VIT
Vellore Institute of Technology

School of Advanced Sciences

Fall Semester 2025-26

Department of Mathematics

PMDS601L - Artificial Intelligence

Course Project

Solar Flare Class Prediction using Machine Learning Algorithms

Gargi Rahul Panchabhai - 25MDT0100

Faculty In-charge

Dr. Prakash M

Table of content

Abstract

This study presents a comparative analysis of machine learning algorithms for solar flare classification using principal component analysis (PCA) as a dimensionality reduction technique. The models evaluated include K-Nearest Neighbors (KNN), Random Forest, and XGBoost, each trained and tested on datasets reduced to 7 and 55 principal components. Model performance was assessed using accuracy, ROC AUC, PR AUC, and F1 Score metrics. The results indicate that increasing the number of principal components improves predictive performance across all models, with XGBoost achieving the highest overall accuracy and robustness, particularly with 55 components. KNN demonstrated strong performance in low-dimensional spaces, while Random Forest maintained consistent accuracy across configurations. The findings highlight the importance of selecting appropriate dimensionality reduction levels and model architectures to balance interpretability and performance. The study contributes to the development of efficient solar flare prediction frameworks and provides insights for optimizing machine learning approaches in astrophysical data analysis. Future work will focus on refining model hyperparameters and integrating real-time solar observation data to enhance prediction accuracy and reliability in operational space weather forecasting systems.

Keywords: Solar flare prediction, machine learning, principal component analysis (PCA), dimensionality reduction, K-Nearest Neighbors (KNN), Random Forest, XGBoost, feature engineering, classification, space weather forecasting

Introduction

A solar flare is an intense burst of radiation coming from the release of magnetic energy associated with sunspots. Flares are our solar system's largest explosive events. They are seen as bright areas on the sun and they can last from minutes to hours. We typically see a solar flare by the photons (or light) it releases, at most every wavelength of the spectrum. The primary ways we monitor flares are in x-rays and optical light. Flares are also sites where particles (electrons, protons, and heavier particles) are accelerated. [1] Scientists classify solar flares according to their X-ray brightness, in the wavelength range 1 to 8 Angstroms. Flares classes have names: A, B, C, M, and X, with A being the tiniest and X being the largest. Each category has nine subdivisions ranging from, e.g., C1 to C9, M1 to M9, and X1 to X9. These are logarithmic scales, much like the seismic Richter scale. So an M flare is 10 times as strong as a C flare. SID space weather monitors can pick up X-class, M-class, and a few strong C-class flares. [2]

Class	*Strength- Peak (W/m^2) between 1 and 8 Angstroms*	*What can they do to Earth?*	*Can SIDS pick up?*
B	$I \leq 10\text{-}6$	Too small to harm Earth.	No, too weak for SID
C	$10\text{-}6 \leq I < 10\text{-}5$	Small with few noticeable consequences on Earth.	SIDs can pick up only the strongest C class flares
M	$10\text{-}5 \leq I < 10\text{-}4$	Can cause brief radio blackouts that affect Earth's polar regions and minor radiation storms.	Yes
X	$I \geq 10\text{-}4$	Can trigger planet-wide radio blackouts and long-lasting radiation storms	Yes

Table 1: Solar Flare Classes

Literature Survey

Solar flares are sudden releases of magnetic energy from the Sun's active regions that can cause significant disruptions to satellite communication, navigation systems, and power infrastructure on Earth. The problem central to solar flare class prediction lies in the inherent complexity and stochastic nature of flare occurrences. The nonlinear interactions between magnetic field parameters and the limited availability of strong-class (M and X) flares make it difficult to build reliable predictive models. Machine learning (ML) algorithms have emerged as effective tools for learning these nonlinear relationships from large observational datasets such as NASA's *Space-weather Helioseismic and Magnetic Imager Active Region Patches (SHARP)*, enabling automated classification of solar flares into Geostationary Operational Environmental Satellite (GOES) classes (B, C, M, and X). The following studies demonstrate progressive advancements in ML-based solar flare forecasting, differing in methodology, feature representation, and learning algorithms.

Wang et al. (2020) employed a Long Short-Term Memory (LSTM) network to predict whether an active region would produce a flare of a given class (≥M, ≥C, or any flare) within 24 hours. Their approach focused on exploiting temporal dependencies in solar active region evolution by feeding 24-hour time sequences of twenty magnetic parameters from the SHARP dataset into a two-layer LSTM model. The study covered data from 2010 to 2018 and demonstrated that incorporating time-series information significantly improved predictive skill compared to static classifiers. However, they also revealed that model accuracy and True Skill Statistic (TSS) scores varied across solar cycles, with better performance observed during the declining phase (2015–2018) than during the solar maximum (2011–2014). This highlighted the solar cycle dependence of model reliability and the importance of temporal context in flare prediction.

Sinha et al. (Sinha, 2022) conducted one of the most comprehensive comparative analyses of classical machine learning models for solar flare forecasting, focusing on identifying the most effective magnetic field indicators. Using data from the entire Solar Cycle 24 (2010–2020) and combining GOES and Hinode flare catalogs, they trained and compared Logistic Regression (LR), Support Vector Machine (SVM), Random Forest (RF), and k-Nearest Neighbors (KNN) algorithms. Through ANOVA F-statistics, they selected fourteen high-impact features out of nineteen SHARP magnetic parameters, including newly derived twist-related quantities such as total absolute twist (TOTABSTWIST). Their results showed that Logistic Regression and SVM achieved the highest TSS values (≈0.97), outperforming other algorithms in distinguishing flaring and non-flaring active regions. The study also introduced two novel evaluation indices—the Severe Space Weather (SSW) and Clear Space Weather (CSW) indicators, to better quantify forecasting outcomes. Overall, this work established a strong empirical baseline for parametric ML methods and clarified which SHARP parameters most strongly correlate with flare productivity.

Bringewald (2025) advanced this research by directly comparing Random Forest, KNN, and Extreme Gradient Boosting (XGBoost) for both binary and multiclass flare classification tasks using the SHARP dataset compiled by Liu et al. (2017). The study uniquely incorporated Principal Component Analysis (PCA) for dimensionality reduction, experimenting with eight and one hundred principal components to capture

95% and 97.5% of the data variance, respectively. By combining grid search optimization with 10-fold stratified cross-validation, the research demonstrated that ensemble methods, particularly Random Forest and XGBoost—consistently outperformed KNN in terms of classification accuracy, precision, and recall. Furthermore, the analysis revealed that model performance improved with higher dimensional representations, indicating that including more components preserved crucial variance in magnetic field data. Bringewald's methodology successfully bridged binary and multiclass flare prediction, offering a robust comparative framework for astrophysical classification problems.

In summary, these studies collectively establish the growing effectiveness of ML approaches for solar flare forecasting. Wang et al. (2020) emphasized the importance of capturing temporal dynamics using deep learning architectures, Sinha et al. (2022) demonstrated the interpretability and strong performance of classical ML models on static SHARP features, and Bringewald (2025) optimized model performance through feature reduction and ensemble methods. Building upon this foundation, the present project replicates the approach proposed by Bringewald (2025) but aims to enhance it by using a smaller set of highly influential magnetic parameters while achieving superior model performance through refined feature engineering and improved ensemble learning techniques. This approach seeks to balance model simplicity with predictive accuracy, contributing toward more efficient and operationally viable solar flare forecasting systems.

Preliminaries

Study area:

The goal of this project is to develop a machine learning model that accurately classifies solar flares into their GOES categories (C, M, X) based on magnetic parameters derived from the SHARP dataset. The challenge lies in managing class imbalance, reducing feature redundancy, and optimizing model performance while maintaining computational efficiency. This study replicates the comparative framework of Bringewald (2025) but focuses on achieving improved performance using fewer, highly informative parameters through enhanced feature selection and optimized ensemble learning algorithms.

Material:

Computational Tools and Libraries

This project was implemented using the Python programming language within the Visual Studio Code (VS Code) environment, utilizing the Jupyter extension for interactive model development and visualization. Python was selected for its strong ecosystem of data science and machine learning libraries. Core libraries employed include NumPy and Pandas for data manipulation and numerical computation, Matplotlib and Seaborn for visualization, and Scikit-learn for preprocessing, model evaluation, and implementation of classical algorithms such as K-Nearest Neighbors (KNN) and Random Forest. The XGBoost library was used for gradient boosting–based classification. Additional modules such as sklearn.metrics and sklearn.utils were applied for resampling, performance evaluation (F1 score, ROC AUC, Precision-Recall AUC), and confusion matrix analysis. All computations and visualizations were performed interactively in VS Code, ensuring reproducibility and clarity of results.

Computing Hardware

The experiments were executed on an Apple MacBook equipped with an M4 processor, 16 GB of unified memory, and 550 GB of SSD storage, running macOS Sonoma. The M4 chip's integrated Neural Engine and GPU cores provided hardware acceleration for TensorFlow operations and efficient parallel computation during model training. This configuration offered sufficient memory bandwidth and processing capability to handle data preprocessing, multi-class classification, and visualization tasks smoothly. The system's performance ensured efficient execution of Python-based machine learning workflows without the need for external GPU resources.

Data Description

For this study, a pre-processed solar flare dataset derived from Space-weather HMI Active Region Patches (SHARP) data was employed. The SHARP dataset, developed by the SDO/HMI team[6], provides vector magnetic field measurements and derived parameters for automatically detected and tracked Active Regions (ARs) on the solar surface. To further support the analysis of AR dynamics, additional Lorentz force estimations from the CGEM Lorentz data series (Fisher et al) were incorporated.

The version of the dataset used in this work was originally compiled by Liu et al. (2017), covering solar activity over a 6.5-year period from May 2010 to December 2016, corresponding to the peak of Solar Cycle 24. Flare records were obtained from the GOES X-ray flare catalog provided by the National Centers for Environmental Information (NCEI) and were cross-referenced with USAF Solar Observing Optical Network (SOON) H-alpha flare listings (Denig et al., 2012) to ensure accurate flare identification and classification.

In this study, only C-, M-, and X-class flares were considered, following the GOES flare classification scheme, which categorizes flares based on their peak X-ray flux intensity. To maintain high data quality, only active regions within ±70° of the central meridian were selected to minimize projection effects and ensure reliable magnetic field measurements.

Unlike the original SHARP dataset, which includes 13–14 magnetic field parameters, the dataset used in this work consists of 10 key magnetic features derived from SHARP parameters, representing the most relevant magnetic and geometric properties influencing solar flare occurrence. These features serve as the input variables for training and evaluating machine learning models aimed at predicting the flare intensity class.

Parameter	Description
TOTUSJH	Total unsigned current helicity in G^2/m
TOTPOT	Total photospheric magnetic energy density in ergs per cubic centimeter
TOTUSJZ	Total unsigned vertical current, in Amperes
SAVNCPP	Sum of the Absolute Value of the Net Currents Per Polarity in Amperes
USFLUX	Total unsigned flux in Maxwells
AREA_ACR	Area of strong field pixels in active region (AR)
MEANPOT	Mean photospheric excess magnetic energy density in ergs per cubic centimeter
R_VALUE	Sum of flux near polarity inversion line
SHRGT45	Fraction of area with shear > 45 degree

Table 2: SHARP parameters description

Data Preprocessing and Feature Engineering

The preprocessing phase of this study involved transforming the raw SHARP magnetic parameters into a standardized and model-ready format. Initially, the dataset was cleaned by replacing comma decimal separators with dots and converting all feature columns to numeric types to ensure computational consistency. Missing values were imputed with zeros to maintain dataset completeness. Subsequently, all 13 selected SHARP magnetic parameters were standardized using the StandardScaler technique. Standardization was an essential step since the Principal Component Analysis (PCA) algorithm assumes that input features have comparable scales and are normally distributed. This transformation ensured that each feature contributed equally to the dimensionality reduction process.

After standardization, interaction features were generated to capture potential non-linear relationships among the magnetic parameters. These interaction terms, created using polynomial feature expansion in interaction-only mode, represent pairwise combinations of features that could reveal latent dependencies not visible in the original dataset. The covariance matrix of these interaction features, visualized in Figure 2, provided insights into how the parameters vary together and indicated strong inter-feature relationships within certain subsets of magnetic parameters. Introducing interaction terms also helped improve the model's generalization ability by capturing a more comprehensive structure of the underlying data patterns.

To reduce redundancy and simplify the high-dimensional interaction space, PCA was employed as a dimensionality reduction technique. PCA identifies the orthogonal linear

combinations of features (principal components) that capture the maximum variance in the dataset. The number of components retained was determined based on a variance threshold criterion, ensuring that the transformed dataset preserved the majority of the original information content. In this study, a cumulative variance threshold of 95% was selected, which corresponded to seven principal components (7PC) required to effectively represent the dataset. To assess the effect of higher variance retention, 55 principal components (55PC) were also analyzed, capturing approximately 95% of the variance. The explained variance ratio plot (Figure 2) illustrates how these components cumulatively account for increasing proportions of the dataset's variability.

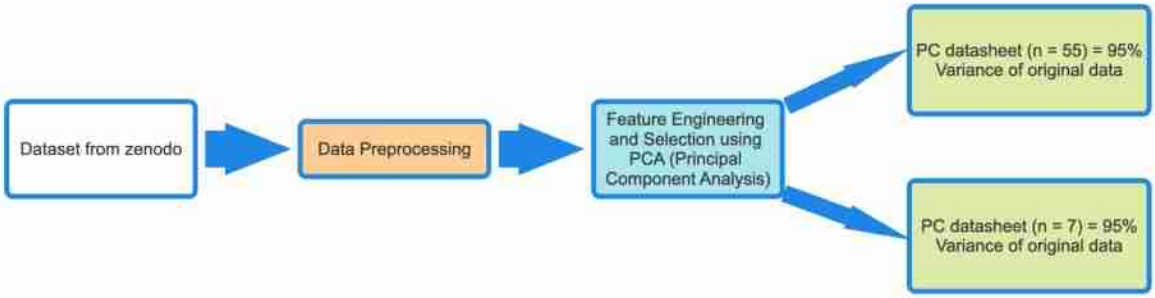

Figure 1: Visualisation of Feature engineering process

Through this structured preprocessing and feature engineering pipeline consisting of data standardization, interaction term creation, covariance analysis, and PCA-based dimensionality reduction, the dataset was transformed into a compact, information-rich representation. This transformation not only enhanced model interpretability but also improved computational efficiency and reduced the risk of overfitting during subsequent classification model training.

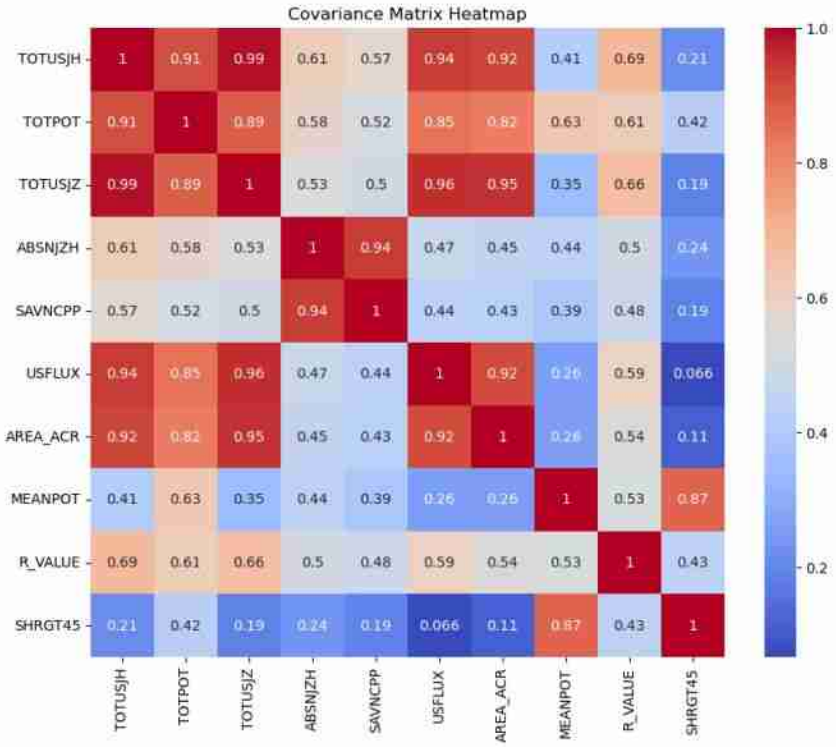

Figure 2: Covariance heatmap of magnetic parameters. Note: A covariance heatmap is a visual representation of the covariance matrix of a dataset's features. Covariance measures the strength of joint variability between two or more variables, as to say how much two variables change together. Positive covariance indicates that as one variable increases, the other tends to increase as well. Negative covariance indicates that as one variable increases, the other tends to decrease.

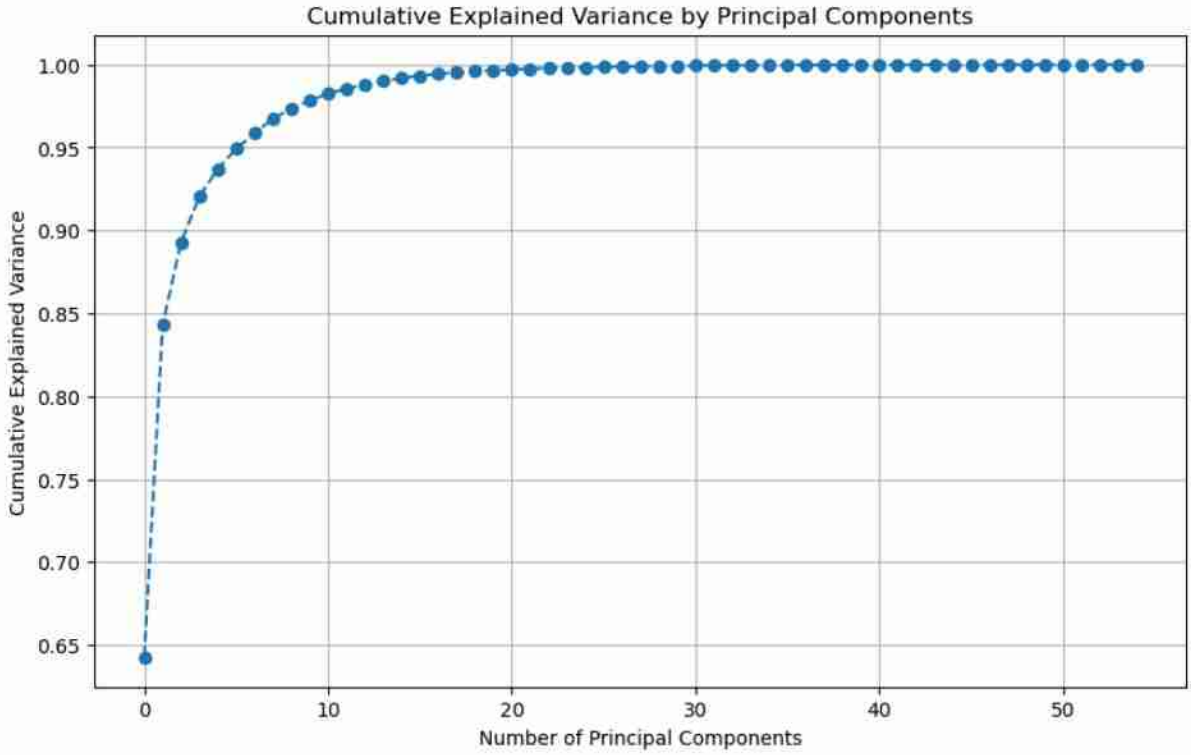

Figure 3: Cumulative Explained Variance by Principal Components (Interaction features) (Variance= 95, n = 7)

Model Training and Evaluation

K-Nearest Neighbours (KNN)

The K-Nearest Neighbours (KNN) algorithm is a supervised learning method introduced by Cover and Hart in 1967. It works on the principle of similarity, where predictions are made based on the closest data points in the feature space. In classification tasks, KNN assigns a class label to a new instance according to the majority class among its K nearest neighbours. In multi-class situations, this process is referred to as plurality voting. The algorithm's simplicity and interpretability make it widely used in various domains.

However, KNN can be computationally demanding since it requires recalculating distances between the input and all data points whenever new data is introduced. The main hyperparameters of KNN include the number of neighbours (K), the weighting scheme (whether all neighbours are equally important or closer ones have greater influence), and the distance metric used to measure similarity.

Random Forest Classifier

The Random Forest (RF) algorithm, developed by Leo Breiman and Adele Cutler, is a widely used ensemble learning method known for its effectiveness and adaptability. It combines the results of multiple decision trees to produce a single outcome, making it suitable for both classification and regression problems. During training, each tree is built using a bootstrap sample of the dataset, where around 37 percent of instances may be repeated. At each node split, a random subset of input features is chosen to determine the best division. This combination of bootstrap sampling and feature randomness helps reduce overfitting and enhances model stability. For classification, the final prediction is made through majority voting among all trees in the forest.

Extreme Gradient Boosting (XGBoost)

Extreme Gradient Boosting (XGBoost) is an open-source, distributed machine learning library designed for high speed, scalability, and accuracy. It is based on the gradient boosting framework, which builds an ensemble of decision trees sequentially. The process begins with a weak learner, known as the base learner, and iteratively adds new trees that correct the errors of the previous ones. The algorithm calculates residuals, representing the difference between predicted and actual values, and uses them to guide the next tree's construction. Model performance is evaluated using loss functions such as mean squared error for regression or cross-entropy loss for classification. Gradient descent optimization minimizes these losses, improving the model's accuracy over successive iterations. XGBoost's ability to handle large datasets efficiently while maintaining strong predictive performance makes it one of the most preferred algorithms in modern machine learning.

In simpler words: XGBoost begins with a simple model that makes basic predictions. Then, it adds new models (decision trees) in a step-by-step process, each one learning from the mistakes of the previous models. These trees try to fix the errors made by earlier predictions. The predictions from all the trees are combined to make a final, more accurate prediction. The algorithm uses a technique called gradient descent to adjust and minimize the errors, making the model better with each new tree added.

Training and Evaluation

For the flare prediction task, all algorithms were implemented in classification mode. During the training phase, each input sample consisted of interaction parameter values (n=7 or n=55) of an active region (AR) and its corresponding maximum GOES flare class, which served as the target variable. In the testing or validation phase, these interaction parameters were used as model inputs, and the classifier predicted the GOES flare class of the given AR.

The dataset used in this study, derived from Liu et al. [2017b], included three flare classes C, M, and X with 7560 C-class, 3440 M-class, and 440 X-class samples. This significant class imbalance posed a challenge for model training, as classifiers tend to favor the majority class when trained on unevenly distributed data. To mitigate this issue, a balanced resampling approach was adopted. Equal-sized subsets from each class were randomly selected to form balanced datasets for classification. This random selection process was repeated multiple times to ensure robustness and minimize bias. The resulting datasets provided a fair representation of each flare class and were used for training and evaluating all machine learning models.

To evaluate the performance of various machine learning algorithms, the widely used 10-fold cross-validation (CV) method is applied. For each dataset, stratified 10-fold partitioning is performed using StratifiedKFold(), ensuring that each fold contains nearly equal-sized groups with balanced class distributions. The algorithm is then trained using nine folds of the data, while the remaining fold is used for validation. This process is repeated for each combination of hyper-parameters being evaluated using the GridSearch() function. Hyperparameters are settings that control the learning process and are not learned from the data. Proper tuning helps improve model accuracy, reduce overfitting, and ensure that the model generalizes well to new and unseen data. During the cross-validation, the model's performance is assessed using predefined metrics such as accuracy, ROC AUC, F1 score, and precision-recall AUC . The indepth functioning of the performance metrics is described as follows.

$$\begin{aligned}
\textbf{Overall Accuracy} &= \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}+\mathrm{FN}+\mathrm{TP}} \\
\textbf{F1 score} &= \frac{2\times\mathrm{TP}}{2\times\mathrm{TP}+\mathrm{FP}+\mathrm{FN}} \\
\\
\mathrm{Precision} &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \\
\mathrm{Recall} &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \\
\textbf{PR AUC} &= \int_0^1 \mathrm{Precision}\, d(\mathrm{Recall}) \\
\\
\mathrm{True\ Positive\ Rate\ (TPR)} &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \\
\mathrm{False\ Positive\ Rate\ (FPR)} &= \frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TN}} \\
\textbf{ROC AUC} &= \int_0^1 \mathrm{TPR}\, d(\mathrm{FPR})
\end{aligned}$$

1. True positives (TP): number of samples correctly predicted as "positive".
2. False positives (FP): number of samples wrongly predicted as "positive".
3. True negatives (TN): number of samples correctly predicted as "negative".
4. False negatives (FN): number of samples wrongly predicted as "negative".

The performance metrics obtained from the 10 folds are averaged to produce the mean performance score for each hyperparameter configuration. This averaging reduces the

effect of data variability and ensures a more reliable evaluation. The hyperparameter combination that achieves the highest mean performance score across the 10 folds is identified as the optimal set. Standard evaluation metrics, including Receiver Operating Characteristic Area Under the Curve (ROC AUC), accuracy, precision-recall AUC, and F1 score, are computed to assess the effectiveness of each model. Higher AUC values, approaching 1, indicate stronger classification performance.

Hyperparameters:

Algorithm	Tuning Parameter	Search Grid
KNN	Number of Nearest Neighbours (n_neighbors)	[3,5,7,9]
	Metric of weights	['uniform', 'distance']
	Metric of distance	['euclidean', 'manhattan']
Random Forest	Number of features	[8,9,10,11,12,13]
	Depth of trees (min_samples_split)	[8, 10]
XGBOOST	Learning rate	[0.1, 0.9]
	Subsample	[0.01, 1]

Table 3: Algorithm Hyperparameter Search Grid

Notes on the hyperparameter search grid:

K-Nearest Neighbors (KNN):

Key hyperparameters include the number of neighbors, weighting method, and distance metric. The number of neighbors controls the model's bias-variance trade-off—fewer neighbors (e.g., 3) increase sensitivity to local patterns but may capture noise, while more neighbors (e.g., 10) produce smoother predictions. The weighting method defines how each neighbor contributes: *uniform* assigns equal influence, while *distance* gives more weight to closer points. The distance metric (e.g., *Euclidean* or *Manhattan*) determines how proximity is measured in the feature space.

Random Forest (RF):

Important hyperparameters include the number of features considered at each split, maximum tree depth, and minimum samples required to split a node. Increasing tree depth allows the model to learn complex patterns but can lead to overfitting. Setting a higher minimum sample split (e.g., 5) ensures reliable node splits and better generalization.

XGBoost:

Key hyperparameters include the learning rate and subsample ratio. A smaller learning rate improves accuracy but increases training time, while the subsample parameter controls the proportion of training data used for each tree, reducing overfitting by adding randomness.

Due to computational limitations, only a limited range of hyperparameter values was explored. All models (KNN, RF, and XGBoost) were implemented using the scikit-learn library (Pedregosa et al., 2011).

Model comparison and selection

	KNN		Random Forest		XGBoost	
	7 PC	55 PC	7 PC	55 PC	7 PC	55 PC
Accuracy	0.7233	0.7493	0.6993	0.735	0.7167	0.7491
ROC AUC	0.8633	0.8737	0.8479	0.8755	0.8547	0.8798
PR AUC	0.8099	0.8056	0.7768	0.8153	0.7967	0.8273
F1 Score	0.7179	0.7376	0.6917	0.7293	0.71195	0.7441

Table 4: Model performance for multiclass classification with n=7 and n=55

This table presents the performance metrics of three machine learning algorithms K-Nearest Neighbors (KNN), Random Forest, and XGBoost, for a multiclass classification task using two feature dimensions: 7 and 55 principal components (PCs) derived from PCA. Each model was evaluated using four standard metrics: Accuracy, ROC AUC, PR AUC, and F1 Score. Overall, performance improves slightly when increasing the number of principal components, with accuracy rising by about 2–4% across all models and modest gains observed in ROC AUC, PR AUC, and F1 Score. This indicates that retaining more components preserves additional information from the original features, leading to better model generalization and classification accuracy.

Among the models, KNN shows stronger results with fewer components (7 PCs), while XGBoost achieves the best overall performance with 55 PCs. Specifically, XGBoost records the highest ROC AUC (0.8798), PR AUC (0.8273), and F1 Score (0.7441), demonstrating its ability to handle complex, high-dimensional data effectively. Random Forest performs consistently across both feature dimensions but does not outperform KNN or XGBoost in any specific metric. In summary, increasing the number of principal components enhances performance for all models, with XGBoost emerging as the most robust and accurate model, while KNN performs best on simpler, lower-dimensional feature spaces.

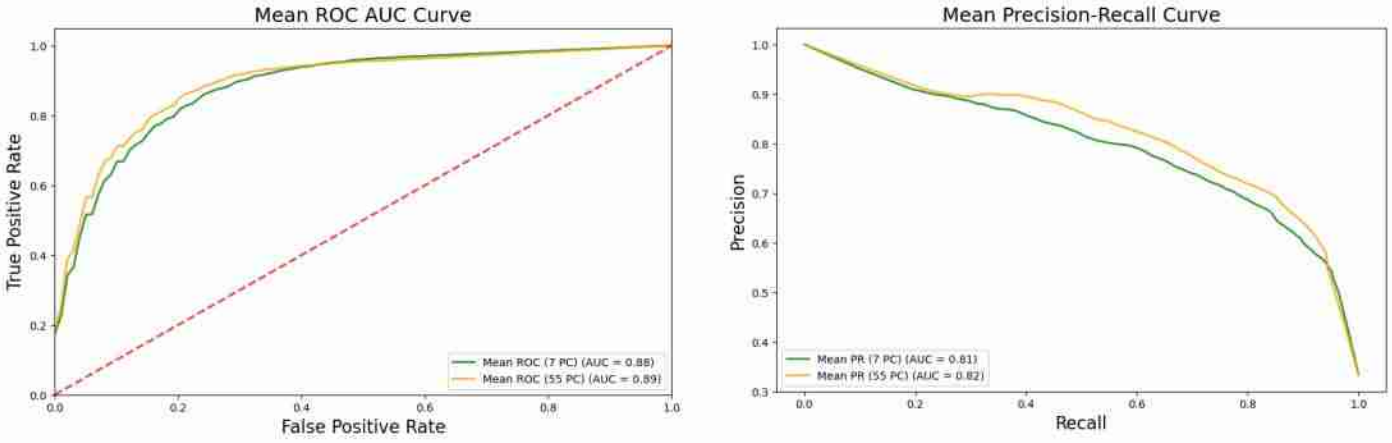

Figure 4: Performance curves for KNN

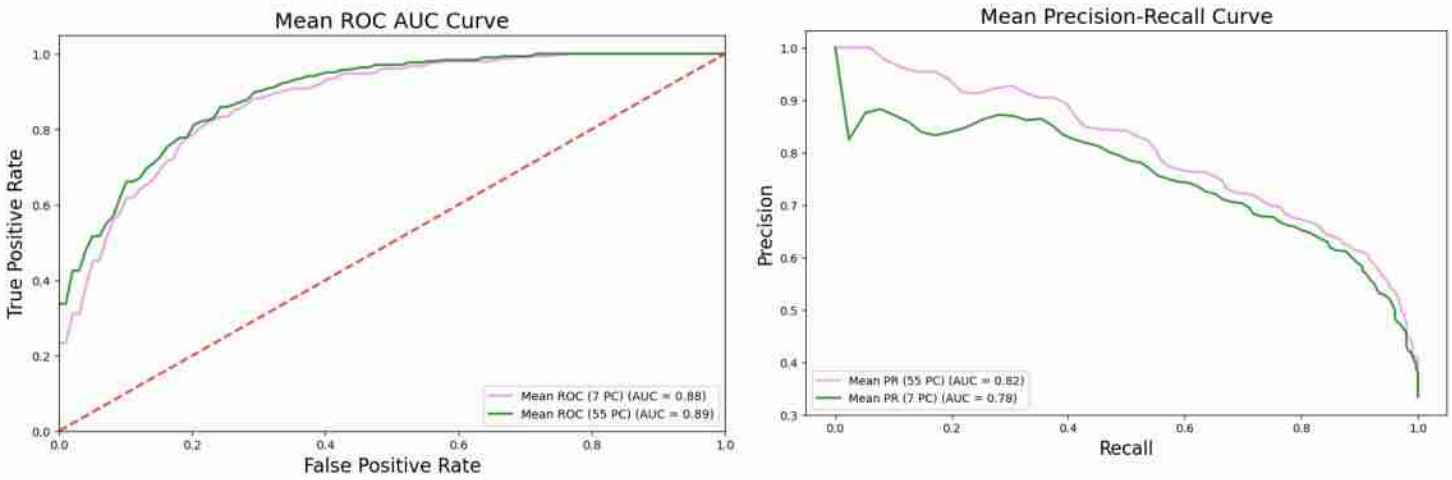

Figure 5: Performance curves for Radom forest

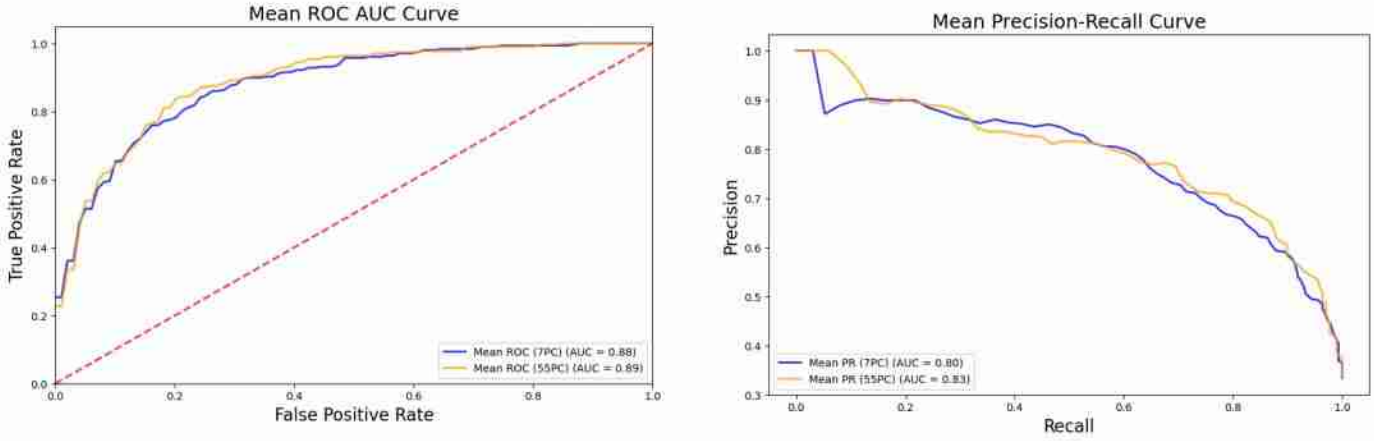

Figure 6: Performance curves for XGBoost

Conclusion and Future Scope

The study evaluated the performance of K-Nearest Neighbors (KNN), Random Forest, and XGBoost models for multiclass solar flare classification using PCA-reduced features. The results showed that model performance improved as the number of principal components increased from 7 to 55, indicating that retaining more components preserved valuable information from the original dataset. Among the models, XGBoost achieved the highest overall performance across ROC AUC, PR AUC, and F1 Score metrics, showing strong capability in capturing complex patterns and relationships in high-dimensional data. KNN performed best with fewer components, highlighting its efficiency in low-dimensional feature spaces, while Random Forest remained stable and competitive across different feature dimensions. These findings demonstrate that model performance is closely tied to the amount of feature information retained during dimensionality reduction and that XGBoost offers a balanced trade-off between accuracy and generalization for solar flare prediction tasks.

The study was limited by the narrow range of hyperparameters tested due to computational constraints, which might have prevented full model optimization. PCA was the only dimensionality reduction technique applied, possibly excluding non-linear feature relationships relevant to flare classification. The insights gained from this study can help improve solar flare prediction models, guide the selection of suitable dimensionality reduction methods, and inform future model selection strategies for astrophysical applications. In future work, a refined solar flare prediction model will be developed and deployed using the most effective hyperparameter combinations identified in this study. Additionally, validated models will be integrated with real-time solar observation data to strengthen the accuracy and responsiveness of space weather forecasting systems. This integration will support the creation of more reliable early-warning mechanisms for solar flare events, contributing to the protection of technological infrastructure and advancing research in space weather prediction.

Resources and References

1) Bringewald, J. (2025, May 02). *Solar Flare Forecast: A Comparative Analysis of Machine Learning Algorithms for Predicting Solar Flare Classes*. arXiv. arxiv.org/html/2505.03385v1
2) Predicting Solar Flares with Machine Learning: Investigating Solar Cycle Dependence. (2020, May 19). *The Astrophysical Journal*, *865*(1). 10.3847/1538-4357/ab89ac
3) Sinha, S. (2022, August 12). A Comparative Analysis of Machine-learning Models for Solar Flare Forecasting: Identifying High-performing Active Region Flare Indicators. *The Astrophysical Journal*, *935*. 10.3847/1538-4357/ac7955
4) *What are the different types, or classes, of flares?* (n.d.). Stanford Solar Center. Retrieved November 3, 2025, from https://solar-center.stanford.edu/sid/activities/flare.html
5) *What is a solar flare?* (2015, March 6). NASA. Retrieved November 3, 2025, from https://www.nasa.gov/image-article/what-solar-flare/
6) Bobra, M. G., and S. Couvidat. "SOLAR FLARE PREDICTION USING SDO/HMI VECTOR MAGNETIC FIELD DATA WITH A MACHINE-LEARNING ALGORITHM." *The Astrophysical Journal*, vol. 798, no. 2, 2015, https://iopscience.iop.org/article/10.1088/0004-637X/798/2/135.
7) Data Accessed from : Antanas, Žilinskas "Solar Flare Prediction Dataset: SHARP Magnetic Field Parameters for Machine Learning" link: https://zenodo.org/records/15570399
8) F. Pedregosa, G. Varoquaux, A. Gramfort, et al. Scikit-learn: Machine learning in python. Journal of Machine Learning Research, 12:2825, 2011.
9) Zhang, H. (2022). *Solar Flare Index Prediction Using SDO/HMI Vector Magnetic Data Products with Statistical and Machine-learning Methods*. IOPScience. https://iopscience.iop.org/article/10.3847/1538-4365/ac9b17
10) Source codes accessed from:

https://github.com/juliabringewald/Solar-Flare-Forecast