Application of Machine Learning Techniques To Predict Wildfire Risk Using Satellite Imagery

Stanford CS229 Project

Anisha Palaparthi
Department of Computer Science
Stanford University
anishapv@stanford.edu

Nachiketh Karthik
Department of Mechanical Engineering
Stanford University
karthi24@stanford.edu

1 Introduction

The frequency of wildfires in North America has been increasing in recent years due to climate change and the associated changes in weather patterns. This trend has had a devastating impact in California, where annual wildfires destroy thousands of communities. The start and spread of wildfires depend on various factors, including fuel conditions, wind currents, terrain features, and human activity. The Machine Learning (ML) community has shown growing interest in applying ML techniques to address the wildfire problem. However, these efforts often require vast amounts of data; many previous studies rely on synthetic data or intricate geospatial files Shen et al. (2023) and Sousa et al. (2020), making them difficult to extend due to advanced knowledge or specialized equipment requirements. Furthermore, the best-performing models are typically deep learning-based, which are computationally expensive and require significant amounts of data to train.

These challenges motivate our project to classify wildfire risk using satellite images. We aim to explore how unsupervised learning as an initial step can improve ML model performance while avoiding the computational overhead of deep learning. The input to our algorithm is a color satellite image of size 320x320 and the task is a classification task to predict between 7 classes of wildfire risk. We use the supervised learning methods of multiclass support vector machines (SVM's) and a naive majority-class predictor to establish a baseline. Finally we propose a K-Means unsupervised learning based method with PCA and hypothesize better quantitative performance compared to the baseline.

2 Related Work

The related work on wildfire risk assessment can be divided into two major categories: GIS-based methods and remote sensing techniques combined with deep learning.

GIS-based methods have traditionally been used for fire risk assessment, relying on geographical information systems to analyze environmental features and create fire risk maps. These methods often consider variables such as vegetation, topography, and climate data to estimate fire hazards. Examples of such work include the study by Chuvieco et al. Chuvieco et al. (2014) which introduced a comprehensive model for fire risk using environmental features, and the work by Oliveira et al. Oliveira et al. (2021) that developed a fire risk assessment framework integrating human and environmental factors. While these approaches provide valuable insights, they often suffer from limited generalizability and inflexibility due to static and regional data features.

In recent years, deep learning models applied to remote sensing imagery have gained momentum in wildfire risk assessment, leveraging high-resolution satellite and aerial images to classify fire risk zones. Researchers such as Wang et al. Zhang et al. (2019) and Ghali et al. Ghali and Akhloufi (2023) have used convolutional neural networks (CNNs) to assess wildfire risks, demonstrating the potential of deep learning for automating feature extraction from complex imagery. Many of these studies employ supervised learning models to predict fire-prone areas, but these approaches are typically data-intensive, requiring large volumes of labeled data to achieve robust performance. To address this, recent works have explored self-supervised learning techniques, as highlighted in studies by Grill et

al. Grill et al. (2020) and Caron et al. Caron et al. (2021), which aimed to reduce the dependency on labeled data by leveraging large-scale unlabelled datasets to pre-train models, thus improving fire risk prediction.

The FireRisk dataset Shen et al. (2023) used in this project aims to bridge the gap between traditional GIS-based methods and modern deep learning approaches by offering a comprehensive labeled dataset for supervised learning benchmarks as well as an unlabelled dataset for pre-training with self-supervised methods. This combination allows researchers to evaluate both data-intensive and data-efficient modeling techniques. The benchmark models evaluated in the paper included ResNet, ViT, DINO, and MAE, focusing on fire risk classification using only remote sensing imagery. Specifically, the use of MAE as a self-supervised model achieved the highest accuracy among all benchmarks, as shown by He et al. He et al. (2021), illustrating the efficacy of self-supervised learning for fire risk assessment.

3 Dataset and Features

We will be using the FireRisk dataset Shen et al. (2023). This consists of 320x320x3 RGB images taken from satellites. The dataset also provides a label along with each image that describes the wildfire risk associated with the area of the image. There are 7 possible labels: "Very Low," "Low," "Moderate," "High," "Very High", "Non-Burnable", and "Water". The task on this dataset is to receive the RGB image as input and classify the image into one of these 7 risk classes. There are approximately 70,000 examples in the training set and 21,000 examples in the validation set.

Data preprocessing includes also a flattening of each image from size RGB 320x320x3 to black-and-white (BW) 64x64 to fit within the scope of the compute available to us in the project. We also use feature scaling using the standardscaler from the scikitlearn before inputting data into PCA as well as inputting data into the SVM models, which are further described below. Example images from each of the classes can be found in Figure 2 in Appendix C.

4 Methods

4.1 Machine Learning Model

We will be considering Support Vector Machines as the supervised learning model. Support Vector Machines (SVMs) can be used as multi-class classifiers with less computational expense compared to deep learning approaches. They are effective when dealing with high-dimensional data as well. In the multiclass setting, SVMs are used to create decision boundaries that separate data points of different classes.

The SVM aims to maximize the margin between data points and the decision boundary. The **optimization objective** for SVM with hinge loss is given by:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)) \tag{1}$$

where $\mathbf{w}$ is the weight vector, b is the bias term, $y_i \in \{-1, 1\}$ is the label of sample i, and C is the regularization parameter that controls the trade-off between maximizing the margin and minimizing classification errors. The SVM optimization problem can be solved using techniques such as **SMO (Sequential Minimal Optimization)**, which decomposes the main problem into smaller subproblems, making it more efficient for large datasets. Furthermore, we use a Radial Basis Function (RBF) Kernel as that has been shown to support non-linear decision boundaries Han et al. (2012).

For multiclass classification, the **One-vs-Rest (OvR)** strategy is often used. In OvR, a separate SVM classifier is trained for each class, treating that class as the positive class and all others as negative classes. The final decision is made based on the classifier with the highest output score.

4.2 Proposed method

We propose to incorporate an initial unsupervised learning step in the pipeline before training any ML models and hypothesize that we can non-trivial performance without using resource expensive deep learning models used by the researchers (see appendix A) using a simpler model and lesser compute.

The ML pipeline will be as follows:

1. The input images will undergo dimensionality reduction using Principle Component Analysis (PCA). Specifically, the 320x320x3 RGB images in the training dataset will be converted to scaled-down BW 64x64 images and normalized. We will then conduct PCA to reduce each image to a 400-dimensional vector that encodes the image features. (See Appendix D for an explanation of PCA)
2. Next, we will conduct KMeans Clustering on the low-dimensional image vectors to find groups of similar images among the dataset (See Appendix D for an explanation of K Means)
3. Finally, we will train k models on each of the subsets corresponding the k clusters found in the previous step (again normalizing the inputs to the models).

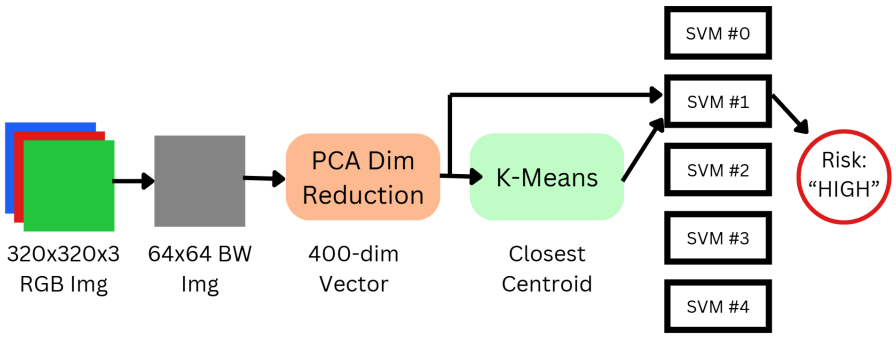

Figure 1: Proposed Method - Evaluation

The reasoning behind this pipeline is that we hypothesize that the SVM on its own may not be able to model the complex decision boundaries of 7-class classification on these images. However, using PCA to extract important features can reduce the need for more data and correlations within the input data that make it difficult for the SVM to model the decision boundaries. Furthermore, we use K-Means to divide the entire training set into 5 subsets (since K=5) that are similar to each other. In doing so, we hypothesize that the decision boundaries between classes within the subset of similar images will be less complex than attempting to model the decision boundaries with respect to the entire, diverse dataset. So, we hope to leverage a less expensive model like an SVM if we employ an initial unsupervised learning enhancement.

When evaluating the unsupervised-enhanced models, we will first compute the closest cluster centroid to the input image. We will then pass the input image through the model corresponding to that cluster's corresponding model. This process is visualized in Figure 1

5 Experiments / Results / Discussion

5.1 Experimental Setup

In order to evaluate our proposed method, we will attempt to address the classification task of this dataset using three models:

1. A rudimentary predictor that always predicts the majority class ("Very Low") - referred to as "Majority"
2. An SVM without the proposed method enhancements. Specifically, a single SVM that takes the 64x64 BW image as input (normalized by mean/standard deviation) and outputs one of the 7 classes - referred to as "Baseline"

3. Our proposed method. Specifically, using PCA to extract features from the 64x64 BW image, finding the closest centroid to this data, and evaluating the class for this data using the corresponding SVM - referred to as "Pipeline"

All three models are trained with 50% of the dataset (approx. 35K sample images) and evaluated on an unseen test set of that is 15% of the total dataset (approx. 10K images).

The primary metrics considered for model evaluation were **Accuracy**, **Precision**, **Recall**, and **F1 Score**. More information on each of these metrics can be found in Appendix C. Furthermore, this risk classification task is difficult, as deep learning approaches such as transformers achieve approx. 62% accuracy on this task (see Appendix C for more details)

5.2 Experiment Results

Tables 1, 3, and 2 show the results of these experiments. Figures 9, 10, and 4 show the confusion matrices produced by each of the three models (the baseline and majority models' confusion matrices can be found in Appendix C).

When we analyze these results, we observe that an SVM by itself is unable to achieve good performance, as the accuracy it obtains is less than that of the naive majority-selection classifier. However, when applying the unsupervised learning enhancement (referenced as "Pipeline" for brevity) of our proposed method, the performance increases non-trivially to 37% accuracy.

Class	Precision	Recall	F1-Score
Very_Low	0.36	0.27	0.31
Low	0.09	0.03	0.04
Moderate	0.14	0.02	0.03
High	0.00	0.00	0.00
Very_High	0.00	0.00	0.00
Non-burnable	0.29	0.75	0.42
Water	0.00	0.00	0.00
Accuracy	0.29		

Table 1: Classification Report - Baseline

Class	Precision	Recall	F1-Score
Very_Low	0.31	1.00	0.47
Low	0.00	0.00	0.00
Moderate	0.00	0.00	0.00
High	0.00	0.00	0.00
Very_High	0.00	0.00	0.00
Non-burnable	0.00	0.00	0.00
Water	0.00	0.00	0.00
Accuracy	0.31		

Table 2: Classification Report - Majority

Class	Precision	Recall	F1-Score
Very_Low	0.37	0.66	0.47
Low	0.28	0.01	0.02
Moderate	0.24	0.01	0.02
High	0.22	0.10	0.14
Very_High	0.00	0.00	0.00
Non-burnable	0.38	0.58	0.46
Water	0.45	0.10	0.17
Accuracy	0.37		

Table 3: Classification Report - Pipeline

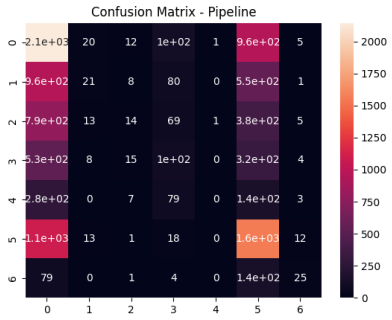

Table 4: Confusion Matrix - Pipeline

5.3 Visualizations of Unsupervised Components

(See Appendix C for Figures). Figure 6 shows the results of applying PCA with 400 components to four example images, leading to explained variance level of 86%. As Figure 6 demonstrates, our implementation of PCA is able to capture the predominant features of the image at the much lower resolution of a 400-dimensional vector.

After conducting PCA on the images, we run KMeans to generate 5 clusters. Figure 5 shows example images drawn from across each of the 5 clusters. We can see that images with similar features are

grouped in the cluster. Cluster 3 as shown in Figure 4 exemplifies how images that contain the feature of large textured patches of grass/trees are grouped together.

5.4 Analysis of Feature Importance

We have analyzed the importances of each feature with respect to our pipelined model. Since we used an RBF kernel to parametrize our SVM, we determine these importances through permutation feature importances Breiman (2001). The importance of a feature X is calculated by holding all other features constant while randomly shuffling feature X and determining the drop in the model's performance after this permutation. Figure 3 (Refer Appendix C) visualizes these feature importances with respect to the 5 SVMs for each of the 5 clusters. As the graphs illustrate, the importance of a given feature changes depending which model is evaluating that data. This is as expected since each of the models are trained on each different cluster, so each model learns to focus on different features in order to classify that cluster's type of image.

One aspect that is particularly interesting to note is that while Models 1-4 hold the majority of features as important, Model 0 actually contains several features that have negative importance. In other words, for the images assigned to Cluster 0, certain features extracted by PCA are, in fact, not important to the risk classification for the SVM.

5.5 Effect of Pipeline on Feature Correlation

One consequence of introducing the unsupervised step, particularly by using PCA, prior to applying the supervised model is its impact on the feature correlation. (See Appendix C for figures referenced) Figure 7 shows the feature correlation matrix when the input to the model are the raw 64x64 black and white images, while Figure 7 shows the feature correlation matrix for 400-dimension feature vector generated by PCA. We know that correlation features can negatively impact simpler models like SVM. Figure 7 shows that many of the features (pixel values) are highly correlated with each other which contributes, in part, to the low performance of the baseline model. On the other hand, features outputted by PCA have much less correlation with each other as illustrated in Figure 8

5.6 Hyperparameter Tuning

The SVM was implemented using a trial and error sweep approach across values 0.001, 0.01, 0.1, 1, 10, 100 for hyperparameters C (Regularization hyperparameter) and Gamma (Kernel coefficient). The Scikitlearn module allows the user to set gamma to *scale* and more information on this argument can be found in Appendix B. Through this process, we chose C=1.0 and Gamma=scale as the hyperparameters for SVM.

For PCA, we chose the hyperparameter for 400 components. This was chosen because 400 dimensions was enough to explain much of the variance (about 86%) while being low enough for the models to perform within the computational constraints. Similarly, we chose 5 clusters for K-Means as we were able to tractably train 5 SVM models.

6 Conclusion / Future Work

Deep learning is capable of modeling complex distributions, and as a result is the focus of much research in the task of wildfire risk classification. However, the large amount of resources and computational capacity make it difficult to apply in all scenarios; accordingly, we aimed to build an ML pipeline that enhances a simpler ML model like an SVM through an initial unsupervised learning step. We have shown that our proposed model achieves non-trivial performance on a difficult image classification task.

For future directions, we would like to explore what other machine learning models would benefit from the pipeline implemented here. We also reduced the RGB image to BW in order to save compute, but we hypothesize that one could achieve better performance if they could encode the color features as well. We would also explore if the same principles of clustering and dimensionality reduction in our unsupervised step could potentially improve the performance of deep learning systems as well.

7 Contributions

Both team members contributed equally to the project. Nachiketh implemented the baseline experiments worked on the related works and other explanations in the paper. Anisha implemented the unsupervised algorithms and worked on the experiments section. Both team members contributed to the idea, coding, and paper writing of the project.

8 Appendices

8.1 Appendix A

The FireRisk dataset paper Shen et al. (2023) presents the performance of several supervised and self-supervised models. The supervised models include ResNet-50 and ViT-B/16, while the self-supervised models include DINO and MAE, both using ViT-B/16 as their backbone. The results highlight that the MAE model pre-trained on ImageNet1k achieved the highest accuracy (65.29%) and F1-score (55.49%) compared to other models.

Table 5: Performance of several supervised and self-supervised models on FireRisk with different sizes. In our implementation, our supervised benchmark uses ResNet-50 and ViT-B/16, and the self-supervised models, DINO and MAE, use ViT-B/16 as backbone.

Dataset	Model	Pre-trained	Acc.	F1
FireRisk	ResNet	ImageNet1k [8]	63.20	52.56
	ViT	ImageNet1k	63.31	52.18
	DINO	ImageNet1k	63.36	52.60
	DINO	UnlabelledNAIP	63.44	52.37
	MAE	ImageNet1k	**65.29**	**55.49**
	MAE	UnlabelledNAIP	63.54	52.04

8.1.1 Appendix B: Gamma = 'scale' Explained

When you set `gamma='scale'`, the value of γ is computed automatically based on the training data.

Specifically, it is defined as:

$$\gamma = \frac{1}{n_{\text{features}} \cdot \text{Var}(X)} \tag{2}$$

where:

- n_{features} is the number of features in the training dataset.
- $\text{Var}(X)$ is the variance of the features.

In other words, `scale` sets γ to the inverse of the number of features times the variance of the data, which makes it data-dependent and provides a reasonable value.

8.1.2 Appendix C: Additional Figures

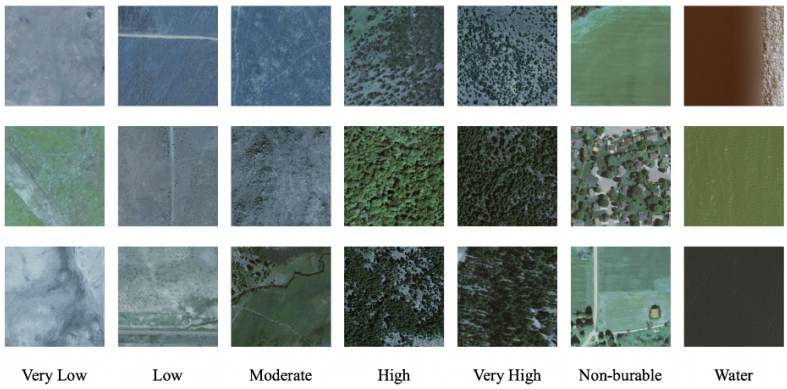

Figure 2: Dataset examples showing three example images of each class (Shen et al. (2023))

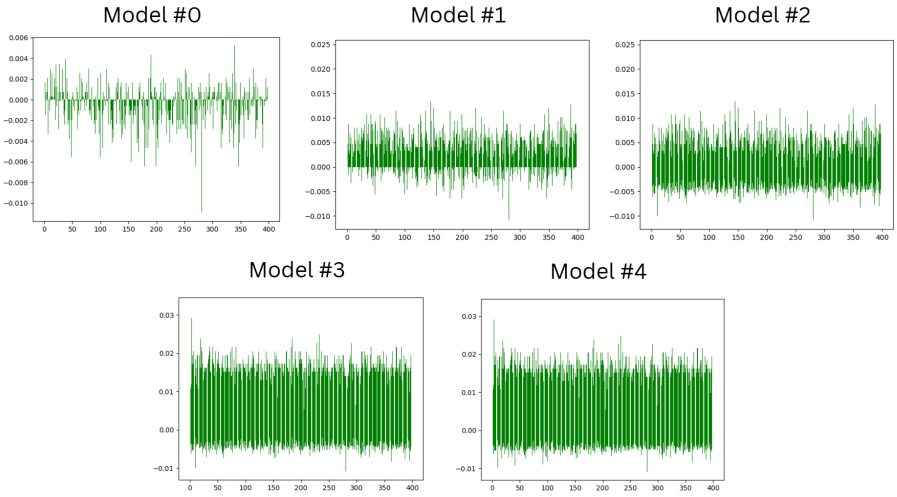

Figure 3: Feature Importances

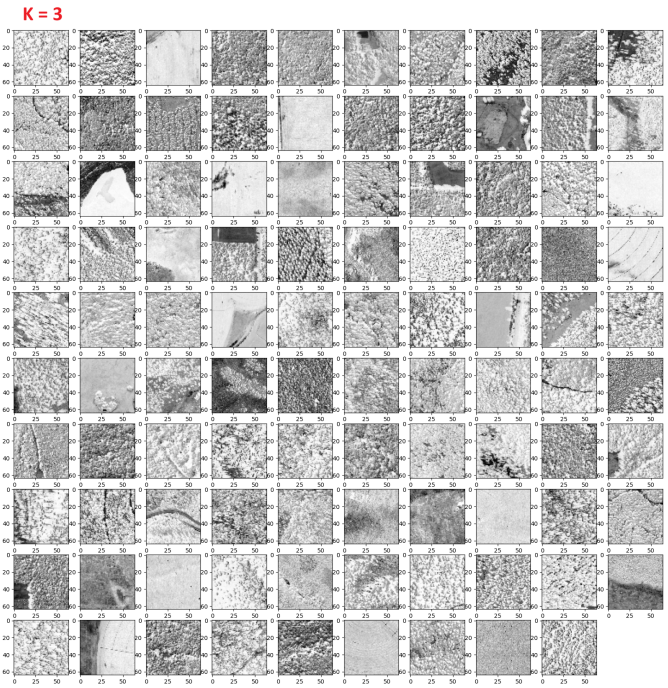

Figure 4: Cluster 3 Magnified

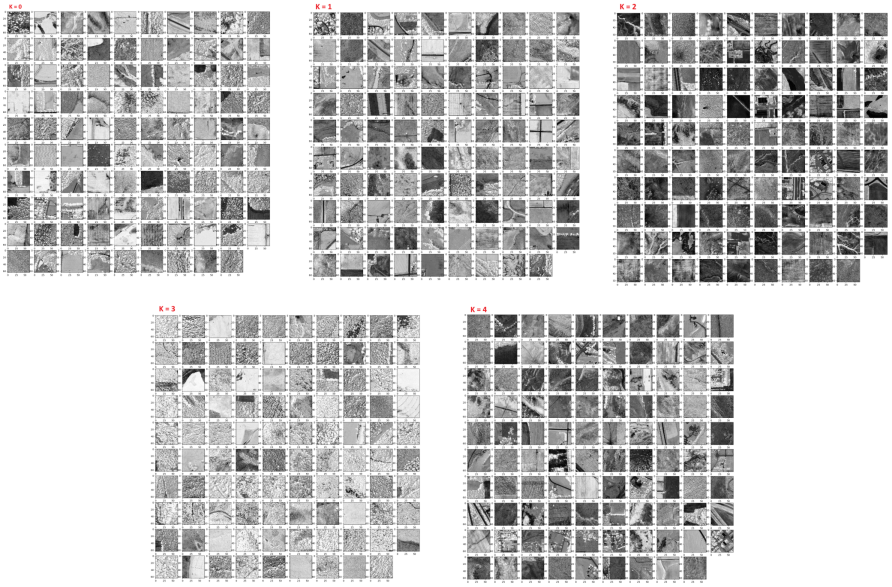

Figure 5: 5 KMeans Clusters Visualized

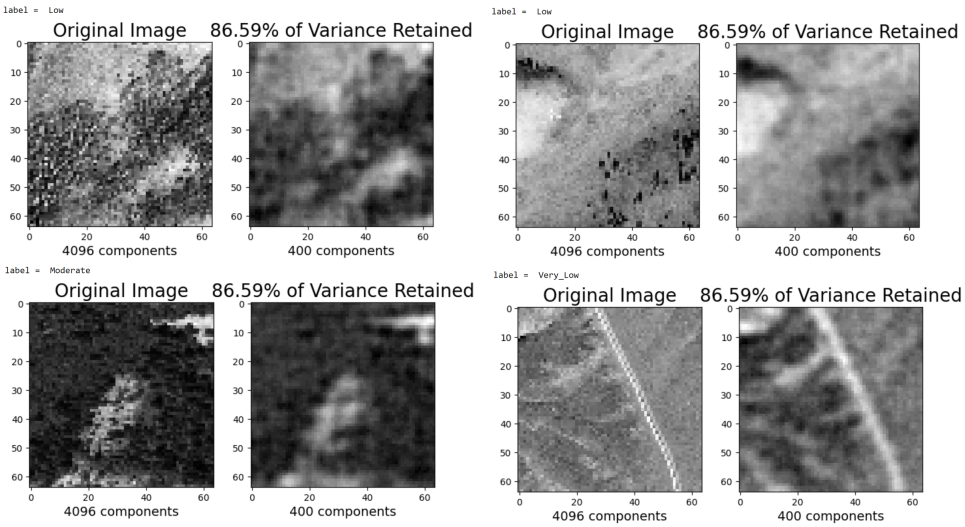

Figure 6: PCA Applied to Several Image Examples

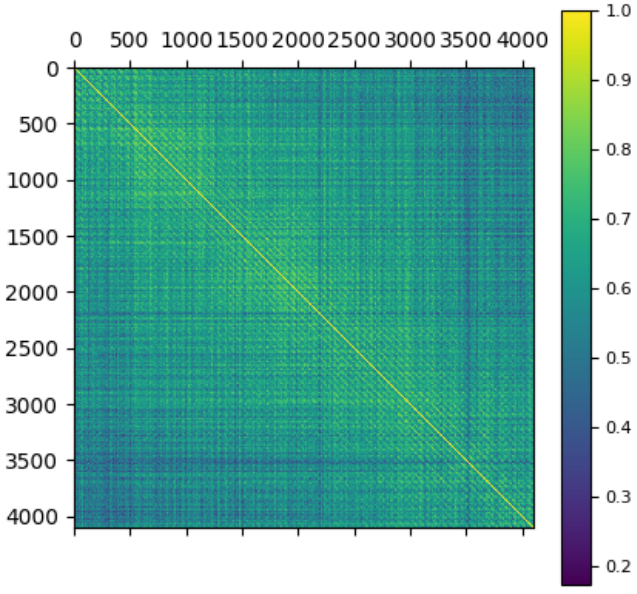

Figure 7: Baseline Correlation

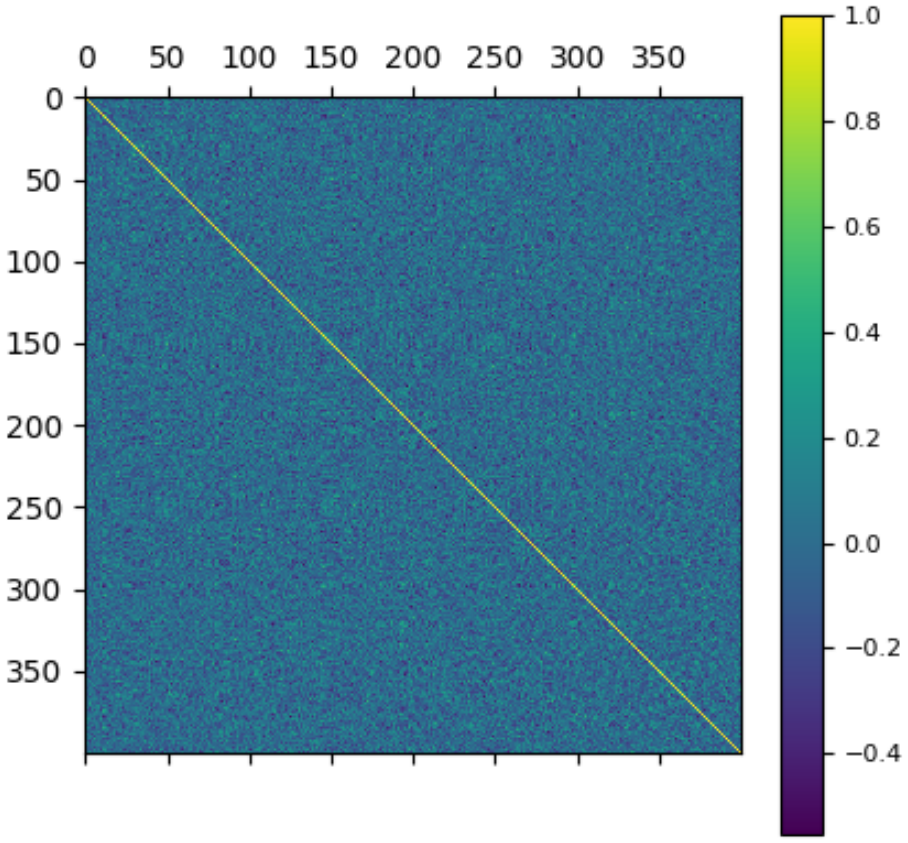

Figure 8: PCA Correlation

8.1.3 Appendix D: PCA and KMeans

Principle Component Analysis is a (unsupervised) dimensionality reduction algorithm. This algorithm can be calculated by computing the covariance matrix in order to identify correlations in the data. Then, the algorithm must compute the eigenvectors/eigenvalues of this matrix so as to identify the components. The first k components (those with the largest eigenvectors) are chosen as the principal component and form the basis of the lower dimension space.

The K-Means algorithm is a popular clustering method used to partition a dataset into k distinct clusters MacQueen (1967). It minimizes the within-cluster sum of squares iteratively. Given a dataset $\{x_1, x_2, \ldots, x_n\}$, where each $x_i \in \mathbb{R}^d$, the algorithm seeks to find k cluster centroids $\{\mu_1, \mu_2, \ldots, \mu_k\}$ that minimize the following objective function:

$$J = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2, \tag{3}$$

where C_i represents the set of points assigned to cluster i, and $\|x - \mu_i\|^2$ is the squared Euclidean distance between a point x and the centroid μ_i.

The algorithm operates in two main steps:

1. **Assignment Step**: Each data point is assigned to the nearest cluster centroid based on the Euclidean distance.
2. **Update Step**: Each centroid is updated as the mean of the points assigned to its cluster:
$$\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x, \tag{4}$$
where $|C_i|$ is the number of points in cluster i.

These steps are repeated until convergence, typically when the assignments no longer change or when the reduction in J becomes negligible.

8.1.4 Appendix C: Primary Metrics

In this section, we describe the primary metrics used to evaluate the performance of our machine learning models. The metrics considered are **Accuracy**, **Precision**, **Recall**, and **F1 Score**. Each of these metrics provides valuable insights into different aspects of the model's classification performance.

8.2 Accuracy

Accuracy measures the ratio of correctly predicted instances to the total number of instances. It provides a general idea of how often the model makes correct predictions.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

where TP is True Positives, TN is True Negatives, FP is False Positives, and FN is False Negatives.

8.3 Precision

Precision is the ratio of correctly predicted positive observations to the total predicted positives. It measures the model's accuracy when predicting the positive class.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{6}$$

8.4 Recall

Recall (also known as Sensitivity or True Positive Rate) is the ratio of correctly predicted positive observations to all observations in the actual positive class. It measures the model's ability to detect all positive instances.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{7}$$

8.5 F1 Score

The F1 Score is the harmonic mean of Precision and Recall. It provides a balanced metric that considers both false positives and false negatives.

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{8}$$

Each of these metrics provides valuable information about the model's performance, and together they offer a comprehensive evaluation of the model's strengths and weaknesses.

8.6 Confusion Matrices for Comparative Experiments

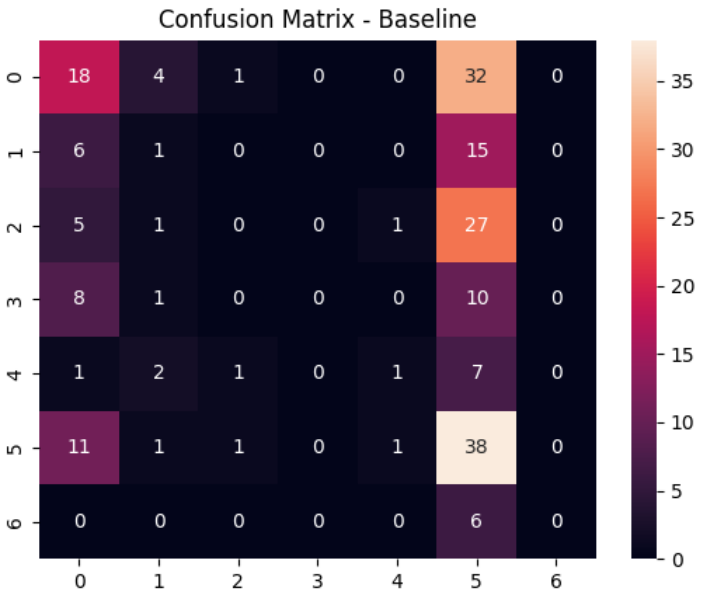

Figure 9: Confusion Matrix - Baseline SVM

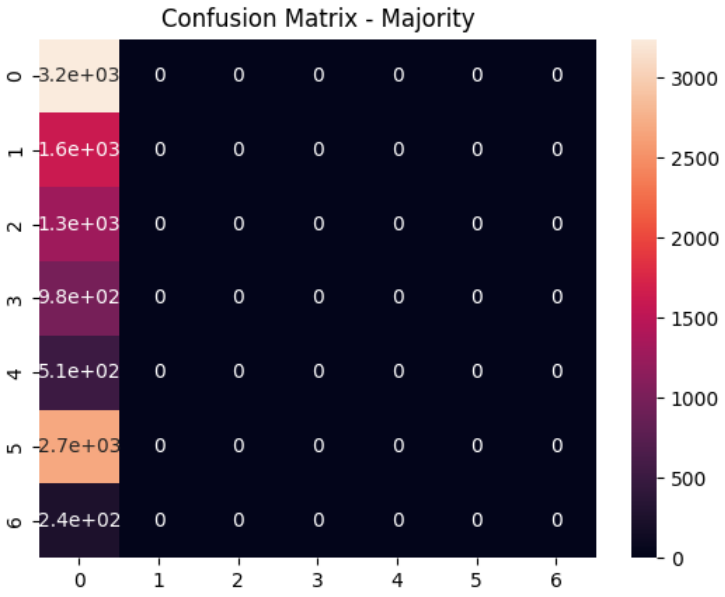

Figure 10: Confusion Matrix - Majority

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. Tensorflow: Large-scale machine learning on heterogeneous systems. `https://www.tensorflow.org/`.

Leo Breiman. 2001. Random forests. *Machine Learning*, 45(1):5–32.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. 2021. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9650–9660.

Emilio Chuvieco, Inmaculada Aguado, S. Jurdao, M. Lucrecia Pettinari, Marta Yebra, Javier Salas, Stijn Hantson, Juan Riva, Paloma Ibarra, Marcos Rodrigues, M. Echeverría, Diego Azqueta, María Román, Aitor Bastarrika, Susana Martinez, Carmen Recondo, E. Zapico, and Javier Martinez-Vega. 2014. Integrating geospatial information into fire risk assessment. *International Journal of Wildland Fire*, 23:606–619.

Rafik Ghali and Moulay A. Akhloufi. 2023. Deep learning approaches for wildland fires using satellite remote sensing data: Detection, mapping, and prediction. *Fire*, 6(5).

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, koray kavukcuoglu, Remi Munos, and Michal Valko. 2020. Bootstrap your own latent - a new approach to self-supervised learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 21271–21284. Curran Associates, Inc.

Shunjie Han, Cao Qubo, and Han Meng. 2012. Parameter selection in svm with rbf kernel function. In *World Automation Congress 2012*, pages 1–4.

Charles R Harris, K Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H van Kerkwijk, Matthew Brett, Allan Haldane, Jaime F del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E Oliphant. 2020. Array programming with numpy. *Nature*, 585:357–362.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. 2021. Masked autoencoders are scalable vision learners.

J. MacQueen. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press.

Sandra Oliveira, Jorge Rocha, and Ana Sá. 2021. Wildfire risk modeling. *Current Opinion in Environmental Science Health*, 23:100274.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Mathieu Brucher, Matthieu Perrot, and Edouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Shuchang Shen, Sachith Seneviratne, Xinye Wanyan, and Michael Kirley. 2023. Firerisk: A remote sensing dataset for fire risk assessment with benchmarks using supervised and self-supervised learning.

Maria João Sousa, Alexandra Moutinho, and Miguel Almeida. 2020. Wildfire detection using transfer learning on augmented datasets. *Expert Systems with Applications*, 142:112975.

Michael L. Waskom. 2021. Seaborn: Statistical data visualization. *Journal of Open Source Software*, 6(60):3021.

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.