Wildfire prediction using Machine Learning

Andrew Bennett
Stevens Institute of Technology
abennett1@stevens.edu

Rahul Gupta
Stevens Institute of Technology
rgupta24@stevens.edu

Chinmay Bhagwat
Stevens Institute of Technology
cbhagwat@stevens.edu

***Abstract*—Machine Learning algorithms have been becoming popular in many research areas due to the high precision for end-to-end prediction. In the wildfire area, there are many types of research that have applied different methods of Machine Learning in predicting fire activities. In this paper, we conducted the research based on the most up-to-date data from 2013 to 2022. The algorithms used in the research include multi-layer perceptron (MLP), logistic regression, decision trees, and random forest.**

I. Introduction

Wildfires has become a big issue in the United States, especially in the states like California, causing casualties and substantial economic losses every year. In the 2022 wildfire season, as of 21 September, a total of 6,473 fires have been recorded, totaling approximately 365,140 acres (147,770 hectares) across the state and killing 9 people so far in California. Predicting such an environmental issue becomes a critical concern to mitigate this threat. Several factors cause wildfires such as climate change and human activities. In this project, we are going to predict wildfire using machine learning models based on different types of weather and geographic data.

II. Related Work

One of the most recent research [1] used an ensemble model to analyze satellite data, weather data, and historical fire data and achieved 100% precision on wildfire risk prediction and 93% on wildfire risk detection. In the paper [2] the fire prediction models utilized were Stochastic Gradient Descent, Decision Tree, Logistic Regression, and Boosted Trees leveraging climate feature data and geo-referenced fire locations to build the predictive models. [3] paper suggested based on the trajectories of fire embers had a significant impact to the spreading of wild fires, looking at a few different models of how the embers and temperature can increase the growth of a wild fire or trigger new wild fires. Another interesting observation, was the when the surface temperature was significantly higher the fire growth was limited and propagated less distance. Based on these different factors, our feature data will include Wind Speed and Soil Temperature to make better predictions on the locations that fire will occur. Various different companies and researchers have created forest fire simulators that use Rothermel's fire spread equation, looking at this equation and the GeoJson data for the fire this can be rewritten to work with our fire data:

$$R = \frac{(I_P)_o(1 + \phi_w + \phi_s)}{\rho_b \epsilon Q_{ig}} \tag{1a}$$

The R represents the rate of spread, in our case we have a fire start point and maximum spread of a fire is known. The equation can be rewritten as the following Rate Equation 2h measuring meters per day to generate the points which correlate to the spread of the fire, calculating the distance in meters propagated using the Haversine Equation 2g, the 1984WGS Geodesic Radius 2b along with the time delta 2a:

$$\Delta T = t_{ContainmentDate} - t_{AlarmDate} \tag{2a}$$

$$radius = 6356752.3142 \tag{2b}$$

$$\Delta\phi = (\phi_{end} - \phi_{start}) \tag{2c}$$

$$\Delta\lambda = (\lambda end - \lambda start) \tag{2d}$$

$$a = (\sin\frac{\Delta\phi}{2})^2 + \cos\phi_{start} * \cos\phi_{end} * (\sin\frac{\Delta\lambda}{2})^2 \tag{2e}$$

$$c = \arctan\frac{\sqrt{a}}{1-a} \tag{2f}$$

$$d = radius * c \tag{2g}$$

$$R = \frac{d}{\Delta T} \tag{2h}$$

This will allow data points to be generated for each day of the wild fire from the alarm date to the containment. This rate equation could also be improved if the wind direction was taken into account to move along a vector relative to the wind direction each day. In the paper [4], the Rate Equation 1a is used to generate a Fire Simulation but in addition to the wind direction, the terrain slope is also taken into account. In order to build a fire simulation, lots of computational resources were utilized in addition to preprocessing the data. Ideally, all of the fire data that we obtained for processing would be stored on the GPU representing the terrain data, wind vectors, Latitudes and Longitudes of the fire points that make up the polygons' boundaries as the fire spreads. Due to limitations of the hardware being utilized the data processing of the fire locations and fire predictions will be limited to the CPU.

III. Feature Data

A. Description of Dataset

The data for training will be focused on California Wildfires from 2013 to 2022. The main source of the training data will be from [5] which will include the following features: Year of occurrence, Fire Name, Alarm Date, Containment Date, Spatial Shape of the fire, Spatial Shape Length, and the Total Acres burned. The data from [5] also includes the shape data that makes up the bounds of each individual fire. In addition, to this data source, the following source [6][7] will supplement the previous data set with GeoSpatial information about the fire location and size of the fire, allowing the Latitude and Longitude to be plotted

from this data. The GeoSpatial data will be in GBD format for the GIS data. Climate features will also be added to the training data set from the following [8][9] to add the features: Date, Reference ETO, Solar Radiation Average, Average Vapor Pressure, Min Air Temperature, Max Air Temperature, Average Air Temperature, Maximum Relative Humidity. Minimum Relative Humidity, Average Relative Humidity, Dew Point, Average Wind Speed, Wind Run, and Average Soil Temperature using the mean daily values. The climate data will be in two different formats netCDF4 and delimited text files.

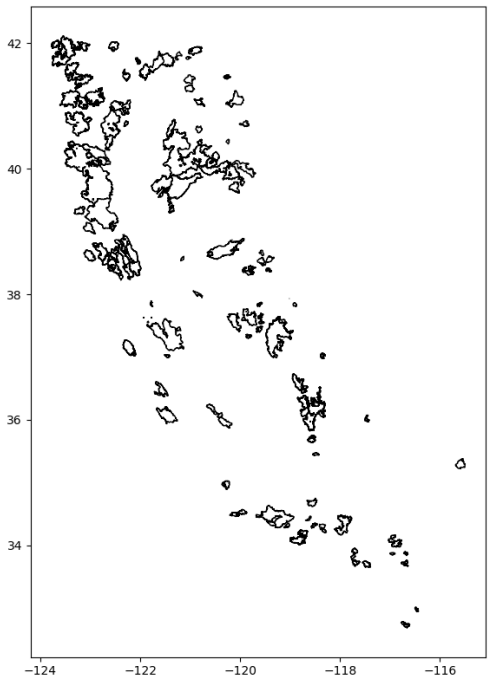

Fig. 1: Spatial Data for Fires

The first step of using the training data will be to compile the data into aligned data sets. Secondly, cross reference the data sources and use GIS data to ensure that each fire has location data and spread data. If the Longitude and Latitude cannot be found for a particular fire entry, this would be a candidate for removal from the training data set. Next, obtain the climate and weather data for each of the fires within the main data set using monthly intervals. Daily info for climate and weather info may be used as well when looking at the spread and size of fire from the alarm date to the containment data.

B. Data Cleaning

In processing the data different anomalies or issues with the data came up and had to be addressed. In the weather data certain dates were missing data completely so the values were set zero for the different weather features. Sometimes months of data was missing from the data set as well. The weather data was compiled for each day for the years 2013 to 2022. Another issue that was faced with the weather data is that some values for a given feature were missing, in these cases the feature value was set to zero. In certain cases when no data was available the mean of the feature was determined and substituted for the missing data.

C. Data Grid

California has a maximum Longitude of -114.1315 decimal degrees and minimum Longitude of -124.6509 decimal degrees. The maximum Latitude of 42.0126 decimal degrees and minimum Latitude of 32.5121 decimal degrees. The weather stations used for California had the following Latitude ranges of (32.411, 42.003) and Longitude ranges (-124.243186, -114.558). Initially, California was broken up into a 10 by 10 one degree grid but has to be recreated due to the unique shape of California. 68 one degree zones were created that would split California into spatial zones, then each of the 121 weather stations were mapped accordingly to each of the zones. In the case of a zone containing multiple weather stations, the weather feature data was averaged taking each station into account. For the other case if a weather station was not in a zone then the closest weather station was used to supplement the weather data for a zone.

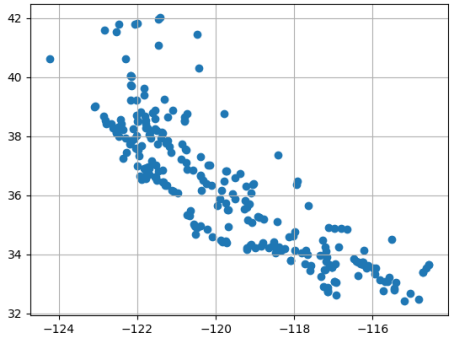

Fig. 2: Weather Stations on Grids

Originally, fire data mapped to weather stations only contained 242,000 rows of data using the centroid approach for spatially locating the data. Later on each fire location centroid and outer polygon coordinates were used to map each fire to a date and grid, next the weather feature data associated with the fire on the alarm date was mapped creating roughly 3.8 million data rows to use for the machine learning model.

IV. Fire Prediction Models

1) Multi-Layer Perceptron: The Multi-Layer Perceptron was chosen due to the binary nature of the output that was being predicted whether there was a fire at a location within the geographic bounds of California. This algorithm was also chosen for the ability to handle nonlinear feature data, due to the fire not always moving linearly as the wind direction and changes while spreading. The weather feature data will not always be linear due to rapid fluctuations in from night to day, in addition to the seasons and coastal fronts of weather. The MLP is composed of an input layer, hidden layers, and

output layer. Using the MLP different activation functions were available for training the model within the hidden layers: Sigmoid, Relu, and Hyperbolic Tangent. The data was split originally into a 60:40 split for training and test but produced only around 50 percent for predicting fires. Therefore, the data was split into the traditional 80:20 and 85:15. The 80:20 split ranged around 66 percent without tuning and the 85:15 was slightly lower, based on the difference the 80:20 split was finalized. So 80 percent of the 3.8 million data points would be used training and 20 percent for testing the MLP model. The data for all features were normalized between 0 and 1 by using using the library [10] which takes each feature value subtracting the mean and dividing by the standard deviation, ensuring features would be equally valued when training model and not to have extreme outliers/unit mismatches.

2) Decision Trees and Logistic Regression: Logistic regression is a classic model for classification. As we have only "has_fire" and "no_fire" class (binary classification), we would like to try logistic regression because it is one of the easiest model to implement and train.

Decision tree is also efficient and requires less effort in preprocessing data. One of the disadvantages of this method is that it tends to be over-fitting.

3) Random Forest Classification: Random Forest is an ensemble classifier, that works on the basis of a decision tree algorithm. Random Forest constructs a group of independent and non-identical decision trees based on the idea of randomization. Each tree in the Random Forest gives a class prediction and the class with the most votes becomes our model prediction. To put it simply Random Forest is creating a group of weak models that are combined to create a strong model. To create random trees for the forest, we need slightly different training data for each tree. This can be done by taking the Bootstrap sample method. Bootstrapping is a resampling method that is used to create a diverse set of training data for each decision tree. The decision tree is then trained on the bootstrapped sample. This process is repeated for each decision tree in the random forest, resulting in a diverse set of decision trees that are trained on different subsets of the original training data. However, this is not enough randomness yet. The other obvious place where it is possible to add randomness is to limit the choices that the decision tree can make. At each node, a random subset of the features is given to the tree, and it can only pick from that subset rather than from the whole set Random Forest classifier tends to outperform most other classification methods in terms of accuracy without issues of overfitting.

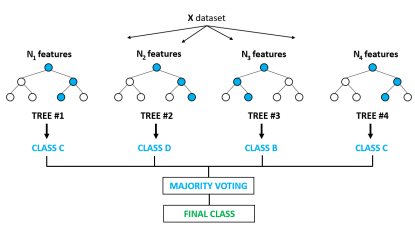

Fig. 3: Illustration of random forest trees

V. Implementation Details

1) Multi-Layer Perceptron: Three different MLP Classifiers were chosen each with one of the activation functions above to ensure different strategies in analyzing the data and converging. The hidden layers were adjusted several times for each MLP from a hundred layers to eight hundred layers. The MLP using Sigmoid activation function settled around 125 hidden layers. The MLP using the Relu activation function settled around 150 hidden layers. The MLP using the Hyperbolic Activation function settled around 100 hidden layers. A constant learning rate of 0.001 was chosen for each of the models. In addition, the models all use back propagation and momentum of 0.9. The momentum was varied from 0.5 to 0.9 to ensure that model was able to converge with the inertia built in the system while training. After matching all the fire data to the weather data, looking at the data there was a large discrepancy between the two binary outcomes of the data. Out of all the data from the fires only 0.05-3 percent of the data had fires, due to this difference after the training data and test data was split, the training data was upsampled. At first the training data was resampled using randomly chosen data points to increase the size of the majority class, but the distribution of the randomly generated data was not consider an accurate representation due to the nature of the fire's expansion when moving across terrain. Next, Synthetic Minority Over-sampling Algorithm (SMOTE) was considered as another alternative for upsampling the training data but did not take into account the local distribution of the data that had active fires. Finally, the Adaptive Synthetic Algorithm (ADASYN) from the Python processing library [11] was chosen as the method to upsample the training data due to considering the local distribution and using the Nearest Neighbors Algorithm in constructing the synthetic data. The number of fires after upsampling was approximately 50% of the training data balancing the number of sample between each binary output. Next challenge of building these models was hypertuning the parameters, two different algorithms were utilized to optimize the parameters within the MLP model. First algorithm was the Grid Search Algorithm from [10] which runs an exhaustive search of all parameters chosen to iterate over. Tuning the learning model parameters lead to an increase of fire predictions of 3 percent. The other algorithm used for tuning was the Randomized Search Algorithm which generates a value between a minimum and maximum value and tests the model's fit against the training data.

A Voting classifier was also used to compare each of the tuned MLP models to assist in predicting a fire by leveraging ensemble learning. The voting classifier uses the output from each of the MLP model to predict as a single model majority voting. The voting classifier performed better than any individual classifier a 2-3 percent. After training the data and running predictions the following performance metrics were used for to analyze each classifier: Precision, Recall, and F1-Score. In all four classifiers, the accuracy for prediction no fire was close to a hundred percent, around 99.5%. The accuracy of predicting no fire was not surprising due to the shear number of samples that hand no fire associated with them. The accuracy of predicting a fire was around 68 to 72 percent which increased for the original 66 percent without tuning different parameters.

The original data set had one location feature, one temporal

feature, and 13 weather features. After analyzing the significance of each of the features, the location of the fire was the most significant feature. The date of the alarm date, wind speed, and soil temperature features had a high significance in the machine learning model. The features with the lowest significance were culled from the data set, reducing the number of feature from 16 features to 9 features. After the feature reduction, the models were retrained on the modified set of data. Reducing the data's features with higher significance had a large impact on the accuracy of some of the models. At this point of training and processing a decision was made to change features that were utilized. Originally, the geographic region of California was broken into 68 one degree grids, each of these grids were split into four quadrants increasing the number of weather grids to 272. This splitting of grids was done to ensure a greater variety of weather data mapping to the different geographic regions and to reduce the averaging of weather station data, when a grid had multiple weather stations within the larger one degree grid. The features originally used grid as the location identifier stripping out the latitude and longitude data using only the grid identifier. Due to large significance of the grid id associated with a geographic region, the longitude and latitude were added back in as features which in turn increased the amount of locations with a fire significantly. The amount of geographic points on the earth with active fires now made up over 50 percent of the data without synthetically generating new data using the Adaptive Synthetic Algorithm. The large number of data points with fire led to over 5GB of data that only represented the following features: Alarm Date, Acres, Latitude, Longitude, and Containment date for the year of 2021 alone. The size of the fire data for years 2013-2022 was over 50GB which led to another problem, that the hardware running the machine learning model was no longer sufficient to hold this much data in RAM. In order to deal with the polygons' massive amount of edge data, random sampling was done of the boundary data points to ensure that spatial representation of the polygon was created, even though not every point was used for training/testing. The locations without fires were now synthetically generated by creating convex hulls in between convex hulls that represented fires, then iterating across the latitude and longitude limits of the no fire convex hull to generate synthetic points that represent non fire zones, in addition to choosing random points between fire zones. The data was stored in two separate files for no fire and fire zones, then run through the data preprocessing algorithm to map all weather features to each of the data points within both zone types. Utilizing [12] library these two data frames were combined into a single set of data to use for training, each data point was also rounded to the first decimal place and converted from float64 to float32 to reduce memory footprint. The final feature set utilized for training was day of the year, latitude, longitude, wind speed, soil temp, average humidity, eto, max humidity, and dew point. The data was still normalized using the mean and standard deviation to standard between 0 and 1. The accuracy for fire prediction increased to 97.47-99.97 showing the importance of good training data and data preprocessing, when dealing with location data having small geographic regions makes a huge impact to the predictability of the hypothesis. As the geospatial region increases the weather features and distinctive fire points were blended together collapsing the significant data points that could be used for training and testing, therefore drastically reducing the accuracy but increasing the performance of the model due to the reduced load on the system by averaging data points. There is a balance similar to the recall/accuracy tradeoff with choosing geographic regions, the smaller sections the geographic regions are broken into the more computing resources and amount of data points grows at a rate of geographic quadrants created and higher accuracy with predictions.

2) Decision Tree and Logistic Regression: All the *NaN* values in a column are replaced by the mean value of all the valid data in the same column type. The data is very unbalanced with the number of "no_fire" incidents is about 80 times more than the number of "has_fire" incidents. To reduce the imbalance, we over-sampled the number of "has_fire" incidents to match the frequency of "no_fire".

All 14 weather data types are used as input features for the model. The true classification is "0" for "no_fire" and "1" for "has_fire".

For *logistic regression*, "max_iter" is set to 1000 to avoid non-convergent error in the model.

For *decision tree*, "max_depth" is set to 10 to avoid over-fitting. Overall, decision tree model has higher precision and recall than logistic regression.

Model	Incident Type	Precision	Recall	F1 Score
Logistic	0	0.80	0.69	0.74
	1	0.73	0.82	0.77
Tree	0	0.93	0.79	0.85
	1	0.82	0.94	0.87

3) Random Forest Classification: The Random Forest model was built using Jupyter Notebook and wrote the code in Python3. At first, we replaced the null values in the weather dataset by taking the mean of the respective columns. Doing this process will help to prevent the loss in important data. As mentioned before, This data contains everyday weather reports with dependent data to check fire in that location. However, just like other real-world application data, this data is highly imbalanced. A classifier built using all data has a tendency to ignore minority classes. There are two common approaches to dealing with imbalanced data. The first is based on cost-sensitive learning and the second is based on the use of a sampling technique: either under-sampling the majority class or over-sampling the minority class. In our case we considered the sampling technique method to overcome imbalanced data.

For this Random Forest model, we used the under-sampling method to balance the data. There are a few ways to perform under-sampling. One approach is to randomly select a smaller number of observations from the majority class. Another approach is to use clustering techniques to group the majority of class observations into clusters, and then select a smaller number of observations from each cluster to keep in the dataset. We went with the random under-sampling method. To achieve that we used the RandomUnderSampler function from imblearn library. This function will Under-sample the majority classes by randomly picking samples with or without replacement. With this, we reduce the data from 356000 to 3789. Once this process is done, data were scaled using the StandardScaler function from sklearn library. Scaling the dataset will make it easy for a model to learn and understand the problem.

Before implementing the model, data was split as training

and testing data to prevent overfitting. The Random Forest classification was added to the program using the function from sklearn library. Taking nestimators as 500, specified the model to create 500 numbers of trees in the forest. After that classification, the model was fitted to the training data using the fit function. At last, a confusion matrix was built to get the precision, recall, and f1-score obtained from the model. By the end of the program, Random Forest Classifier is able to achieve 78.88 percent of accuracy to predict the wildfire using this weather data.

The Hyperparameter tuning is added to this program to maximizes Random Forest model's predictive accuracy. Hyperparameter tuning refers to the process of adjusting the hyperparameters of a machine learning model in order to improve its performance. Hyperparameters are the settings that control the behavior of the model, such as the learning rate and regularization strength. There are several approaches to hyperparameter tuning, including manual tuning, grid search, and random search which can be implemented to this machine learning model. We found out that grid search will gives the optimal result and maximizes the accuracy compare to other models for this application. Grid search involves specifying a grid of hyperparameter values and evaluating the model performance for each combination. This can be computationally expensive, but it is guaranteed to find the optimal combination of hyperparameters if the grid is fine enough. The result got remarkably improved after adding hyperparametr tuning to the model.

Model	Incident Type	Precision	Recall	F1 Score
Random Forest	0	0.78	0.78	0.78
	1	0.78	0.79	0.78
After Hypertuning	0	0.92	0.87	0.89
	1	0.88	0.92	0.90

VI. Comparison

The Multi-Layer Perceptron performed well in the no fire predictions due to the shear number of dates without fires, in the predictions for fire the model was only able to achieve around 72 percent accuracy after tuning the amount of layers within the classifiers and varying the momentum, learning rate, and different activation functions used within the layers of the neural network. The large limitation to increasing the accuracy of the model was the way the data was sampled for the grids defined within the geographic region of California.

The decision tree and logistic regression have higher precision and recall than MLP. However, these two methods might not perform well in the case of larger data.

Random Forest is one of the best machine-learning model to predict wildfires in California. Even though, the prediction accuracy achieved from the Random Forest was around 78 percent, improvisation using the hyperparameter tunning gave the best result compare to other models. hyperparameter tunning using grid search method able to evaluate maximum model performance by trying diffrent combinations, and able to give the final precision of 91 percent.

VII. Future Directions

For future models training on this data a few different approaches can be taken. In [3] paper to predict the motion of the fire based on these feature characteristics, Equations of Motion were created to model to movement of these embers from wild fires. The data set we have is limited only has the final polygon of the fire and the centroid of the fire. In order to build out the movement of the fire the following things can be done: First, the fire data is a geospatial polygon with a unique center, from the center point of the polygon the fire expands outwards based on fuel, resources, terrain, and wind direction. Since the fire is not a linear shape there are multiple points that define the edges of this polygon and each day that the fire is not contained the shape/size of the fire region increases in size. Based on these changes of the shape of the fire the location data would change day to day, the changes could be approximated over the lifetime of the fire to provide more accurate details of the fire boundary as it progress, in addition the date can be tied to each of these spatial boundary points.

If the dates from the Alarm Date to the Containment Date are taken into account, the geospatial temporal data can be generated that correlates to the spread of the fire over time allowing the machine learning model to take into account starting points of a fire and how the fire moves across the geographic region of California. Changing the sampling of the fire data to take into account these factors would allow more time based and geographic samples to be attached to weather features. This approach would minimize the amount of empty time periods without fire data. The approach will also provide data that shows the growth of fires over time, one thing that cannot be determined is how long a fire position remains active after becoming a location with a fire. At some point a resource/fuel for a fire will become unavailable starving the fire and only the areas of the fire with fuel and the right wind conditions will continue to grow and spread. In order to build a better model a decay algorithm could be added based on the average amount of time to consume resource within a spatial region, with this approach the decay algorithm must take into a spatial area in which to apply these restrictions on the data. Essentially, by adding a decay algorithm to the growth of the fire to the boundaries of the polygon the growth would be regularized and reduce the bias in the data generated from the centroid to the boundaries of the fire.

The strategy for dividing the geographic region California also needs to be reevaluated due to the large size of the tiled approach. The tiles should be broken up into smaller tiles to better represent the sub regions within California's geographic boundaries. Due to the high significance of the location feature, the 68 one degree grids can be broken down into 15 minute grids to create more location data within training set. In addition, another binary feature can be added called IsCentroid this feature will indicate if the location data is the origin point of the fire. Originally only the alarm date was used for weather data, the alarm data and all dates until the containment date should also be included create more data points of fire activity. Two other features can also be added as binary points, alarm date and containment date to indicate significant events in the date data associated with each fire.

In changing the amount of data points a model is needed that is able to process large amount of data, which in computational intensive. Another model that should be considered is

a Geographic Weighted Regression due to spatial components of the data. Another alternative that could be taken with the data, is to convert the fire centers and growth over time as images, then use models that can handle image processing. Each pixel within the image would have a geographic location tied to the pixel as well as weather data to map to that pixel at a given time. The images could also be gray-scaled, where no fire data is represented as a white pixel and areas with fires can be represented as black pixels. Another alternative with using images, is to change the intensity of a pixel based on the amount of fire that overlaps a particular region allowing a model to leverage the pixel intensity as part of training. In [3] the propagation of fire embers were based on the elevation of the starting point of the fire, considering this feature our dataset would need a way of determining the elevation of the fire data points. A few different approaches could be leveraged to determine the elevation at given such as using the Geoid model of the earth to determine the Mean Sea Level (MSL) at each location or Shuttle Radar Topography Mission (SRTM) data. The elevation would be another feature to add to the existing features used for build the training model. Due to wind being a significant feature in making predictions, the wind profile increases with the elevation of terrain which would lead to faster spread of fires at higher elevation. The fire embers would propagate faster as long as there were resources at these higher elevations. The terrain profiles would also allow for other things to be calculated such as the slope between data points and the average gradient either positive or negative depending on the terrain.

VIII. Conclusion

While wildfire is hard to predict in general, the machine learning algorithms could achieve decent precision in predicting whether with a given weather condition, there will be a wildfire or not. In this paper, we use weather data and wildfire incidents from 2013 to 2022. We implemented multiple machine learning methods to predict wildfire occurrence and perimeter based on weather data. The models we attempted to apply include multi-layer perceptron (MLP), logistic regression, decision trees, and random forest. Although we did not applied any optimization method, we still able to achieve accuracy which is more then 80 percent. For the future development of wildfire prediction, we think it is better to include human factors, such as regional population and industrial activities to establish a more robust model. Also, more reliable geographical features including elevation, slope, and soil type may also help improve the performance of prediction models

References

[1] A. Malik, N. Jalin, S. Rani, P. Singhal, S. Jain, and J. Gao, "Wildfire risk prediction and detection using machine learning in san diego, california," 10 2021.

[2] M. N. Mohammad Khaled Al-Bashiti, "Natural hazards research," 9 2022, pp. 154–165.

[3] A. C. F.-P. R.A. Anthenien, S.D. Tse, "On the trajectories of embers initially elevated or lofted by small scale ground fire plumes in high winds," 7 2006, pp. 349–363.

[4] S. M. D. J. Smith, L. Barfed and F. C. Harris, "Highly parallel implementation of forest fire propagation models on gpu," 2016, pp. 917–924.

[5] [Online]. Available: https://gis.data.ca.gov/datasets/CALFIRE-Forestry::california-fire-perimeters-all-1/about

[6] [Online]. Available: https://frap.fire.ca.gov/mapping/gis-data/

[7] [Online]. Available: https://www.kaggle.com/rtatman/188-million-us-wildfires

[8] [Online]. Available: https://psl.noaa.gov/data/gridded/data.ncep.reanalysis2.html

[9] [Online]. Available: http://ipm.ucanr.edu/WEATHER/wxactstnames.html

[10] P. et al., "Scikit-learn: Machine learning in python," 10 2011, pp. 2825–2830.

[11] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *Journal of Machine Learning Research*, vol. 18, no. 17, pp. 1–5, 2017. [Online]. Available: http://jmlr.org/papers/v18/16-365.html

[12] T. p. d. t. . Zenodo., "Pandas (v1.5.1)," 10 2022.