Wildfire Spread Prediction

Introduction

Problem statement: In recent years, the frequency and intensity of wildfires have been increasing, due in part to climate change and human interaction. Wildfires are a major threat to human safety and property and also have a significant impact on the environment. Our goal is to predict wildfire spread [location of wildfire] after 24 hours from the initial satellite images.

Impact statement: Correctly predicting the wildfire spread is critical for environmental management and disaster preparedness, which can protect life and property. This makes it more important than ever to develop accurate methods for predicting wildfire spread.

Machine learning methods: This project aims to develop a machine-learning model with supervised learning techniques of logistic regression, Random Forest, Gradient Boosted Classification, Linear Support Vector Machine, Stochastic Gradient Descent Classifier, Complement Naive Bayes, Gaussian Naive Bayes and a Generative Autoencoder combined with unsupervised learning methods of Principal Components Analysis, KMeans Clustering, and a Feature-Reduction Autoencoder. These models are used to predict the next-day spread of wildfires. The models will be trained on a dataset of historical wildfire data, including information about the fire's location, size, and variable environmental conditions. The optimal model will then be used to predict the spread of future wildfires.

Novel contribution statement: The primarily related work associated is the publication made on the same dataset. It is called *Next Day Wildfire Spread: A Machine Learning Data Set to Predict Wildfire Spreading from Remote-Sensing Data* provided by Fantine Huot et al. We furthered the approach made by utilizing a wide array of models including an auto-encoder and building custom features, an example being calculated pixel distance from the closest fire pixel as a representation of real-world distance.

Main findings: The main finding that we identified was during feature ablation that showed vegetation and its local gradient (rate of change) as a key metric for the classifiers to decide fire or no fire. Another finding was that even though our model accuracy was high, it's because our target value was binary with more than zero than 1 value. Our top model was a Complement Naive Bayes. Further approaches towards dealing with the classification imbalance should be considered for more competent runs of our machine learning models that work with high dimensional data.

Related Work

Next Day Wildfire Spread: A Machine Learning Data Set to Predict Wildfire Spreading from Remote-Sensing Data: [A.1]
Our project takes advantage of the data built upon a Google Earth Engine API pipeline. We utilized their approach to the problem of making this a binary classifier (fire or no

fire) for each pixel. They used logistic regression, random forest, and an autoencoder CNN for this prediction.

A Mathematical Model for Predicting Fire Spread in Wildland Fuels: [A.2]
This is a report done by Richard C. Rothermel in the U.S. Department of Agriculture and it focuses on deriving a formula based on the variables of heat sink, heat sources, wind coefficients, and slope coefficients. By doing the experiment by controlling the variable of “wind” and “slope” separately, different coefficients are calculated for 1 unit change of the other variable, which inspires us to generate the smoothened value, gradient value, and mean value for every feature in the raw data set in the stage of feature engineering.

Wildfire Spread Simulation: [A.3]
In this case study, the authors built a simulation model by calculating the fire spread rate with respect to the empirical parameters derived from observations and experimental data. In the experiment, the authors paid attention to controlling the moisture content and detecting the change in fire spread simulated by the variable of heating number and heat of pre-ignition.

Data Sources: Located on Kaggle, “Next Day Wildfire Spread” contains data from 2012 to 2020 and has a total of 18,445 samples. The dataset is 3.96 GB and is represented as compressed TFRrecord files. The timeline featured in our dataset ranges from 2012 to 2020 and involves accumulated satellite images covering 64 km^2 at 1 km resolutions, meaning each pixel is representing 1 km^2 of various locations where a wildfire occurred within the United States.

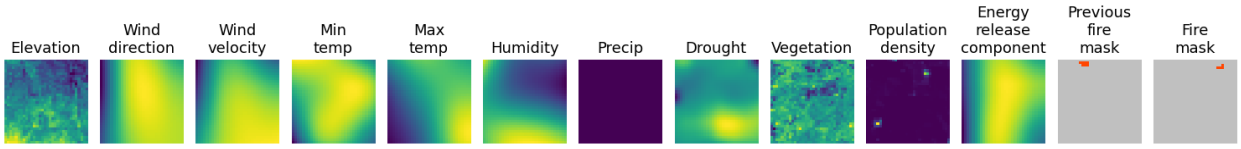
Data Source Variables: There are 11 features in total from the TFRecords. These features are categorized in the fields of wind, drought/moisture index, and temperature; the most important being the previous day’s fire mask and fire mask. The previous day is taken 24 hours prior to the fire mask. All features are in the format of image-based numpy arrays of dimension 64 pixels x 64 pixels. We made use of all TFRecord files that were split into train (15 TFRecords), evaluation (2 TFRecords), and test (2 TFRecords).

Figure 1. Sizes of Dataset After TFRecord-Tensor Extraction

	Training	**Testing**	**Evaluation**
Number of Images	14979	1689	1877
Extracted size after feature generation [rows]	61353984	6918144	7688192
Extracted size after feature generation (bytes)	14.675 GB	1.991 GB	2.208 GB

Initial pre-processing: The feature data was relatively clean. We applied Gaussian smoothing across all arrays main features, not including each fire mask, to minimize noise. There is missing data in the FireMask category, being our target label. The target label is binary as per the description by the dataset maker: 0 indicating no fire found, and 1, indicating fire was present. But due to the presence of objects, most likely cloud cover, the satellite images for variable quadrants of the image were not able to be determined and thus were labeled -1. For -1 labels in Previous FireMask, we did not change or edit, but they were dropped from the FireMask target variable. Any predictions made on these would be difficult to label. Furthermore, between the data description provided and the actual values, there were non-binary values present in FireMask and previous FireMask between 0 and 1. We took that as an assumption that there was some degree of fire present and thus we transformed them into the label 1 for our classifiers.

Figure 2. Plotted Single Random Sample of Training Data Set



The above figure is an example of one row of training image data out of the 14,979 training samples. Included variables from this historical wildfire information are elevation, wind direction, wind velocity, and more. Appendix B can be referenced or the image above for the entirety of base features.

Feature Engineering:

Since our model was built in a format of a pipeline, Every stage affected the features after that. Starting with the features extracted from the above tf records in the format of a Numpy array. Our pipeline followed the following steps:

Stage 1: TF record extracted features: Elevation, Wind direction, Wind velocity, Min temperature, max temperature, Max temp, Humidity, Precipitation, Drought, vegetation, population density, energy release component, previous fire mask.

Processing 1: Feature Generation: Gaussian smoothen arrays*, local pixel gradient*, local pixel mean*, fire at similar altitude (built using edge detection), pixel distance to nearest fire pixel (EDT between pixels). **Total feature count: 35**

* These features were created for all original features extracted.

Stage 2: Standardization: All features were Standardized using Sklearn standardScaler, this was done to better suit the needs of the upcoming PCA.

Stage 3: PCA: Optimal PCA components were 18, nearly half of the earlier feature count from earlier and they accounted for 90% variance in the training data.

Stage 4: K-means clustering: 18 PCA components were clustered into 8 clusters. We then built classifier models for each cluster.

Stage 5: Classifier models: this applies to all classifier models built: for each cluster, we trained a variety of classifier models, their inputs were the 18 principal components.

Part A. Unsupervised Learning

(1) Principal Component Analysis (PCA): Workflow & Evaluation:

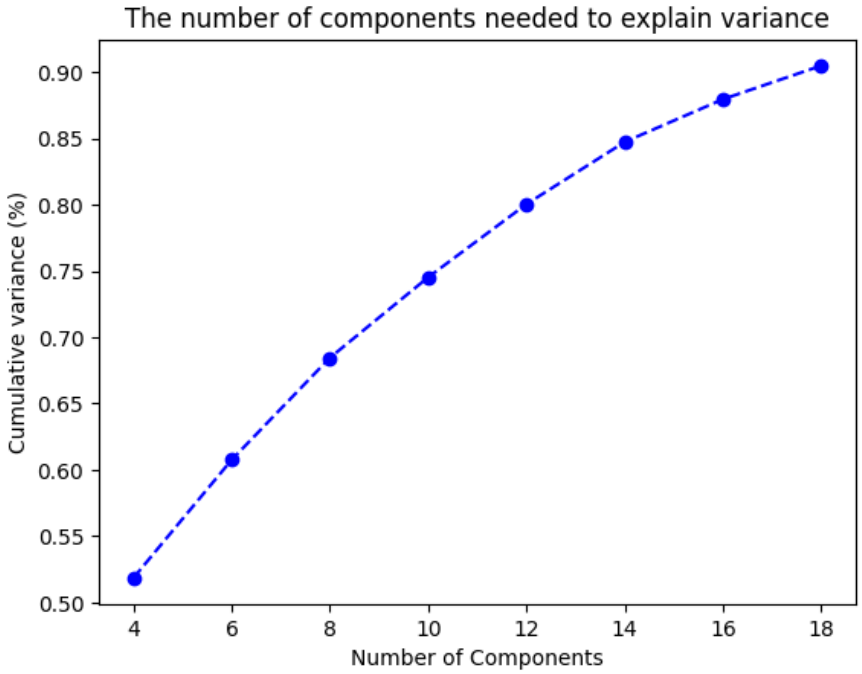
In order to find the optimal linear combinations of different numbers of variables, we performed hyperparameter tuning from 4 to 18 clusters with a step size of 2. Therefore, for every choice of the number of components, there is a corresponding PCA model generated and finally, we have 8 PCA models with different numbers of components. Having done feature generation on an already large-dimension tensor to numpy array data, we then make use of PCA. PCA is a good statistical procedure that is used to reduce the dimensionality of a dataset while preserving as much of the variance as possible: we retained over 90% variance from the dataset. Hence, to evaluate these 8 PCA modes, we stored the explained variances of all the variables' ratios for each model in a list of dictionaries, by sorting them in descending order by the ratio and we found the PCA_8 model with 18 components has the highest explained variance ratio. The final components from the PCA_8 model are 18 principal components.

The reason behind using PCA was that our data had original features and derived features from those original features, the data also needed to be reduced in size to allow model training as it was a relatively large dataset, especially with resource-constrained computing.

Figure 3. PCA Variance

PCA Model Name	Number of Components	Cumulative Explained variance ratio
pca_1	4	0.5184214069574967
pca_2	6	0.6075005628401904
pca_3	8	0.6844859985713041
pca_4	10	0.7452184719323623
pca_5	12	0.8003604683322194
pca_6	14	0.8472537840191601
pca_7	16	0.8794673767179796
pca_8	18	0.904786406168631

Figure 4. PCA Component Visual



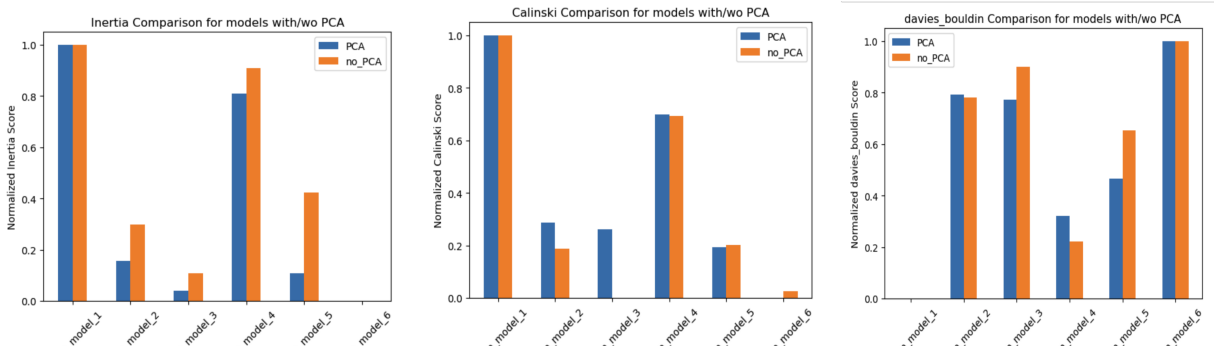
(2) KMeans Clustering Workflow & Evaluation:

Based on the optimal number of components we got from the result of PCA, we built the function to initialize the MiniBatchKMeans clustering model with a different number of clusters. Starting from the input file, we chunked them into subsets, each of which contains 1000 images. Next, for each subset, we first apply a scaling transformation followed by using the best model we found in PCA to convert it to a dataframe for training the clustering model. Since our training data set is extensive, we used the function of "partial_fit" to fit in memory at once, which is called several times consecutively on different chunks of the training data frame we chunked ahead of the training work.

To find the optimal number of clusters, we used three metrics: inertia (lower - indicates that the data points are more tightly clustered), Calinski and Harabasz score (higher- indicates that the cluster is well spread), and the davies_bouldin score (lower - clusters are not similar with the other). Based on these metrics choosing standards - lowest inertia, highest Calinski Harabasz score and lowest davies_bouldin score, we detect the optimal k-means clustering model of the parameters as 8 clusters with an inertia value of 12667339.851, Calinski Harabasz score of 480989.759 and davies_bouldin score of 1.722, which are in original value without any normalization processing.

Overall, we compared three metrics of “inertia”, “Calinski” and “davies_bouldin” on 6 KMeans clustering models (with different number_of_clusters). In order to properly compare these three measuring metrics among 6 models, we apply the min-max scaling and generate 3 bar plots, each of which shows one measuring metric across 6 models as follows:

Figure 5. Kmeans Metric Evaluation



According to the model selection methods we have demonstrated previously, we are supposed to choose the model with the lowest Inertia score, highest Calinskui score, and lowest davies_bouldin score. In this case, we can see that **model_1** has the highest Calinski score and lowest davies_bouldin score no matter whether the data set has been reduced dimensionality by PCA or not. However, the measuring metric of inertia is not suitable when the input is high-dimensional data. Because in high-dimensional spaces, the distance between data points tends to become less meaningful due to the "curse of dimensionality." Inertia is based on squared Euclidean distances, which may not reflect the actual similarities or dissimilarities between data points accurately. The dataset in our project is exactly this case. Besides, Inertia assumes the data is distributed in a way that makes squared Euclidean distances meaningful, and all the features in our dataset are not evenly distributed nor non-linearly separable clusters, therefore inertia is not the first reliable evaluation of clustering quality.

Part B. Supervised Learning & Evaluation

Due to the size and computation necessities of our dataset (as well as our processing limitations), supervised learning consisted of the pipeline of: load in entire csv data of training pixel information, scale, PCA transform, and then K-means transform. This process is consistent with testing data and evaluation data as well. PCA transformation utilized 18 components, a 19 variable reduction. The K-means transform separated the data into 8 clusters; this enabled iteratively faster model training as we had models representative per cluster. For supervised learning methods, we train multiple classification models.

Classification training, with varying hyperparameters, consisted of **Logistic Regression, Stochastic Gradient Descent Classifier (SGD), Random Forest, XGB Classifier, Complement Naive Bayes,** and **Gaussian Naive Bayes**. There were further attempts as well towards an **Generative AutoEncoder Neural Network** that

worked with only Previous Fire Mask data and the day after Fire Mask data. Varying hyperparameters for each model type are listed in Appendix C. These methods were used to train a classification model per cluster.
Using these allowed a transformed pixel-by-pixel evaluation in the prediction of binary classification of Wildfire presence for the label, Next-Day WildFire Mask.

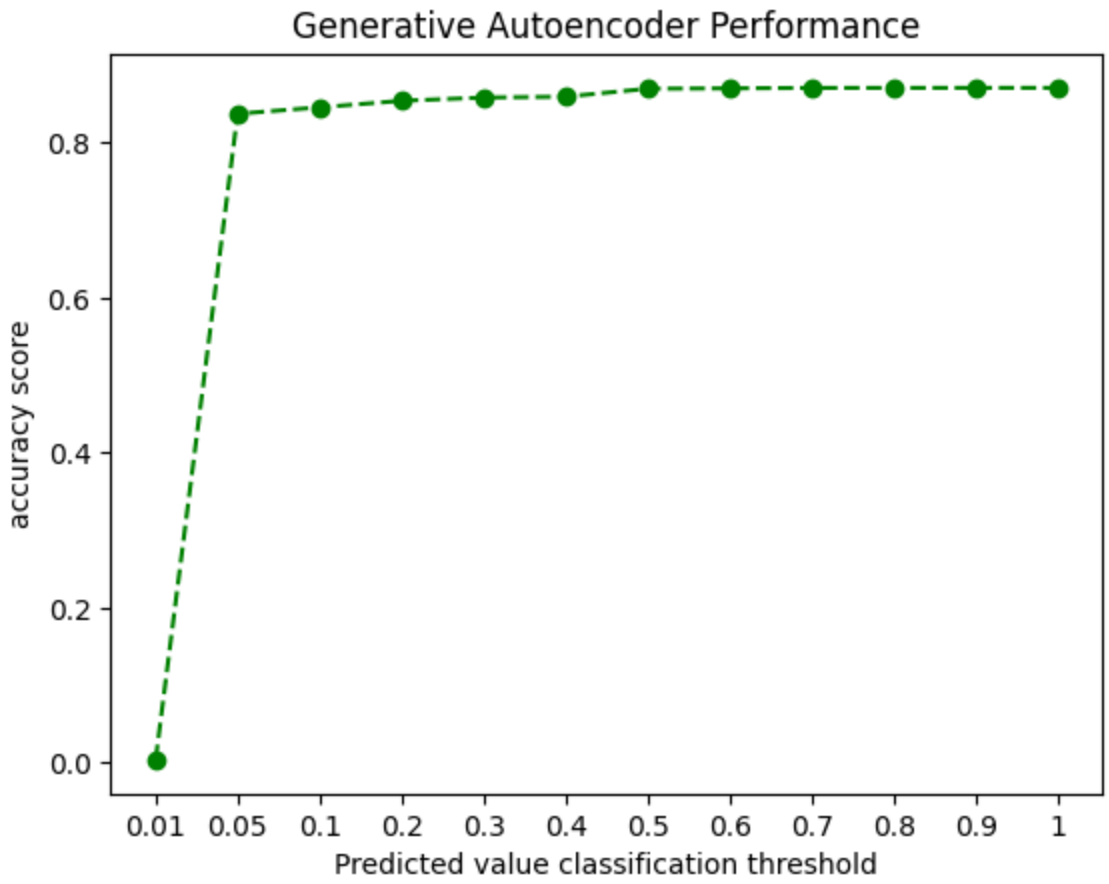
These classifiers were chosen for capability in binary classification, some in cases of high-data and others in classification imbalance. In cases like Complement Naive Bayes, we specifically used this model because of its design in the correction of “severe assumptions.” Meaning, it is well applied to imbalanced data. In the case of Gaussian Naive Bayes, that is used as a baseline comparison towards the use of Complement Naive Bayes.

XGB Classifier was used for its impressive ability to handle large-scale data and its binary classification accuracy. Random Forest was used as a baseline comparison as they are part of the same family of machine learning models.

SGD was applied for its range of parameters that could be used in its implementation for image classification. A baseline comparison used against this was Logistic Regression, as they use similar techniques.

Wanting to explore the potential possibilities in relation to autoencoders and their well-recorded performance with image data, an autoencoder was applied. Accuracy was recorded across the evaluation data set, with varying thresholds of predicted values being classified as 1.

Figure 6. Generative Autoencoder Neural Network



Most of the Supervised learning algorithms subsisted from the same pipeline of training. Since we approached most algorithms, other than the Neural Net, on a transformed pixel-to-pixel basis; there were balance issues in FireMask classification. There were many more pixels in which there was no fire compared to having fire. These results ended up in evaluation metrics with very high accuracies, which we determined to **not** be a good metric for those very reasons.

Therefore, we consider precision, recall, and F1 score more precisely as evaluations unless we only run predictions on target labeled data with Fire present. That alternative filter towards our data evaluations gives us a more accurate accuracy score. The below figure is a collective resulting metric measurement from each cluster related model over 5-Fold Cross Validations on the evaluation data after the varying models had been fitted on training data.

Determining scores for this portion was difficult, as the percentage of 1 labeled FireMask data greatly outnumbers out 0 labeled FireMask data during training. Many scores resulted in 0 as a result. Models performing the best are listed in the table below. There were also failures of loading data that resulted from truncated models.

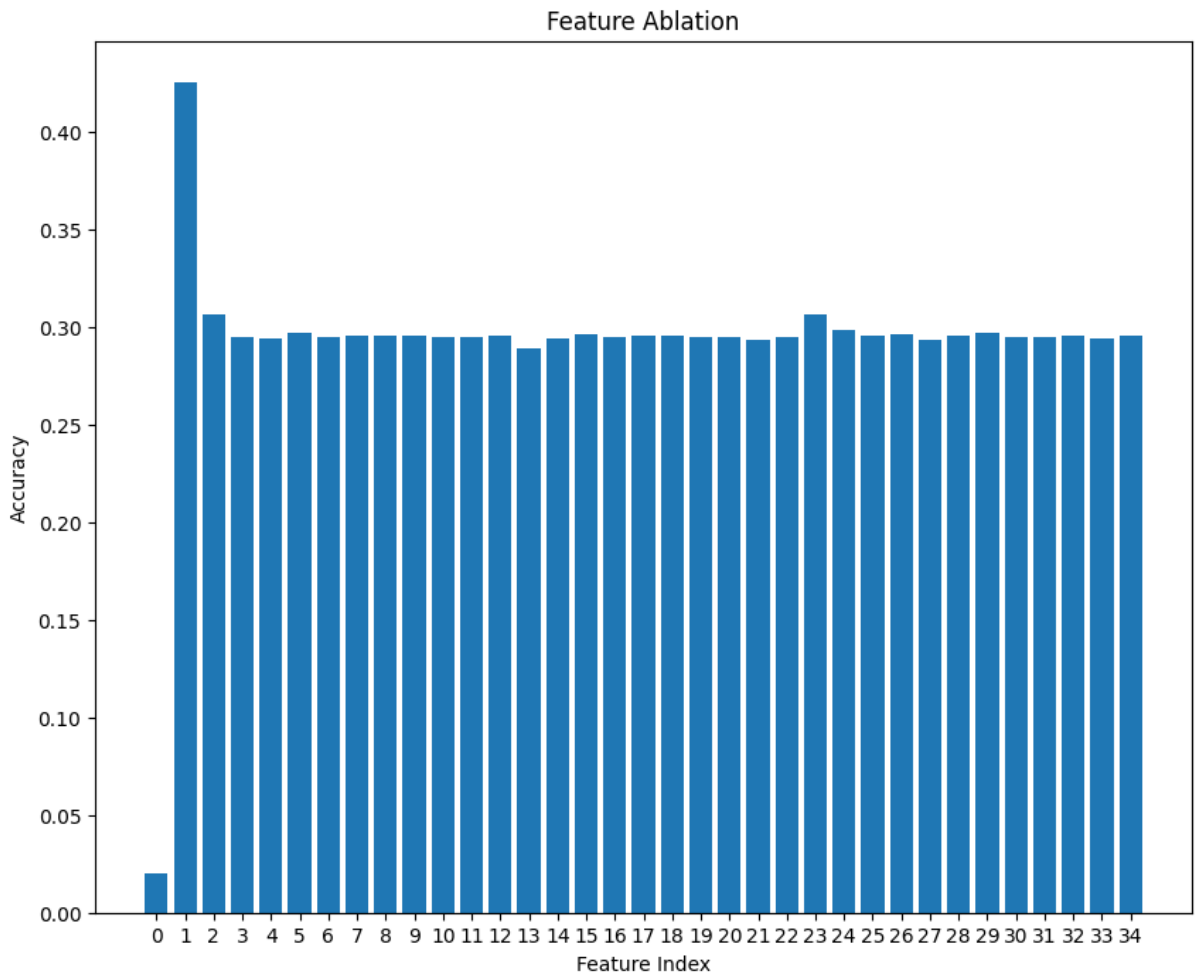
Figure 7. Metric Scores and Models Before and After Filtered Evaluation Data

	Accuracy	Precision	Recall	F1
Before Filter Model	XGBoosted Trees	Complement Naive Bayes	Complement Naive Bayes	Complement Naive Bayes
Before Filter Score	0.9989	0.003564	0.019373	0.003842
After Filter Model	Complement Naive Bayes	Complement Naive Bayes	Complement Naive Bayes	Complement Naive Bayes
After FilterScore	0.029059	0.75	0.029059	0.054829

Reference in Appendix C, there is a full graphic demonstrating the varying metric evaluations across the different model types. It was very notable that the non-filtered data and the filtered data result in vastly differing accuracy measures. The imbalance of classes in each cluster before merging of scores also resulted in some metrics resulting as NAN rather than 0. This was prominently seen in the filtered results, as we conclude that some clusters simply did not have enough positive Fire labels in the resulting Fire Mask.

Feature ablation was performed on the test data in prediction of our best performing model, Complement Naive Bayes. Two features have a noticeable impact on our already low accuracy metric, that being index 0 and 1; otherwise known as **NDVI_scaled_smoothened_values** and **NDVI_local_gradient**.The NDVI_scaled_smoothened_values end up really decreasing our performance when removed, whereas the NDVI_local_gradient end up increasing our performance.

Figure 8. Feature Ablation on Testing Data



Determining feature importance towards our best performing model, it was noticed that Feature 0 from PCA transformation has the most impact. Feature 8 (out of the 18) was seen to have the least. A table with Feature names and Importance values is shown in the below figure.

Figure 9. Feature Importance of Complement Naive Bayes Model From PCA Imputed Components

Feature 0	6.49	Feature 1	1.61	Feature 2	-0.18
Feature 3	0.09	Feature 4	0.25	Feature 5	-1.22
Feature 6	0.53	Feature 7	0.47	**Feature 8**	-2.86
Feature 9	-1.84	Feature 10	-1.27	Feature 11	0.90
Feature 12	-0.47	Feature 13	-1.05	Feature 14	-2.57
Feature 15	0.05	Feature 16	0.38	Feature 17	-1.64

Failure Analysis:

We found that the classifier of 1) XGBoot, 2) Random Forest, and 3) SDG classifier did not result in viable or non-zero precision, recall, or F1 scores. On the contrary, the accuracy of the prediction under these 3 types of classifiers is super high reaching almost 100%. It leads us to believe that the high accuracy rate from the correct prediction of the no-fire cases. For an imbalanced dataset, we can not rely on the accuracy to evaluate. F1 score provides a balanced evaluation metric that considers both precision and recall, and it is uncommon for a good model to have both 0 precision and recall, which performs poorly in predicting the real positive and negative values. Therefore, these 3 classes of classifiers fail the evaluation process.

Discussion

Unsupervised Learning

Usually, the optimal number of clusters is affected by the size of the dataset, and in general, the larger the dataset the more clusters we need for better performance. Given our extremely large dataset, the optimal result is not as large as we expected before applying any clustering techniques. The reasons for this we can infer that it is due to the layout and structural characteristics of our data set that we have a lot more rows than the number of columns and for each row, the values for each row and each column is an array with many duplicate numbers. This is caused by the method we read the TFRecord files so that there are lots of duplicate numbers/ pixel values, so the actual informative values for each row are not as much as the number of rows itself. Therefore, indeed don't we need so many clusters to separate different information clusters.

The first challenge we met was noise in the data units and data points distribution are different for each feature, and how to choose a proper way to normalize the data can help the best avoid the effect of the extreme value is the first important thing we need to consider in the process of preprocessing the raw data. Because the features in our dataset have a different range of values and not so many are dense in the center (around median if we put them in the quantiles measurement) in which case, the most proper way for us is to choose the min-max scaling. Secondly, because there is no definitive way to find the optimal number of clusters, we address this challenge by using different evaluation metrics of scores to measure the similarity between clusters and variance between clusters and choose the overlapped model parameter as the optimal model instead of relying on only one metric.

If we have more time and resources, we would collect more data on the historical data in different regions, sizes and spread of wildfires. Besides, we can also search for other factors that could affect wildfire spread, such as the slope of the terrain (mentioned in the 2 related works), the presence of the road and building surrounding and even the presence of firefighters. Due to the restriction of computation resources we can use, we did not choose hierarchical clustering since the time complexity of which is O(n^3),

which is pretty time-consuming. Therefore, if we have more resources, we could try different clustering techniques and do a comparison of results from different clustering methods.

These varying clustering techniques may have also been able to impact how we approached our supervised methods, due to the nature of our pipeline.

Supervised Learning

For the supervised learning part, from the accuracy measuring metric, there is no big difference in all the supervised learning methods. In addition, we found that the variables of Vegetation, wind direction and “distance_from_fire” play the most important role in classifying the fire mask. Within these important features, we found the feature of wind direction is also mentioned in both related work papers.

Firstly, since all the prediction work is done by applying the supervised learning techniques to each cluster generated from the unsupervised learning part, which leads to the issue that it is hard to interpret the model. To respond to this problem, we decide to calculate a mean of 4 elements of the confusion matrix, as an overall mean for all the clusters for one supervised learning algorithm.

Secondly, for our attempt towards the Neural Network, due to the nature of complex image inputs and outputs and our lack of domain knowledge, we attempted a variation of different CNN architectures but then decided to train an AutoEncoder Neural Network instead. The auto encoder was simple due to the training resource constraints. It takes the previous day's fire mask and tries to predict the next day's fire mask. It utilizes 4 layers in total. Its best-tuned model achieved an accuracy close to 90% which is lower than the Complement Naive Bayes before filtering for only positive fire values. We assume that, in a comparison of just filtered data, the autoencoder would’ve outperformed the other machine learning model.

If we had more time, we could investigate whether the high accuracy rate is from the correct prediction of the no-fire portion whereas we could use our model to predict the fire portion is more important under the project background. We can also spend some time identifying whether there is an issue of overfitting by gathering more test sets to test and tune the model to avoid the problem that our optimal model performs excellently in training and performs poorly given a new test set. Thirdly, CNN typically requires a large amount of labeled training data to learn complex patterns, therefore we need to spend more time augmenting the data to mitigate the limited training data challenge. Following this, we need to investigate deeply the regularization techniques such as dropping out and early stopping to avoid overfitting.

A final point that would’ve been ideal to explore would be visually comparing graphically produced predictions of wildfire.

Ethical Considerations

Unsupervised learning algorithms can be biased, which means that they may learn to associate certain features with wildfires that are not actually predictive of wildfire spread

and this could lead to the algorithm making inaccurate predictions. Besides, it is difficult to hold the algorithm accountable for its mistakes, which means that if an unsupervised learning algorithm makes an inaccurate prediction and leads to a loss of life or property, it may be difficult to hold anyone accountable.

To mitigate the ethical risks of using unsupervised learning in the case of wildfire spread, the data set we use to train should be as diverse as possible. Secondly, we also need to consider applying some specific privacy-preserving algorithms ahead of training the unsupervised learning algorithm. Thirdly, in the event that the algorithm makes an inaccurate prediction that leads to a loss of life or property, there should be a clear and transparent plan for holding someone accountable.

There is a potential ethical issue of data privacy since for now the fire data was collected from a certain geographic location and demographic groups, therefore the hybrid model (combination of clustering nested with a supervised learning model) might work diversity in this case and this may also cause the privacy concern with the risk of sensitive information leak. From the population density feature, we can find that in some images, the population density is really high. If the home address information and other demographic information about residents who live in the area belongs to our data set we need to protect their privacy before we do the prediction work. For this project, to address this potential risk, we used 6 different supervised learning algorithms which are based on different calculation and classification methods. Besides, we also implement regularization in Lasso regression to give a penalty if the classifier relies too heavily on one or several specific variables.

Appendix

Appendix A - Bibliography

[1]for F. Huot, R. L. Hu, N. Goyal, T. Sankar, M. Ihme, and Y.-F. Chen, "Next Day Wildfire Spread: A Machine

Learning Data Set to Predict Wildfire Spreading from Remote-Sensing Data", arXiv preprint, 2021.

https://www.kaggle.com/datasets/fantineh/next-day-wildfire-spread

https://arxiv.org/abs/2112.02447

[2] A Mathematical Model for Predicting Fire Spread in Wildland Fuels

https://www.fs.usda.gov/rm/pubs_int/int_rp115.pdf

[3] (PDF) wildfire spread simulation - researchgate. (2015, December). https://www.researchgate.net/publication/325085096_Wildfire_Spread_Simulation

Appendix B - Data Schema

Data Source:

- https://www.kaggle.com/datasets/fantineh/next-day-wildfire-spread

List of complete variables before feature generation:

- Elevation
- Wind Direction
- Wind Velocity
- Min. Temperature
- Max. Temperature
- Humidity
- Precipitation
- Drought
- Vegetation
- Population Density
- Energy Release Component
- Previous Fire Mask
- Fire Mask

Appendix C - Hyperparameter Tuning

Figure 10. Hyperparameter options utilized for Supervised Classification models

Model	**Param1 Options**	**Param2 Options**	**Param3 Options**
Logistic Regression	Penalty: [l1]		
SGD	Penalty: [l1 ,l2 ,elastic net]	Random State: [0]	
Random forest	N_estimators: [100,200,400]	Min_samples_split: [4096]	Random State: [0]
Linear SVC	Penalty: [l2]	Random State: [0]	
XGB Classifier	N_estimators: [100,200,400]	Tree_method: [‘gpu_hist’]	Objective: [‘binary:logistic’]
Complement Naive Bayes			
Gaussian Naive Bayes			

Appendix D - Figure 11 Mean Metrics of 5-Fold CV across Supervised Classification Models