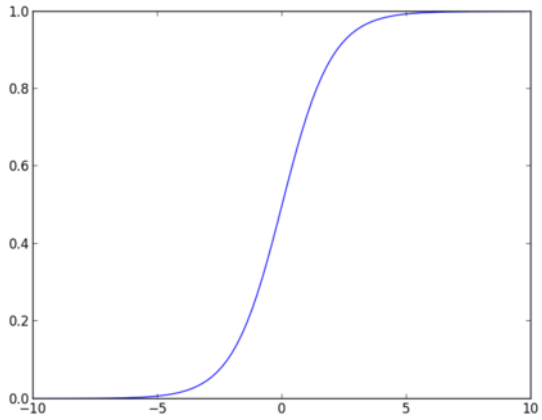


Figure 7: Logistic Function

3.2.7 Gaussian Naive Bayes. Naive Bayes is a statistical classification technique based on Bayes Theorem. Naive Bayes classifier is the fast, accurate and reliable algorithm. Naive Bayes classifiers have high accuracy and speed on large datasets. Gaussian Naive Bayes (GaussianNB) implements the Gaussian Naive Bayes algorithm for classification. The likelihood of the features is assumed

to be Gaussian [2]:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

In this project, GaussianNB is being used from Scikit-learn (default setting) as a starting point, and will be changing parameters to improve the performance of the model.

3.3 Quality Measurement

The Quality Measurement phase is the last phase in the design and the work flow can be observed in Fig. 8. In this phase, we will be collecting the prediction results from our trained models, applying evaluation methods, comparing and finalizing our results. The result can be measured in various ways such as classification metrics, K-fold cross-validation, confusion matrix, Receiver Operating Characteristic (ROC) Curves, and Area Under the Curves (AUC).

Basic Terminology:

- **Accuracy:** Overall, how often is the classifier correct?
- **Precision:** When a positive value is predicted, how often is the prediction correct?
- **Recall/Sensitivity:** When the actual value is positive, how often is the prediction correct?
- **F-Measure/F1 score:** Calculates the harmonic mean of the precision and recall (harmonic mean because the precision and recall are rates).
- **True Positive:** Correctly predicted that there is A Fire
- **True Negative:** Correctly predicted that there is NO Fire
- **False Positive:** Incorrectly predicted that there is A Fire (a "Type I error" - falsely predict positive)
- **False Negative:** Incorrectly predicted that there is NO Fire (a Type II error" - falsely predict negative)

The statistical models calculate the average distance of error between the predicted and the actual outcome. After the accuracy has been calculated, we will compare the results.

The accuracy of each model be compared to each other as well as the scores from previous works to see if adjustments are needed. If adjustments should be made, we will go back into the Data Preparation Phase or more specifically, the Data Preprocessing Phase, as shown in Figure 8, to make adjustments to parameters. The adjustments will be making are either to change the weight of the parameters or to replace them. After we have reached the desired results, we will finalize them by writing our report on our process and findings.

3.3.1 Experiments. Multiple datasets were created to analyze the California region, one at the state level, one at the county level, and another at the county level that focuses on the fire season of the collected years. The state level dataset treats the entirety of CA as one region for the purposes of averaging input conditions and interpreting the fire incident record into the target class of fire/no_fire. The county level dataset consists of a subset of CA, specifically 44 out of the 58 counties that had reported forest fires in the years 2019-2020, according to the Cal Fire incident record. This second dataset also treats each of the 44 counties as an individual region using it's county ID, which allows for more localized classification of the input data and the target class (fire/no_fire)

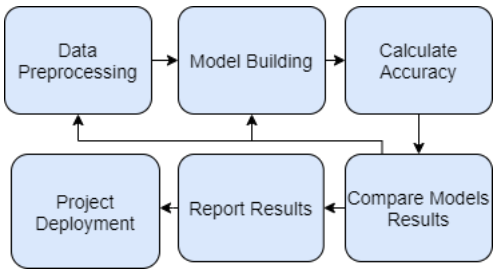


Figure 8: Quality Measurement Design which shows the work flow for this phase

from the fire incident record, which specifies regions on fire down to the county level. The last experiment selects a subset of the previous county level dataset, specifically the months of May through November, where the majority of fires were reported according to the fire incident record. All three datasets will be tested with all the chosen models and have various techniques applied to them such as normalisation, dropping of outliers, and resampling of the data. All datasets were then validated by comparing forecasting results from the input features to the fire incident record from Cal Fire. The full subset of CA counties used in the last two datasets are: Siskiyou, Merced, San Joaquin, Alameda, Contra Costa, Shasta, San Luis Obispo, Monterey, Riverside, San Diego, Ventura, Stanislaus, Santa Clara, Madera, Calaveras, El Dorado, Sacramento, San Bernardino, Lake, Tuolumne, Mendocino, Solano, Colusa, San Benito, Santa Cruz, Tehama, Butte, Amador, Napa, Fresno, Sonoma, San Mateo, Neveda, Placer, Mariposa, Tulare, Lassen, Kern, Los Angeles, Glenn, Yolo, Plumas, Yuba, Orange.

4 IMPLEMENTATION

Implementation section will include steps in data pre-processing, parameters used in models and evaluation methods used.

4.1 Data Preprocessing

Since multiple similar datasets were collected, the following steps will describe the process of accessing and processing the state level dataset with mentions of the county specific steps when they differ. A general note for the preprocessing sections: the county level data uses the same sattelite images used in the state level processing, except there is an extra step of separating the CA images into county images using geojson geometries for a masking technique described in more detail in the data clipping step.

4.1.1 Data Collection. To prepare the dataset for this project, the first step is accessing the Remote Sensing Data from the Land Processes Distributed Active Archive Center (LP DAAC) and California fire incident record from Cal Fire, as referenced in Fig. 9. For the LP DAAC data, the data extract tool AppEEAR [1] was used. Once inside the AppEEAR tool, select Extract,area sample, and then start new request. On the extract page, select the boundaries of California via shape file or geojson, adjust the time frame to be between 1/1/2019 to 12/31/2020, select the target layer, select GeoTiff file format, and select geographic projection. The target layers used were:

[1] https://lpdaac.usgs.gov/tools/appeears/

MOD13A1_006_500m_16_days_NDVI, MOD11A1_006_LST_Day_1km, VPN14A1_001_FireMask, and MCD64A1_006_Burn_Date for NDVI, LST, TA, and burn area, respectively. One can find more details of all of the products on a GitHub repository set up for the preprocessing steps that includes all files used for processing the datasets on state and county levels [2]. Next, download CA fire incident data from the Cal Fire website [3].

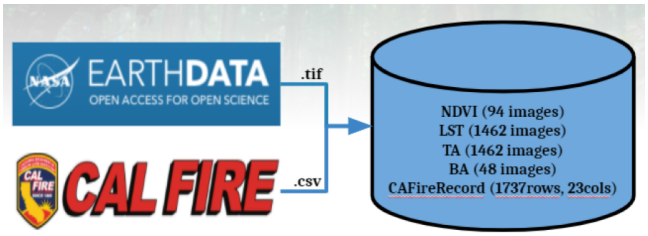


Figure 9: Data Sources

4.1.2 Data Cleaning. Data cleaning consists of correcting imperfections that result from satellite imaging equipment caused by various occurrences such as cloud cover, satellite instrument malfunctions, or by the sources that contain and distribute the images. MODIS data is already processed with respect to geo-referencing, so the focus of this section is to weather related imperfections. The data was cleaned by masking the raw images by the corresponding quality lookup table (LUT) that was acquired from the LP DAAC website. LP DAAC has great documentation regarding masking, visualizing, and plotting appears output [4]. In essence, all of the low-quality data or data with meteorological interference are dropped from the images using the masking technique. After the data is cleaned, the values are averaged across the specific region, either state or county, and stored with the corresponding date. The following subsection includes the steps to follow for cleaning and masking a GeoTIFF file.

[2]https://github.com/dmw01/Beat-the-Heat-Data-Preprocessing
[3]https://www.fire.ca.gov/incidents/
[4]https://lpdaac.usgs.gov/resources/e-learning/masking-visualizing-and-plotting-appeears-output-geotiff-time-series-python/

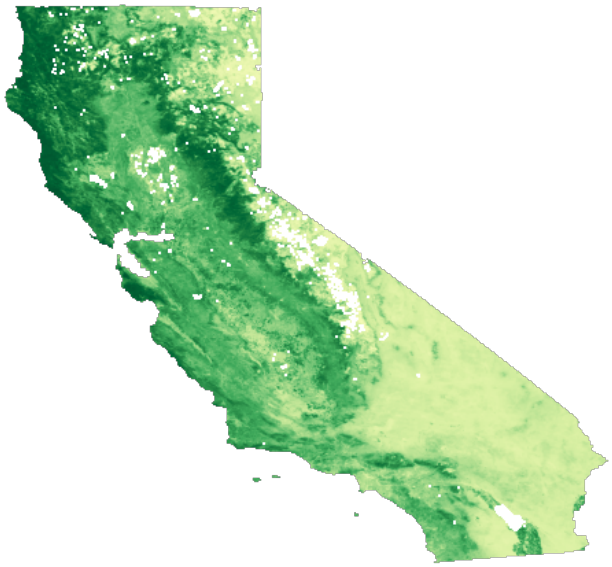


Figure 10: California NDVI Data Visualization

4.1.3 Data Clipping. Data clipping or sub-setting of the images was done to extract only the parts of the GeoTIFF files that overlapped the burn area dataset. The different Libraries used in this step include:

- **P**andas Library to open and export CSV files.
- **G**eoPandas Library to read the GoeJson California county shapefile.
- **G**eospatial Data Abstraction Library (GDAL) to open and read the layers of the GeoTIFF files.
- **R**asterio Library to mask a GeoTIFF file by the shapefile
- **N**umPy Library to mask arrays.

The steps to mask the NDVI GeoTIFF file by its Quality data are the following:

- **O**pen the NDVI tif file to be analyzed.
- **O**pen the Quality tif file associated with the product.
- **O**pen the lookup table containing the quality values of the product.
- **R**ead the raster band of the product as an array.
- **S**cale the product by its scale factor.
- **O**btain an array of good quality values by masking the scaled product by its quality data.

The exact process is applied for the rest of the products: LST, TA, and BA. At the county level, first, the Geotiff file is masked by the shapefile of the county selected. Then, the same process of masking by the quality data is applied.

4.1.4 Data Interpolation. Next step is to interpolate the NDVI, LST, and TA datasets so they have the same temporal frequency. This step is only necessary for the NDVI dataset because it is in a 16 day temporality and the other two are daily. Linear interpolation from the Pandas library was used for the state level data.

4.1.5 Target Class from Fire Incident Record. Now the fire record is to be processed and used to create the target class column of

fire/no_fire. The Cal Fire .csv dataset contains a number of columns of interest: type of fire, start and end dates, and the county of the fire. First, the years of interest are isolated to only include 2019-2020, followed by transforming start and end dates into a list of all the days in between. For the dataset that is categorized at the county level, there is an extra step of separating fires that had multiple counties into separate counties, as well as keeping track of the county ID during the previous process. The dataset that looks at the state as a whole unit, labels any fire in the fire record as a fire for all of California. After creating this new dataframe of the fire record with the columns of: date (daily), fire/no, and county ID if appliciable, the fire record dataframes data's dates are then merged with one of the datasets (NDVI, LST, or TA), specifically looking at the date column to create a new column of fire/no_fire. If counties are involved, an extra condion is used to see if a given county is on fire with respect to the day in question. The remaining two datasets are then merged with whichever was used first, followed by removal of the date and county ID (if applicable) to complete the dataset with the following columns: NDVI, LST, TA, fire/no_fire, and county ID, if applicable. An example of our final state level dataset can be seen in Fig. 11.

	NDVI	LST	TA	Class
0	0.241233	280.170902	5.0	No_fire
1	0.242536	281.678667	5.0	No_fire
2	0.243840	285.865983	5.0	No_fire
3	0.245143	283.249994	5.0	No_fire
4	0.246446	275.546052	4.0	No_fire

Figure 11: Final Dataset Example

4.2 Applying Models and Evaluation

All models used train/test split as 80 percents for training and 20 percent for testing. Evaluate models using scikit-learn classification metrics that return (precision, recall, f1-score, support, and accuracy score), and confusion matrix.

- NN: The neural network model is implemented using the MLPClassifier in the Scikit-Learn (Sklearn) library. MLPClassifier is a multi-layered perceptron, which is another term for a neural network with multiple hidden layers. The network was trained and tested using varying hidden layer and neuron amounts but performed the best with 12 hidden layers and 12 neurons per layer. Model parameters: (solver= 'lbfgs', alpha= 0, activation= 'relu', hidden_layer_sizes= 12, random_state= 1). Sklearn's KFold method was used during model training and testing, with an average taken out of 7 folds.
- SVM: As mentioned prior, the Support Vector Machine (SVM) model is highly dependent on a whole suite of configurations. We namely, needed to find what kernel configuration and hyper-parameters would yield us the best results. After testing and running several grid searchers, we found the following to give us the best results: (kernel = 'rbf', C = 9.9, gamma = 1). All other model parameters were left as default. For further clarification of the SVM's parameters one may visit sklearn.svm.SVM
- KNN: Parameters used (n_neighbors= 8, weight= 'uniform', algorithm= 'auto', leaf_size= 30, p= 3, metric= 'minkowski', metric_params= None, n_jobs= None). Further explanation on each parameter can be found at sklearn.neighbors. KNeighborsClassifier.
- GaussainNB: Parameters used(priors= None, var_smoothing= 1e-09). Further explanation on each parameter can be found at sklearn.naive_bayes.GaussianNB.
- LR: Parameters used (penalty= 'l2', dual= False, tol= 1e-4,C= 1, fit_intercept= True, intercept_scaling= 1, class_weight= None, random_state= None, solver= 'lbfgs', max_iter= 100, multi_class= 'auto', verbose= 0, warm_start= False, n_jobs= None, l1_ratio= None). Further explanation on each parameter can be found at sklearn.linear_model.LogisticRegression.

5 RESULTS

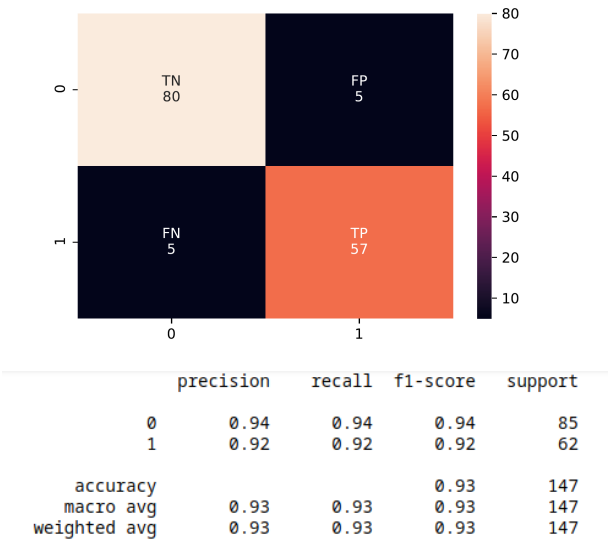
Fig. 12 shows the result of all models applied on the data across CA. Which is mean the data include all parts that might not have fire at the same time. This dataset is content 731 rows and 4 columns including the 292 Fires and 431 No_Fires.The size of the dataset is small for NN so the result for NN is not as high as the other models.

Accuracy Score	
Neural Networks	81%
Support Vector Machine	93%
K – Nearest Neighbors	87%
Gaussian Naïve Bayes	84%
Logistic Regression	85%

Figure 12: Result on data across CA

Fig. 13 below depicts the results of our Support Vector Machine model on our custom CA dataset. Here we can see that we our values for: true negative, false positive, false negative, and true positive. In addition, in the classification repository the values for precision, recall, f1-score, and support are displayed.

Fig. 14 shows the result of all models applied on the data of 44 counties CA that had fires in the past two years. Even though this result is very high accuracy score but it might be too biased due to the imbalance class in the data. There were 1370 Fires and 41028 No_Fires. The ratio is approximately 1:30 and is considered a severe imbalance. This imbalance could lead the model to predict on only No_Fire most of the time.



	precision	recall	f1-score	support
0	0.94	0.94	0.94	85
1	0.92	0.92	0.92	62
accuracy			0.93	147
macro avg	0.93	0.93	0.93	147
weighted avg	0.93	0.93	0.93	147

Figure 13: Confusion matrix and classification report for our SVM model

Accuracy Score	
Neural Networks	97%
Support Vector Machine	98%
K – Nearest Neighbors	97%
Gaussian Naïve Bayes	94%
Logistic Regression	97%

Figure 14: Result on data of 44 counties in CA

6 CONCLUSION AND FUTURE WORK

The work outlined acts as a small step in the direction of combating a force as unpredictable and devastating as forest fires. This paper is not the first with the goal of fighting forest fires from behind the front lines, and surely far from the last, however the approaches taken provide promising results for the field of machine learning and remote sensing to tackle very ambiguous problems that have long caused ruin for eons. The work highlights the use case of forecasting areas that a fire may occur in, given highly influential, and widely available factors: NDVI, LST, and TA combined with a fire incident record. The models used give highly accurate predictions of regions that are likey to be on fire with the most notable being 87% for KNN and 93% for SVM for the state level dataset, as well as 98% for SVM and 97% for NN, GNB, and LG models for the county level dataset.

There were a few directions that we did not have time to explore but may yield promising results.

One step that could increase the accuracy of our fire indicator would be to use burn area as a fire indicator instead of the fire incident record. While burn area doesn't tell which areas were burned with 100% accuracy, it has very fine spatial and temporal resolution down to the pixel and daily levels, respectively. The fire record on the other hand does not tell the location of the fire, only the county, acres burned, and dates the fire is active. This introduces uncertainty because the finest spatial resolution of fires than can be extracted from this, is the county. Burn area on the other hand labels fires down to the daily temporally and 500m grid pixels. This can be used similar to how burn area was used to extract all days that had fire conditions for a given month, except this masking process would extract an area sample for all the areas that had fires for a given day, and flag that data's target class as fire. To do this with daily temporality, the monthly burn area data would need to be masked into daily images, using the 'Burn Date' values which represent calendar days (1-366).

Once we achieve the accuracy result desired, the next step will be implementing the project to real time data. This can be done by using NASA API to get the most recent data from the satellite images. Then extracting the values of the satellite images and preparing the data to run the model with. Running the model with the most recent data will result in a prediction if there is going to be a fire or not. Implementing reinforcement learning will make sure that the data accuracy is kept high, by looking at the fire record data and seeing if there was a fire as predicted by the model. In case of a false prediction, a method will change the parameters of the model and retrain with the most recent 2 years of data.

ACKNOWLEDGMENTS

We would like to thank our advisor Dr. Xunfei Jiang and California State University of Northridge for the resources and assistance provided throughout the duration of this research.

REFERENCES

[1] 2020 (accessed October 1, 2020). *2020 Incident Archive.* https://www.fire.ca.gov/incidents/2020/
[2] 2021 (accessed April 27, 2021). *In Depth: Naive Bayes Classification.* https://jakevdp.github.io/PythonDataScienceHandbook/05.05-naive-bayes.html
[3] 2021 (accessed April 27, 2021). *KNN Classification using Scikit-learn.* https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn
[4] 2021 (accessed April 27, 2021). *Understanding Logistic Regression in Python.* https://www.datacamp.com/community/tutorials/understanding-logistic-regression-python
[5] J. Amrutha and A. S. Remya Ajai. 2018. Performance analysis of backpropagation algorithm of artificial neural networks in verilog. *2018 3rd IEEE International Conference on Recent Trends in Electronics, Information and Communication Technology, RTEICT 2018 - Proceedings* (2018), 1547–1550. https://doi.org/10.1109/RTEICT42901.2018.9012614
[6] Shruti Lall and Bonolo Mathibela. 2017. The application of artificial neural networks for wildfire risk prediction. *International Conference on Robotics and Automation for Humanitarian Applications, RAHA 2016 - Conference Proceedings* (2017). https://doi.org/10.1109/RAHA.2016.7931880
[7] Hao Liang, Meng Zhang, and Hailan Wang. 2019. A Neural Network Model for Wildfire Scale Prediction Using Meteorological Factors. *IEEE Access* 7 (2019), 176746–176755. https://doi.org/10.1109/ACCESS.2019.2957837
[8] By Hariharan Naganathan, Sudarshan P Seshasayee, Jonghoon Kim, Wai K Chong, and Jui-sheng Chou. 2016. Wildfire Predictions : Determining Reliable. 16, 4 (2016).
[9] Rylan Perumal and Terence L.Van Zyl. 2020. Comparison of Recurrent Neural Network Architectures for Wildfire Spread Modelling. *2020 International SAUPEC/RobMech/PRASA Conference, SAUPEC/RobMech/PRASA 2020* 2020-Janua (2020). https://doi.org/10.1109/SAUPEC/RobMech/PRASA48453.2020.9078028

[10] David Radke, Anna Hessler, and Dan Ellsworth. 2019. Firecast: Leveraging deep learning to predict wildfire spread. *IJCAI International Joint Conference on Artificial Intelligence* 2019-Augus (2019), 4575–4581. https://doi.org/10.24963/ijcai.2019/636

[11] Hadi Raeisi Shahraki, Saeedeh Pourahmad, and Najaf Zare. 2017. K Important Neighbors: A Novel Approach to Binary Classification in High Dimensional Data. *BioMed Research International* 2017 (2017). https://doi.org/10.1155/2017/7560807

[12] Younes Oulad Sayad, Hajar Mousannif, and Hassan Al Moatassime. 2019. Predictive modeling of wildfires: A new dataset and machine learning approach. *Fire Safety Journal* 104, September 2018 (mar 2019), 130–146. https://doi.org/10.1016/j.firesaf.2019.01.006

[13] Jeremy Storer and Robert Green. 2016. PSO trained Neural Networks for predicting forest fire size: A comparison of implementation and performance. *Proceedings of the International Joint Conference on Neural Networks* 2016-Octob (2016), 676–683. https://doi.org/10.1109/IJCNN.2016.7727265

[14] Dieu Tien Bui, Quang Thanh Bui, Quoc Phi Nguyen, Biswajeet Pradhan, Haleh Nampak, and Phan Trong Trinh. 2017. A hybrid artificial intelligence approach using GIS-based neural-fuzzy inference system and particle swarm optimization for forest fire susceptibility modeling at a tropical area. *Agricultural and Forest Meteorology* 233 (feb 2017), 32–44. https://doi.org/10.1016/j.agrformet.2016.11.002

[15] Guoli Zhang, Ming Wang, and Kai Liu. 2019. Forest Fire Susceptibility Modeling Using a Convolutional Neural Network for Yunnan Province of China. *International Journal of Disaster Risk Science* 10, 3 (2019), 386–403. https://doi.org/10.1007/s13753-019-00233-1