# A comparative study of Pattern Recognition models on the *PaviaU Dataset*

## Abstract

This report provides a comprehensive study focused on land cover classification and remote sensing image analysis using the *PaviaU dataset*. The report first introduces the basic characteristics and application background of the data set, and then discusses in detail the application of feature selection and dimensionality reduction techniques, especially linear discriminant analysis (LDA) and principal component analysis (PCA). In terms of model application, the report not only uses traditional machine learning models, such as support vector machines, Bayesian and KNN posterior probabilities, but also explores the performance of these models under different parameter settings. Through a series of experiments, the study found that optimizing feature selection and dimensionality reduction techniques can significantly improve the classification accuracy of the model. Finally, the report compares different models and suggests alternative improvements.

## Contents

# 1. Dataset

## 1. 1 Dataset Description

*PaviaU* is one of the two datasets obtained from Pavia, Northern Italy using the ROSIS sensor [1].

The *PaviaU* dataset captures the scene acquired by the *ROSIS* sensor during a flight campaign over Pavia, Northern Italy. It features a total of 103 spectral bands for Pavia University, which is originally 610x610 pixels in size. However, some samples within the images contain no information and need to be removed prior to analysis. It is particularly valuable for tasks such as urban land cover classification [2].

Pavia scenes were provided by Prof. Paolo Gamba from
the Telecommunications and Remote Sensing
Laboratory, University (Italy)

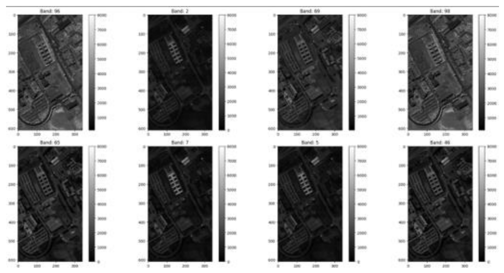
The PaviaU dataset is mainly utilized for research in remote sensing image processing and computer vision, as well as for land cover classification and remote sensing image analysis. It provides high spatial and spectral resolution to enable detailed studies of the Earth's surface, vegetation monitoring, and object recognition.

In addition to the hyperspectral image, the dataset is accompanied by a ground truth map of dimensions 610x340 pixels which differentiates 9 distinct classes. It is used to label different land cover categories within the image：

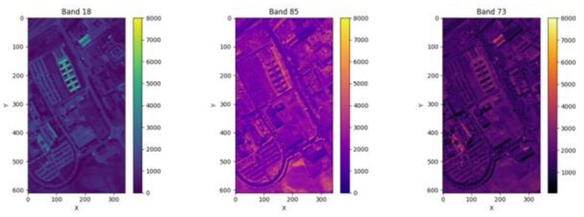
| # | Class | Samples |
|---|---|---|
| 1 | Asphalt | 6631 |
| 2 | Meadows | 18649 |
| 3 | Gravel | 2099 |
| 4 | Trees | 3064 |
| 5 | Painted Metal Sheets | 1345 |
| 6 | Bare Soil | 5029 |
| 7 | Bitumen | 1330 |
| 8 | Self-Blocking Bricks | 3682 |
| 9 | Shadows | 947 |

## 1.2 Feature Visualization

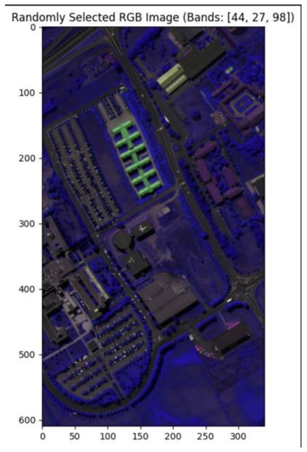
Here's a visualization of 8 different randomly selected bands:



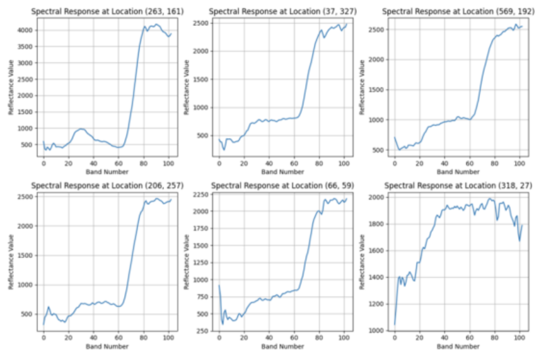
Next, we concentrate on a monochromatic picture presented in a color spectrum. To improve contrast and separation, we have applied color mapping to three bands that were chosen at random in this instance. This method helps with the correlation between the image and the ground truth data by offering a more illuminating representation:



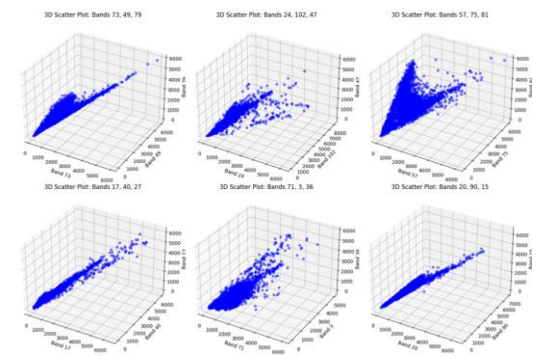
Furthermore, to explore the data in a different way, we performed a random selection of three bands. We then combined these bands into RGB channels, creating a composite image that allows us to examine specific aspects of the data in a more visually intuitive manner:



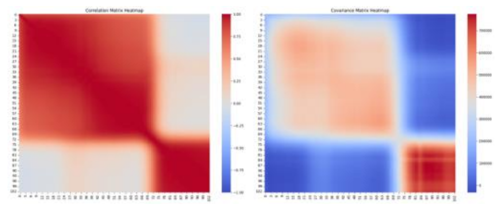
In addition to the composite RGB image, we also delved into the spectral characteristics of the data. For this purpose, we randomly selected six different locations and plotted their spectral response curves. This insightful approach aids in understanding the unique spectral characteristics of various materials present in the scene, offering valuable insights into the composition of the observed areas.:



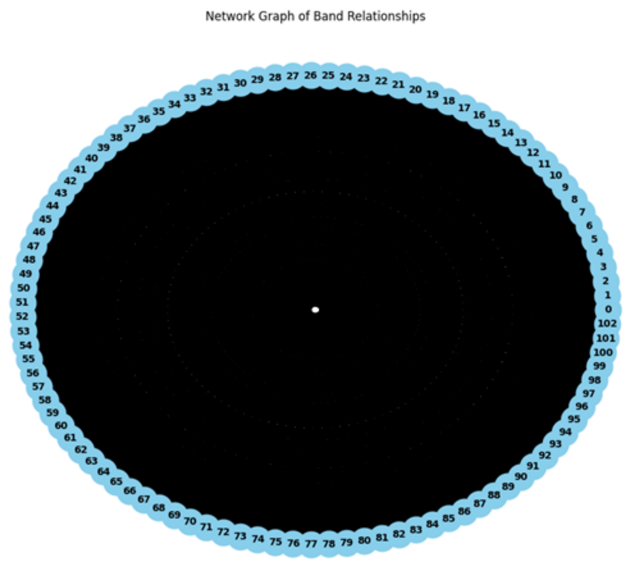
To gain a deeper understanding of the data, we've created a 3D scatter plot in feature space. In this visualization, data points are displayed in three-dimensional space, where each dimension corresponds to a different wavelength. This representation provides an effective means to explore the distribution and relationships of data points based on their spectral characteristics:



Expanding our exploration, we've generated a heatmap as an additional visualization tool. This heatmap offers an intuitive representation of the correlation or covariance between different spectral bands. The color intensity within the heatmap corresponds to the magnitude of the correlation or covariance, providing a valuable insight into the relationships and dependencies among the various spectral bands:

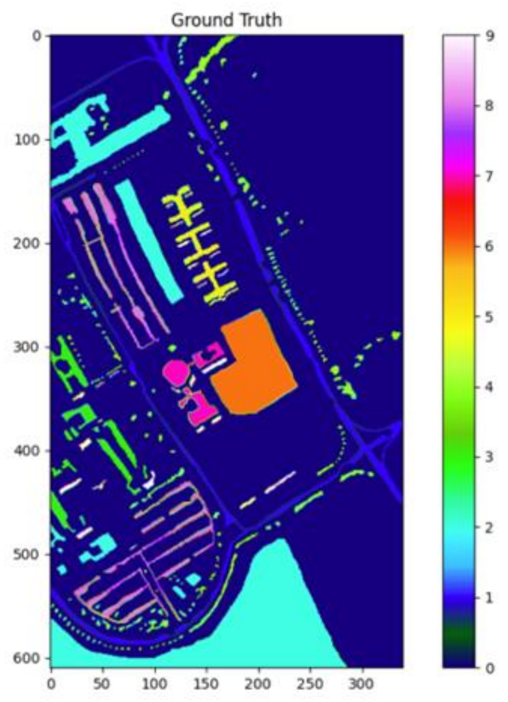


Expanding our visual analysis, we've created a network graph to illustrate the intricate relationships between spectral bands. In this graph, nodes represent specific spectral bands, and edges signify the correlations or covariances between these bands. The width or color of the edges can serve as visual indicators, conveying the magnitude of these correlations or covariances. This network graph provides a comprehensive overview of the interconnections between different spectral bands, aiding in the interpretation of the data:

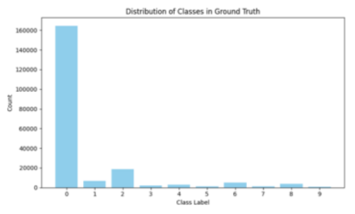


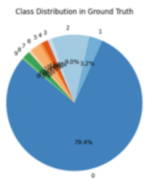
For the ground truth:

This is a sample of the ground truth map. It displays the land cover classification labels for each pixel in the hyperspectral image using different colors:



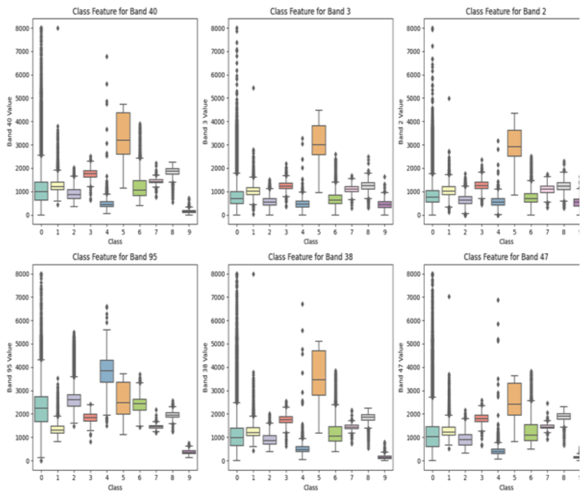
Moving forward, we delve deeper into the analysis by employing histograms and pie charts to visually illustrate the distribution of each category within the Ground Truth image. This method allows us to gain valuable insights into the prevalence of different land cover types across the scene:





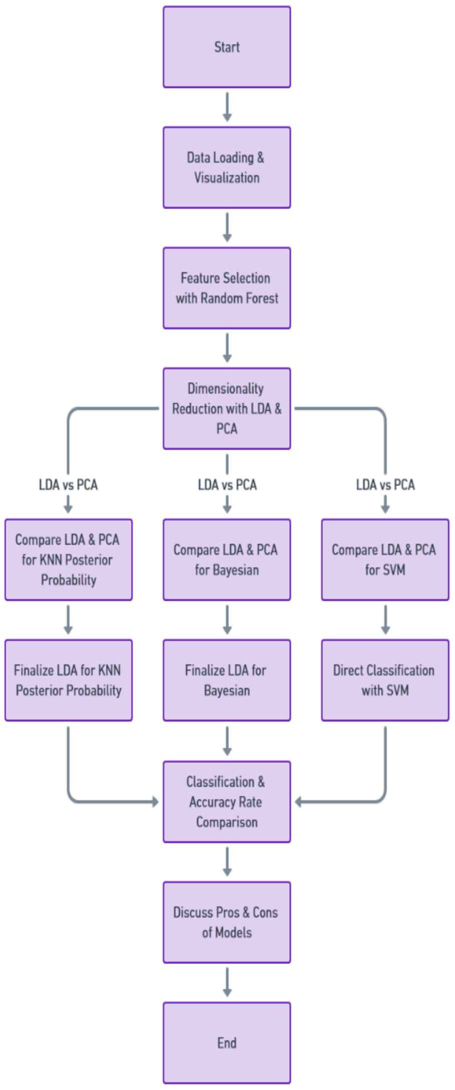
It is evident that the region labeled as 'class 0' in the Ground Truth image constitutes a significant portion, accounting for approximately 80% of the total data. Typically, 'class 0' represents the background or unlabeled areas within the image, i.e., regions that are not of interest or do not contain relevant objects or features. The choice of '0' for the background class is a common convention, signifying areas that are unassigned to any other specific class. Consequently, for our forthcoming classification task, we will exclude 'class 0' to create a more efficient and clearer model. By doing so, we aim to simplify the classification process and facilitate a more focused and meaningful evaluation of our model's performance.

In our comprehensive analysis of the PaviaU remote sensing image and its ground truth data, we take a step further in understanding the dataset. To achieve this, we combine the image data and ground truth data into a unified data frame. We then leverage this merged dataset to create a boxplot. This approach offers a powerful means to comprehend the variations in data distribution among different land cover classes, detect possible outliers or anomalies, and extract valuable insights regarding the overall distribution of the data. By integrating image data and ground truth information, we gain a more comprehensive understanding of the dataset's characteristics:



# 2. Implementation

## 2.1 Workflow


Start
Data Loading & Visualization
Feature Selection with Random Forest
Dimensionality Reduction with LDA & PCA
LDA vs PCA
LDA vs PCA
LDA vs PCA
Compare LDA & PCA for KNN Posterior Probability
Compare LDA & PCA for Bayesian
Compare LDA & PCA for SVM
Finalize LDA for KNN Posterior Probability
Finalize LDA for Bayesian
Direct Classification with SVM
Classification & Accuracy Rate Comparison
Discuss Pros & Cons of Models
End

## 2.2 Feature Selection

Why we need feature selection?

The dataset *PaviaU.mat* has 103 features, which is too large to make the model too complex and difficult to interpret. For faster training and prediction time, we must perform feature selection. At the same time, fewer features can help the risk of overfitting the training data.

1. Basic principles of Random Forest (RF)

We utilize the RF for selection features. RF build multiple decision trees, each of which is trained on a subset of the data and uses a random subset of features in its construction [3]. RF predictions are obtained by voting (classification problems) or averaging (regression problems) the predictions of all trees.

$$\mathrm{RF}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$$

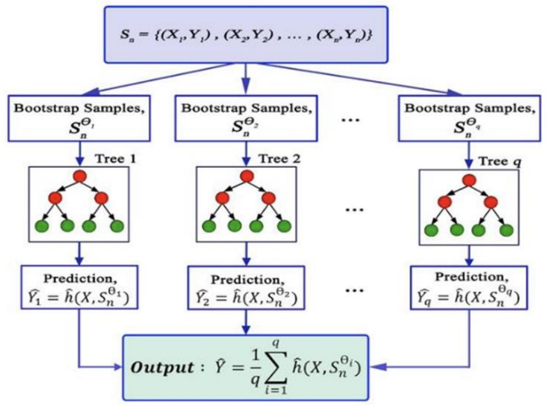
Where:
B is the number of the trees
Tb(x) is the prediction of sample x by the b$^{th}$ tree

2. Feature Selection

In addition to having good prediction accuracy, RF provides two feature selection methods: Mean Decrease Impurity (MDI) and Mean Decrease Accuracy (MDA). Our feature selection for *PaviaU.mat* is based on the MDI. The MDI score of a variable is calculated by summing the reduction in node impurity (usually called Variance), especially when feature X^j is used for data partitioning. According to Scornet (2020) [4], this impurity reduction is defined as:

$$\Delta V(t) = V(t) - p_L V(t_L) - p_R V(t_R)$$

When data is partitioned at node t, two child nodes are generated, denoted tL and tR, tL= NtL / Nt and PR= NtR / Nt represent the ratio of data arriving at child nodes tL and tR, respectively.
This figure below shows the structure of MDI method:


$S_n = \{(X_1,Y_1), (X_2,Y_2), \dots, (X_n,Y_n)\}$
Bootstrap Samples, $S_n^{\Theta_1}$
Bootstrap Samples, $S_n^{\Theta_2}$
Bootstrap Samples, $S_n^{\Theta_q}$
Tree 1
Tree 2
Tree q
Prediction, $\hat{Y}_1 = \hat{h}(X, S_n^{\Theta_1})$
Prediction, $\hat{Y}_2 = \hat{h}(X, S_n^{\Theta_2})$
Prediction, $\hat{Y}_q = \hat{h}(X, S_n^{\Theta_q})$
Output : $\hat{Y} = \frac{1}{q}\sum_{i=1}^{q} \hat{h}(X, S_n^{\Theta_i})$

In the case of the RF model, the score of MDI is acquired by averaging the scores of all q trees in the forest [5].

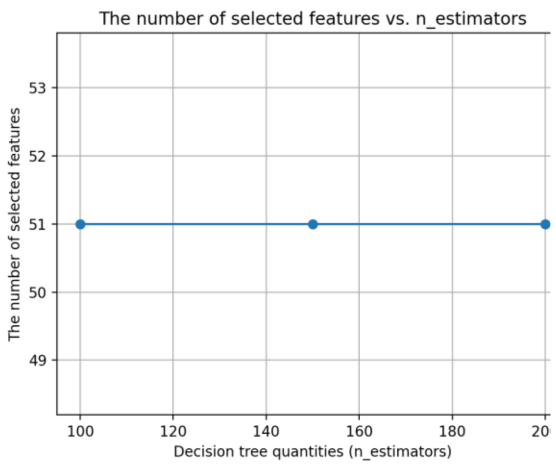
$$MDI = \frac{1}{q}\sum_{q}\sum_{t\epsilon q} p(t)\ \Delta V(t)$$

Where:
q is the number of trees
qP(t) is the node set of the b$^{th}$ tree

3. Algorithm Application

We iterate over different numbers of decision trees (n_estimators), from 100 to 200 with a step size of 50. For each number of decision trees (n_estimators), we create a random forest classifier and train it on the data. Then the feature importance is obtained and the median value is calculated as the threshold for feature selection. Select features based on a threshold, update the best number of features (best_num_features), the best number of decision trees (best_n_estimators), and a list that stores the number of selected features in each iteration (best_select_feature). Finally, we get the optimal n_estimator value and the number of selected features.

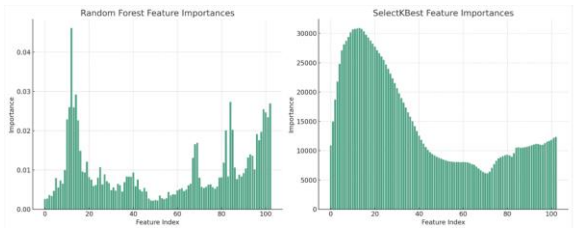
According to the figure of the number of selected features and decision tree below, we can see that the quantities of decision trees are from 100 to 200, the step size is 50, and the number of filtered features is 51.



By using the median feature importance as a threshold, we can determine that the filtered feature has the best value when the decision tree is 100, and the shape of the filtered data becomes (42776, 51).

4. Strengths of Random Forest

We also use SelectKBest for feature selection and compare it with RF to illustrate the superiority of RF.



Advantages of Random Forest:

(1) Model complexity:

Random forest can handle complex data sets and non-linear relationships, while SelectKBest is mainly suitable for linear models.

(2) Generalization ability

Random forest generally has better generalization ability because it is an ensemble method that combines the predictions of multiple decision trees.

(3) Multi-objective support:

Random Forest can be used for multi-classification problems, while SelectKBest requires feature selection for each category, which can be more complex.

(4) No extra steps required:

Random Forest performs feature selection naturally during training, while SelectKBest requires an extra step to select features.

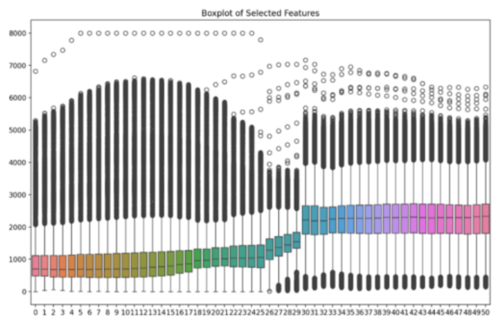
5. Investigate and experiment on the data

In order to analyze why the eigenvalue is optimal when the decision tree is 100, we calculated the mean, standard deviation and variance of these 51 eigenvalues.

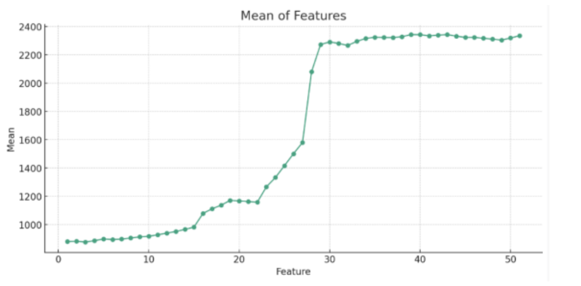
| Feature | Mean | Standard Deviation | Variance |
|---|---|---|---|
| 1 | 880.8961 | 660.1023 | 707.0 |
| 2 | 883.0966 | 685.5730 | 699.0 |
| 3 | 877.5243 | 726.7483 | 687.0 |
| 4 | 886.9314 | 751.6635 | 687.0 |
| 5 | 898.7503 | 775.7333 | 695.0 |
| 6 | 895.4971 | 789.0844 | 693.0 |
| 7 | 897.9141 | 805.6931 | 694.0 |
| 8 | 905.6715 | 820.4087 | 700.0 |
| 9 | 913.7094 | 830.2729 | 705.0 |
| 10 | 917.8967 | 836.2771 | 708.0 |
| 11 | 928.3730 | 842.4697 | 722.0 |
| 12 | 940.2070 | 845.0398 | 739.0 |
| 13 | 951.9673 | 842.5662 | 755.0 |
| 14 | 966.5009 | 838.6600 | 774.0 |
| 15 | 982.8947 | 832.2484 | 794.0 |
| 16 | 1079.6560 | 791.3762 | 899.0 |
| 17 | 1112.0332 | 786.2305 | 932.0 |

(Only show part of the content)

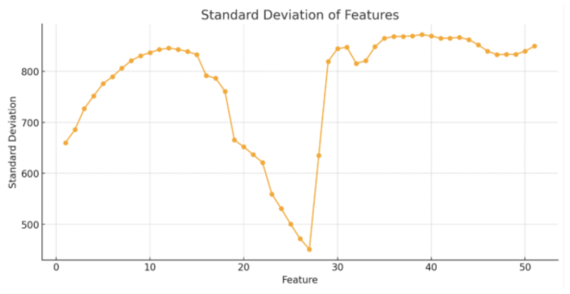
At the same time, we employ the box plot to better observe the differences between these 51 eigenvalues, and we analyze these 51 eigenvalues from the mean, standard deviation, and variance respectively.



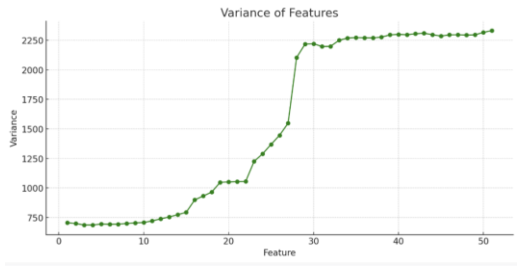
**Mean:**
Lower range: Features 1 to 5 have relatively low average values.
Midrange: Features 6 to 24 have gradually increasing average values.
Higher range: Features 25 to 51 have higher averages, with feature 30 showing a significant jump.



**Standard deviation:**
Increasing trend: The standard deviation generally increases from feature 1 to feature 9.
Downtrend: From feature 15 to feature 25, the standard deviation decreases slightly.
Lower variability: Features 26 to 29 have lower standard deviations, indicating a smaller spread around the mean.
Higher variability: Features 30 to 51 have higher standard deviations, indicating a larger spread around the mean.



**Variance:**
General trend: Variance values seem to increase with more features but with some fluctuations.



**Conclusion:**
The data shows a trend of increasing mean values, features can be divided into different groups based on their statistical properties, and there are clear differences between them.

## 2.3 Dimensionality Reduction

Why dimensionality reduction is needed?

Even after feature selection, the number of features may still be large, and dimensionality reduction can reduce the computational complexity. And helps remove noise and potentially improve model performance.

### 2.3.1 Linear Discriminant Analysis (LDA)

**Principal of LDA**
**Between-class variance:**
Between-class variance represents the difference between the means of different classes. The aim is to maximize this spread, making the classes as different as possible [8].

$$S_B = \sum_{i=1}^{C} N_i(\mu_i - \mu)(\mu_i - \mu)^T$$

Where:
C is the number of classes
Ni is the number of samples in class i
μi is the mean vector of class i
μis the overall mean vector of the data
(μi-μ) (μi-μ) ^T is a matrix that represents the spread of class i's mean from the overall mean

For each class, we consider the number of samples in that class and observe the distance between its mean and the overall mean of the data. We then sum these values across all classes. The result is a matrix $S_B$ that represents the between-class scatter.

**Within-class Variance:**
Within-class variance represents the distribution within each class. The purpose of this is to minimize the distribution so that samples in the same class are as close to each other as possible [8].

$$S_W = \sum_{i=1}^{C} \sum_{x \in C_i} (x - \mu_i)(x - \mu_i)^T$$

Where:
Ci is the set of samples belonging to class i
x is a sample vector
(μi-μ) (μi-μ) ^T is a matrix that represents the spread of sample x from the mean of

its class

For each class, we observe the distance of each sample from the mean of the class and then sum the values of all samples in the class. We then sum these values across all classes. The result is a matrix Sw that represents the within-class scatter.

**The objective of LDA:**
The goal of LDA is to find a projection that maximizes inter-class variance and minimizes within-class variance. This is achieved by finding a matrix *W* that maximizes the following objective function [8]:

$$J(W) = \frac{\text{Between-Class Variance}}{\text{Within-Class Variance}} = \frac{W^T S_B W}{W^T S_W W}$$

Where:
W is the matrix containing the projection vectors
SB is the between-class scatter matrix
Sw is the within-class scatter matrix.

The optimal projection vector *W* can be found by solving the generalized eigenvalue problem:

$$S_B W = \lambda S_W W$$

**Dimensionality reduction with LDA:**
Using LDA can reduce the data to at most c-1 dimensions, where c is the number of classes. This is because LDA tries to find directions that maximize the separation between categories, and if there are c categories, there can be at most c−1 such directions [9].

In the following content, we will select the dimensionality reduction number of LDA with the highest accuracy through cross-validation.

**Summary:**
LDA (Linear Discriminant Analysis) is a statistical technique used for data classification and dimensionality reduction. Its core is to find the best projection of the data by maximizing the between-class variance and minimizing the within-class variance. Inter-class variance measures the difference in means between different categories, with the goal of making different classes as distinct as possible; while intra-class variance measures the distribution of data within each class, with the goal of making samples in the same class as close to each other as possible. By solving the generalized eigenvalue problem, LDA finds a projection matrix that reduces the data to at most c−1 dimensions, where c is the number of classes, to maximize the separation between classes. This dimensionality reduction facilitates classification problems while maintaining the class distinction of the data.

**LDA suitable model:**
(1) Classification model: LDA itself is a classification algorithm and therefore works well with other classification

models such as logistic regression, SVM, etc.

(2) Gaussian distribution assumption: LDA usually performs well if the data (or features) roughly fit a Gaussian distribution.

(3) Small sample size: LDA usually outperforms PCA for small sample size problems.

### 2.3.2 Principal Component Analysis (PCA)

1. Theoretical Foundations of PCA
Principal Component Analysis (PCA) is a Dimensionality reduction technique in multivariate analysis. Its primary objective is to orthogonalize a set of potentially correlated variables into a reduced set of linearly uncorrelated variables [10].
For the dataset X, aiming to utilize linear transformation **a** and make the variance maximum, we get:

$$J_v = \frac{1}{N}\sum_{n=1}^{N}(a^T x_n - a^T \overline{x})^2 = a^T S a$$

where:

$$S = \frac{1}{N}\sum_{n=1}^{N}(x_n - \overline{x})(x_n - \overline{x})^T = \frac{1}{N}X_c X_c^T$$

With the restriction of $a^T a = 1$, we can transform the problems into Lagrange Optimization problem:

Maximum $J = a^T S a - \lambda(a^T a - 1)$

After differentiation, we can get:

$$\frac{\partial J}{\partial a} = 2Sa - 2\lambda a = 0$$

It shows that the **a** is the eigenvector of S.

2. Core Objectives and Implications of PCA

In PCA method, there are three main objectives of data visualization, feature selection, and dimension reduction.

Dimensionality Reduction: High-dimensional data often suffers from the "disaster of dimensionality," making analyses computationally intensive and less interpretable [11]. PCA solves this by transforming the original variables into a new data set of uncorrelated principal components. It ensures the initial components encapsulate the majority of the data's variance.

Data Visualization: By reducing data to two or three dimensions, PCA offers a tangible representation. Visualization makes the representation more visible.

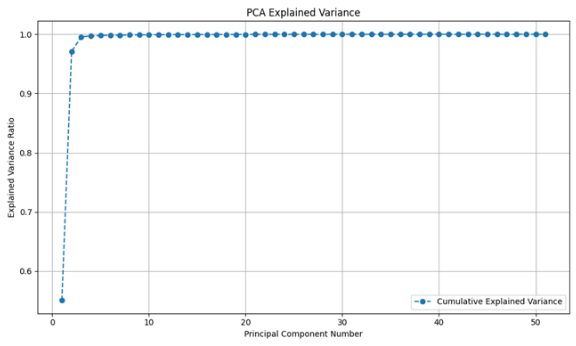
Feature Selection: Beyond mere reduction, PCA's transformative capabilities can distill salient features, potentially enhancing the performance of the model.

3. PCA implementation
We utilize the "sklearn.decomposition" library to implement the PCA method. It receives the data from the feature

selection implemented by random forest and executes PCA.

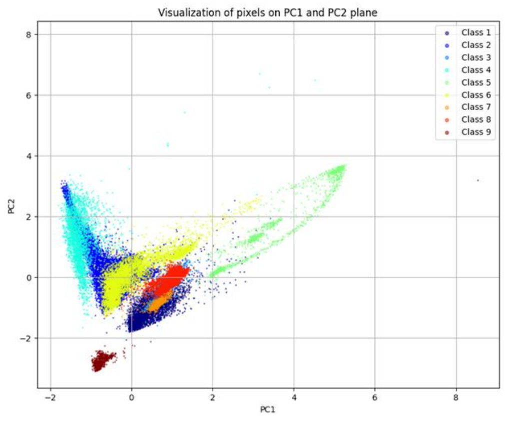
Explained Variance:
Explained variance provides a quantifiable measure of how much information (variance) each principal component retains [12]. In essence, this could be used to evaluate the importance of each component in representing the original data's structure.

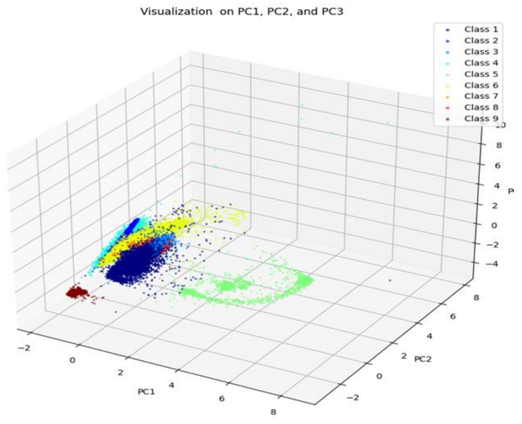
We plot a figure to delineate the explained variance across dimensions. It could guide the selection of the optimal number of components.



From this figure, we can find that the explained variance exceeds 95% when we select 2 principal components. When the component number equals 3, it approaches 99%. This indicates that 3 components are able to contain most of the information for pattern recognition. Thus, in the next step, we choose the three components and visualize them to check whether these classes can be clearly classified.

Visualization Post PCA Transformation:
When the number of principal component equal to 2 or 3, the data was then transmuted and visualized. This figure offers a spatial representation, elucidating the distribution and potential clustering of data points.





In these figures, we find that data points in different classes were not distinctly separable.
the data points exhibited significant overlap, suggesting that a two or three-dimensional representation might not be sufficient for clear class differentiation.

Augmented Analysis and Validation:
Initial visual inspections intimated suboptimal linear separability in reduced dimensions. To redress this, we need to

choose save feature information. In next model section, we e will utilize classification accuracy as a metric and seek for an optimal principal component number for each model.

## 2.4 Model 1: Naive Bayes Classifiers

### 2.4.1 Model Description

The Naive Bayes classifier is a classifier based on the Bayes theorem, it depends on the assumption that there is strong independence between the features. It has shown remarkable success in various applications even it is simpler than the other pattern recognition algorithm [13]. This section conducts a review of the basic theory of naive Bayes classifiers.

The primary goal of Bayes classifiers is to maximize the posterior probability, which means assigning a sample to the class that has the highest posterior probability given the observed features of the sample.
Mathematically, this can be represented as:

$$\omega_i = argmax(P(\omega_i \mid X)$$

Where: $\omega_i$ is the class that maximizes the posterior probability.
X is the observed feature points of the sample

Using the Bayes theory, we can find that this formula can be written as:

$$P(\omega_i \mid X) = \frac{P(x|w_i) \cdot P(w_i)}{P(X)}$$

Considering that P(X) is certain in one sample, we transfer the problems into

$$\omega_i = argmax(P(\omega_i \mid X) \propto argmax(P(x|w_i) \cdot P(w_i))$$

Function and classification process：
We have introduced the basic principle of the Bayes classifier. Then we will introduce the detailed working process of Bayes classifiers.

1. Classification Ability:
The Naive Bayes is fundamentally a probabilistic classifier. It is generally used for binary or multiclass classification tasks.

2. Adaptable to Multiclass Problems:
Due to its inherent design, the Naive Bayes classifier can be seamlessly adapted to scenarios requiring multiclass predictions.

3. Text Classification/Spam Filtering
Bayes classifiers show effectiveness in text categorization tasks, such as spam email filtering and sentiment analysis.

### 2.4.2 Environment and Hardware

Environment: Python 3.10, anaconda.
Hardware: 13th Gen Intel(R) Core (TM)

i5-13500HX 2.50 GHz
Special Library: Intel extension for scikit-learn (accelerate the training on CPU.

### 2.4.3 Parameters and Training Procedure

In this section, we implement the parameters estimation procedure in the Bayes classifier, which includes mean μ and variance $\sigma^2$. Additionally, the reduced dimension optimization will be conducted and we will choose the dimension for dimensionality reduction based on accuracy to achieve the optimal result.

#### 2.4.3.1 Parameters Estimation

**Prior Probability Estimation**

We utilize the default prior probability estimation method in sk-learn Bayes classifiers. It calculates the percentage of one class in the training set as its prior probability.

$$\boldsymbol{P(\omega_i) = \frac{\mathrm{N}(w_i)}{N}}$$

**Mean and variance Estimation with Maximum Likelihood Estimation**

MLE aims to find the parameter values that maximize the likelihood function, which measures how well the model explains the observed data. The likelihood for a set of parameter values is the probability of the observed data given those values [14].

Given a dataset, the goal is to estimate the parameters that maximize the likelihood of observing this data under a normal distribution assumption.

The likelihood function for a normal distribution is given by:

$$L(\mu, \Sigma) = \prod_{i=1}^{n} \frac{1}{(2\pi)^{\frac{d}{2}} |\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(X_i - \mu)^T \Sigma^{-1} (X_i - \mu)\right)$$

To find the MLE estimates, we differentiate the log-likelihood and equate it to zero.

For mean:

$$\hat{\mu}_{ML} = \frac{1}{N} \sum_{k=1}^{N} x_k$$

For the variance:

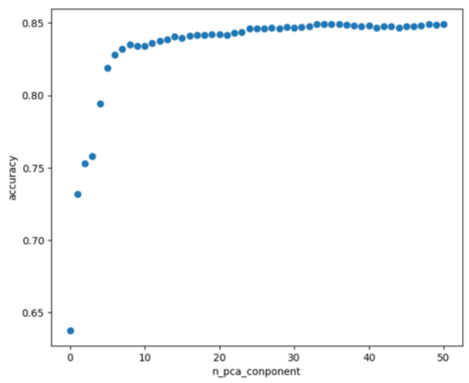
$$\hat{\sigma}^2_{ML} = \frac{1}{N} \sum_{k=1}^{N} (x_k - \mu)^2$$

#### 2.4.3.2 Dimension Optimization with PCA and LDA

This section presents a comprehensive study on the application of Bayesian classification post dimensionality reduction with Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA). Through rigorous validation experimentation across varying dimensions, this research aims to identify the optimal dimensionality that maximizes classification accuracy.

The **Dimension Optimization with PCA**

Since we have selected 51 spectrums from initial datasets with a random tree algorithm, the PCA dimension will range from 1 to 51. In order to determine the best dimension, we conducted a cross-validation experiment on Bayes classifiers. As a result, we utilize a figure to measure the accuracy on different dimensions with PCA.

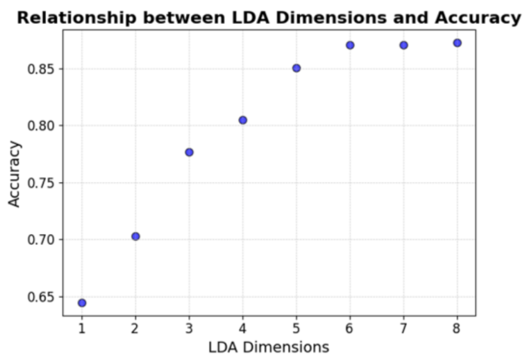


Upon applying PCA, it was observed that the accuracy of the Bayesian classifier varied across dimensions. In the case of PCA, when the dimension equals 33, the highest accuracy was 85%.

**(2) Dimensionality Optimization with LDA**

Since we only have 9 classes in datasets, the LDA dimension ranges from 1-9. Similar to PCA, we plot a figure to show the accuracy in different dimensions.

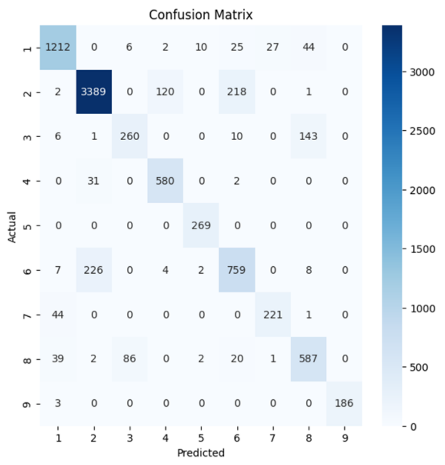
**Sin**



From the plot, we can find that when the dimension is equal to 8, the Bayes classifier achieves the highest accuracy of 87%.

### 2.4.4 Evaluation

After the dimension validation, we have confirmed that when the dimension is 8 with the LDA method, Bayes could achieve the highest accuracy. Here we will utilize the confusion matrix and other metrics to conduct an evaluation.



Here we use a confusion matrix to evaluate the model. The given confusion matrix presents the predictive performance of the model across various classes. Observing the diagonal values, it's evident that most predictions are accurate, specifically in class 2 and class 5. Furthermore, several classes exhibit misclassification counts of zero or very low, suggesting that the model distinctly recognizes them with high confidence. However, for classes 1, 3, and 7, although the correct predictions remain the highest, they present relatively higher misclassifications when compared to other classes. Overall, Bayes classifiers exhibit a high predictive capability.

The confusion matrix is intuitive, now we could use data to analyze the model more concretely.
Here we utilize accuracy, precision, recall, and F1-score.

| | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| 1 | 0.92 | 0.92 | 0.92 | 1326 |
| 2 | 0.91 | 0.91 | 0.91 | 3730 |
| 3 | 0.76 | 0.59 | 0.66 | 420 |
| 4 | 0.80 | 0.95 | 0.87 | 613 |
| 5 | 0.95 | 1.00 | 0.97 | 269 |
| 6 | 0.75 | 0.70 | 0.72 | 1006 |
| 7 | 0.88 | 0.82 | 0.85 | 266 |
| 8 | 0.75 | 0.82 | 0.79 | 737 |
| 9 | 1.00 | 0.99 | 0.99 | 189 |
| Accuracy | | | 0.87 | 8556 |
| Macro Avg | 0.86 | 0.86 | 0.86 | 8556 |
| Weighted Avg | 0.87 | 0.87 | 0.87 | 8556 |

## 2.5 Model 2: K-NearestNeighbor

### 2.5.1 Model Description

#### Theory

KNN (K-Nearest Neighbor Algorithm) is an instance-based learning algorithm commonly used for classification and regression tasks. For a new unknown sample, KNN finds the k samples closest to that sample in the training set (i.e., the "nearest neighbors"), and then predicts the properties of the unknown sample based on the properties of those neighbors.[15]

#### Function

To determine which samples are the "nearest neighbors" of the new sample, we need a distance metric. Common distance measures are:
Euclidean Distance: For two points P (x1, y1) and Q (x2, y2). The Euclidean distance is defined as:

$$d(P,Q) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

For multi-dimensional Spaces, this formula can be extended to:

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$$

or a classification task, KNN considers the number of samples from each class in the k nearest neighbors and assigns new samples to the class with the most number.

The posterior probability of KNN is the probability that, given an input sample, each class is predicted to be the correct class. For classification problems, not only the most likely category is represented, but also the probability of this prediction can be known [16].

Calculate a posteriori probability: For each class ki, calculate how often it occurs in these k neighbors, which can be expressed by the following formula:

$$p_n(x,\omega_i) = \frac{k_i}{nV}$$

$$P_n(\omega_i|x) = \frac{p_n(x,\omega_i)}{\sum_i^N p_n(x,\omega_i)} = \frac{k_i}{k}$$

Pn(x, wi): The sum of the joint probabilities of all possible classes equals the number k of all samples in the window.

### 2.5.2 Environment and Hardware

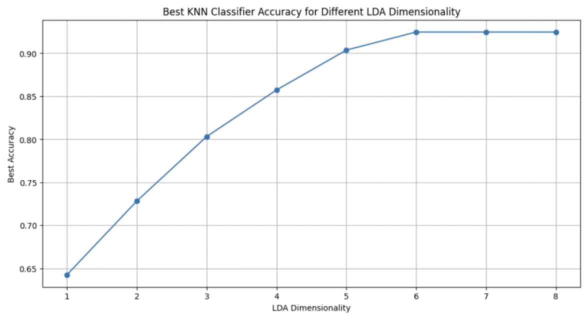
**Hardware**
CPU:
13th Gen Intel(R) Core (TM) i9-13980HX 2.20 GHz
RAM:
16GB

**Environment**
Sklearnex: To improve CPU utilization, the sklearnex package, a Python package developed by Intel to accelerate Scikit-learn, was installed in the environment.

### 2.5.3 Parameters and Training Procedure

#### 2.5.3.1 Select the dimensional reduction method

The dimensionality reduction methods were PCA and LDA. The two methods are traversed and the one with the highest accuracy is chosen as the dimensionality reduction method.



This graph shows the best accuracy using K-nearest neighbor (KNN) classifiers in different linear discriminant analysis (LDA) dimensions.

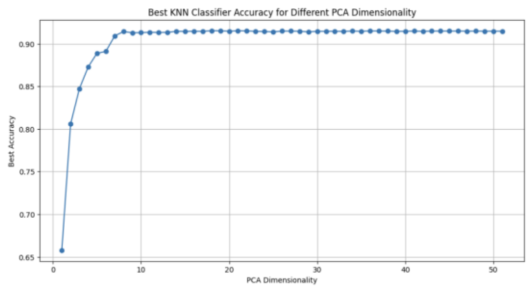
1. The increased LDA dimension is

positively correlated with improved accuracy: As the LDA dimension increases from 1 to 6, the accuracy of the KNN classifier also increases.

2. Accuracy saturation: at LDA dimensions 6, 7, and 8, accuracy appears to have saturated and remained at the same level without further increase.

3. Initial gain is significant: The increase in accuracy is particularly significant as the LDA dimension increases from 1 to 3. This could mean that these preliminary dimensions capture major changes in the data and help improve the classifier's performance.

In linear discriminant analysis (LDA), the optimal dimension is 6. For this dimension, the optimal value of k is 13. The highest accuracy obtained is 92.46%.



This graph shows the best accuracy of a K-nearest neighbor (KNN) classifier in different principal component analysis (PCA) dimensions.

The best dimension of PCA is 22. The best k of the best dimension 22 is 5. The best accuracy is 91.60%.

Because the accuracy of dimensionality reduction with LDA is higher, we decided to use LDA.

### 2.5.3.2 Select the Best Parameters

In order to find the best KNN parameters, I planned to traverse all the parameters but found that there was not enough memory. Decide to use a distributed grid search to determine the best parameters for each in turn. Iterate through each parameter in turn, selecting the parameter with the highest accuracy.

**n(1)_neighbors**: The core parameter of KNN represents the value of "k", that is, how many neighbors are considered to make decisions.

**w(2)eights**: It is used to determine the neighbor's weight.

The best weight option is distance with an accuracy of 91.39%.

'uniform': In this case, each neighbor has the same voting rights. This means that the algorithm simply decides which category to belong to base on the number of neighbors, regardless of their distance to the query point.[17]

'distance': In this setting, the weight of neighbors is inversely proportional to their distance to the query point [17]. This means that the closer neighbor has more decision-making power than the farther neighbor.

**m (3) metric:** The metric parameter is used in the KNN algorithm to determine the distance or similarity between data points. xi and yi are the coordinates of

two points in the i - th dimension.[18]
The best metric option is manhattan with an accuracy of 91.77%.
'euclidean': Euclidean distance. This is the "straight line" distance between two points on a plane

$$\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

'manhattan': Manhattan distance, also known as L1 distance or city block distance. It is the distance measured along the axis (not diagonally).

$$\sum_{i=1}^{n}|x_i - y_i|$$

'minkowski': Minkowski distance, when p=2, this is the Euclidean distance; When p=1, it's the Manhattan distance; As p→∞, it approaches the Chebyshev distance.[19]

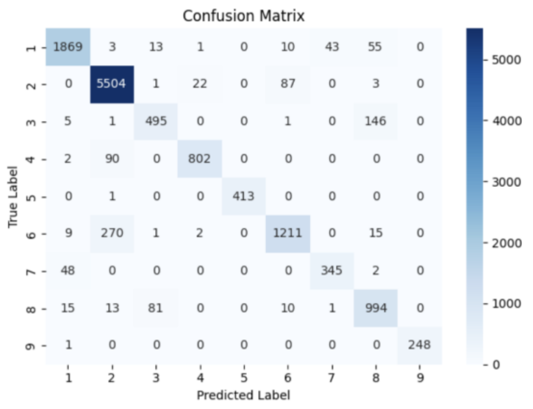
$$(\sum_{i=1}^{n}|x_i - y_i|^p)^{\frac{1}{p}}$$

#### 2.5.3.3 The Best Accuracy under the Best LDA

The best dimension of LDA is 6. The best k of the best dimension 6 is 12. The best accuracy is 92.58%.

### 2.5.4 Evaluation

#### 2.5.4.1 Confusion Matrix



From this confusion matrix, it can be observed that:
Category 2 (with a value of 5504) was predicted very well, and most of the samples were classified correctly.
Category 6 had a large misclassification, with 1211 samples misclassified into category 5.
For Category 7, 345 samples were misclassified as Category 6.
Some other categories, such as categories 1, 3, 4, 5, 8, and 9, also have relatively good classification results, but there are still some misclassifications.

#### 2.5.4.2 Classification Report

| | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| 1 | 0.96 | 0.94 | 0.95 | 1994 |
| 2 | 0.93 | 0.98 | 0.96 | 5617 |
| 3 | 0.84 | 0.76 | 0.80 | 648 |
| 4 | 0.97 | 0.89 | 0.93 | 894 |
| 5 | 1.00 | 1.00 | 1.00 | 414 |
| 6 | 0.92 | 0.80 | 0.86 | 1508 |
| 7 | 0.89 | 0.87 | 0.88 | 395 |
| 8 | 0.82 | 0.89 | 0.85 | 1114 |
| 9 | 1.00 | 1.00 | 1.00 | 249 |
| Accuracy | | | 0.93 | 12833 |
| Macro Avg | 0.93 | 0.90 | 0.91 | 12833 |
| Weighted Avg | 0.93 | 0.93 | 0.92 | 12833 |

This is a category report that provides details on accuracy, recall, F1 scores, and support for each category.
**Precision:**

For a particular category, accuracy is the ratio of the samples that are correctly predicted for that category to all the samples that are predicted for that category.

**Recall:**
For a particular category, recall is the ratio of the samples that were correctly predicted for that category to all the samples that actually belong to that category.

**FF1 Score:**
The F1 score is a harmonic average of accuracy and recall. It tries to provide a balanced measurement of both.
Support: For each category, support is the number of samples in the data that actually belong to that category. For example, category 1 has 1994 samples.
accuracy: This is the overall correct classification rate. Here, the model has an accuracy of 0.93, meaning that 93% of the sample was correctly classified.
macro avg: This is a simple average of accuracy, recall, and F1 scores for each category.

**Weighted Avg:**
This is a weighted average of class accuracy, recall, and F1 scores based on support for each category.

From the classification report:
Category 5 and Category 9 performed particularly well, as they achieved 1.00 on all three metrics (accuracy, recall and F1 score).
Category 3 had a lower recall rate of 0.76, which could mean that some samples in that category were misclassified into other categories.
Overall, the models performed relatively well, as most of the accuracy, recall, and F1 scores were above 0.8.

## 2.6 Model 3: Support Vector Machines

### 2.6.1 Model Description

#### 2.6.1.1 SVM Theory principals and implementation process on the *PaviaU* dataset

**SVM Intuition:**

SVM in Hyperspectral Imaging: In the context of *PaviaU* (or any hyperspectral image), each pixel has multiple bands, providing a high-dimensional feature vector for each pixel. SVM attempts to find a hyperplane that separates pixels of one class from others in this high-dimensional space [20].

Support Vectors: The pixels that lie closest to the decision boundary and influence its orientation and position are termed support vectors. The hyperplane is driven by these pixels [21].

**Formulization:**

Decision Boundary: The hyperplane is mathematically represented as:

$$w \cdot x + b = 0$$

Where *w* is the weight vector and *b* is the bias.

**Objective:**

For a *PaviaU* image, considering its high dimensionality, the aim is to maximize the margin between different classes. If *yi* represents the label of the i*th* pixel and *xi* . Its feature vector:

$$y_i(w \cdot x_i + b) \geq 1$$

for every pixel *i*.

**Lagrange Duality:**

Lagrangian Formulation: The above can be turned into an optimization problem using Lagrange multipliers. The Lagrangian is:1

$$L(w, b, \alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{N} \alpha_i [y_i(w \cdot x_i + b) - 1]$$

Where *αi* are the Lagrange multipliers and *N* is the number of pixels in the *PaviaU* image.
Dual Problem: By converting the primal problem to its dual, we have:

$$\max_\alpha \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i \cdot x_j$$

Ensuring:

$$\alpha_i \geq 0 \text{ and } \sum_{i=1}^{N} \alpha_i y_i = 0.$$

Kernel Trick:

Higher Dimensionality in *PaviaU*: Some classes in *PaviaU* may not be linearly separable in the original band space. The kernel trick helps project these pixels into a space where they can be linearly separated [22].

Kernel Function: This function computes the dot product in the transformed space without actually doing the transformation. For pixel *xi* and *xj*:

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$$

Some kernels include:

Linear:

$$K(x_i, x_j) = x_i \cdot x_j$$

Polynomial:

$$K(x_i, x_j) = (1 + x_i \cdot x_j)^d$$

RBF:

$$K(x_i, x_j) = e^{-\gamma||x_i - x_j||^2}$$

Soft Margin:

Handling Noise in *PaviaU*: Given the noise and inherent variability in hyperspectral images, it's hard to achieve perfect separation. Soft margin SVM introduces slack to tolerate some misclassifications [23].

The objective with Slack: By introducing Slack variables *ξi*:

$$\frac{1}{2}||w||^2 + C\sum_{i=1}^{N}\xi_i$$

Subject to:

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i \text{ and } \xi_i \geq 0$$

Where *C* strikes a balance between margin maximization and misclassification minimization.

#### 2.6.1.2 General Functionality and Applications of SVM

Multi-Class Classification: SVM is used for multi-class classification in *PaviaU* remote sensing image classification. It aims to find the best hyperplane to effectively distinguish between multiple different land cover or feature classes in the image, allowing each pixel to be assigned to the corresponding class.

Regularization: Support Vector Machines contains the regularization process, which is crucial for controlling model complexity and preventing overfitting [24]. Regularization balances the trade-off between maximizing the margin and minimizing classification mistakes, ensuring that the model performs effectively with unknown data.

Soft Margin Classification: Support Vector Machines (SVM) introduce the notion of soft margin classification, which permits a specific tolerance range for some data points to be incorrectly classified. This flexibility improves the robustness of classification, especially when working with noisy or non-linearly separable data. Given that remote sensing data frequently contains noise or intricate non-linear features and that soft margin classification can increase classification robustness, this can be extremely helpful for your PaviaU image classification problem.

### 2.6.2 Parameters Estimation

Common Hyperparameters in SVM: Kernel Function, C (Regularization Parameter), γ (RBF Kernel Bandwidth Parameter), Degree (Degree of Polynomial Kernel), Coef0 (Sigmoid Kernel Parameter).

Several kernel functions that match various parameters are used while utilizing Support Vector Machines (SVM) for classification in order to determine the optimal model performance. The definitions of each kernel function and its accompanying parameters are listed below:

1. Linear Kernel (linear)

Purpose: The linear kernel is the simplest kernel function. It assumes the data is linearly separable, meaning there exists a hyperplane that can separate the data.

**Parameters:**

C: Penalty parameter. It controls the penalty for misclassified points. A smaller C enables the classifier to be more forgiving of misclassifications, which may result in a bigger decision boundary. A greater C indicates the classifier is more anxious to ensure each data point is correctly classified, but it may also cause overfitting.

2. Polynomial Kernel (poly)

Purpose: The polynomial kernel can capture the nonlinear patterns in the data. It allows for improved classification performance when the data isn't linearly separable by introducing polynomial terms.

**Parameters:**

C: Same as linear kernel

degree: Determines the order of the polynomial. A higher degree can result in more complex decision boundaries.

gamma: Controls the influence of each training sample.

coef0: A constant term in the kernel function. It can influence the shape of the decision boundary.

3. RBF Kernel (rbf)

Purpose: The RBF kernel is a highly flexible kernel function capable of capturing complex nonlinear decision boundaries. It's based on the Euclidean distance between data points.

**Parameters:**

C: Same as linear kernel.

gamma: Controls the shape of the RBF kernel. A smaller gamma results in a wider decision boundary, while a larger gamma results in a narrower boundary.

4. Sigmoid Kernel (sigmoid)

Purpose: The sigmoid kernel function is a common activation function in neural networks. In SVM, it can introduce nonlinear decision boundaries.

**Parameters:**

C: Same as linear kernel.

gamma: Same as RBF kernel.

coef0: Controls the threshold of the sigmoid function.

Hyperparameter Combination: Kernel, gamma, and C are crucial in SVM. Turning these enhances performance [25]. Besides, adjusting the other two would result in a lack of computing resources. Therefore, we choose to consider only the three most important hyperparameters, the kernel function, gamma, and C.

### 2.6.3 Environment and Hardware

We briefly describe our computing resource for the SVM model.

Environment: Python 3.8.17, anaconda.

Hardware: 13th Gen Intel(R) Core (TM) i9-13900HX 2.20 GHz

Library: Using Sklearnex to accelerate CPU

### 2.6.4 Parameters and Training Procedure

#### 2.6.4.1 Hyperparameter value

(1)C (Slack Parameter)

We select the values [0.001, 0.01, 0.1, 1, 10, 100] for the C parameter for the following reasons:

1. Logarithmic Scale: Many hyperparameters have optimal values spanning a wide logarithmic range. A logarithmic scale ensures comprehensive search and efficiency.
2. Regularization Balance: Small C values (e.g., 0.001) offer strong regularization, useful against overfitting. Large C values (e.g., 10) provide weak regularization, suitable for intricate data.
3. Common Practice: These values are common starting points in literature and practice for initial hyperparameter searches [26].

(2) Gamma

We select the values [0.01, 0.1, 1, 10, 'auto', 'scale'] for the gamma parameter for the following reasons:

1. Varying Scale: The chosen gamma values cover a wide range, from small to large, testing the model's complexity. Small gamma means more flexibility, while large gamma results in a tighter fit to the data.

2. Computational Efficiency: These selections balance thoroughness and computational efficiency. Trying all possible values can be computationally expensive, especially with large datasets.

3. 'auto' and 'scale' Options:

   'auto' uses 1/n_features as gamma, providing a heuristic for datasets with many features.

'scale' considers dataset variance, often leading to better generalization by default [6].

#### 2.6.4.2 Parameters Cross-validation

After cross-validation with different kernel functions for different parameters, we find that the model performs best with an accuracy of 0.95 when the C parameter of 10 and gamma parameter of 0.1 are chosen for the rbf kernel function without using downscaling methods.

The accuracy is 95.14%

#### 2.6.4.3 Select the dimensional reduction method

In order to achieve the best accuracy, we separately test the accuracy under PCA and LDA. We kept the hyperparameters of the SVM fixed with the previous best parameters combination.

For the PCA model, we iterate through all possible values of the retained principal components and select the one that yields the highest accuracy through a for loop.

The best PCA components number is 9 and the best accuracy is 94.52%.

The maximum dimension for the LDA model is one fewer than the total number of classes. We identify the dimension that yields the maximum accuracy by iteratively going through all the dimensions.

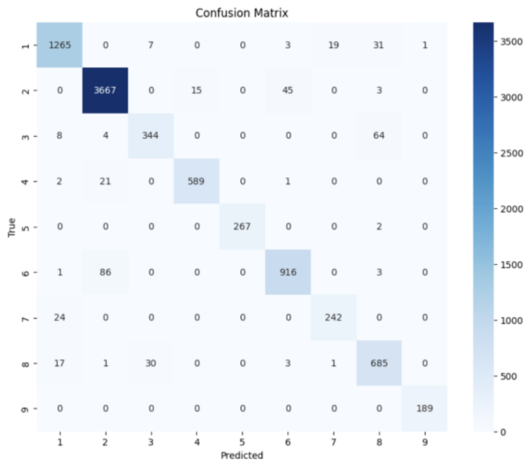
The best LDA dimension is 8 and the best accuracy is 92.82%.

Although the use of dimensionality reduction methods can significantly reduce the training duration, we found that the accuracy rate decreases after using dimensionality reduction methods. This could be due to the following reasons:

1. Loss of Important Variance (PCA): PCA wants to retain the dimensions that explain the most variance in the dataset. However, for classification, not all high-variance features are necessarily the most discriminative ones. With PCA, we might lose some features that are important for the SVM to draw optimal decision boundaries, even if they account for less variance.

2. Class Discrimination vs. Class Separation (LDA): While LDA attempts to maximize the distance between class means and minimize the spread (variance) within each class, it might not always be the best approach for complex datasets like remote sensing data. *PaviaU*, being a hyperspectral dataset, might have subtle spectral signatures that LDA fails to capture, leading to sub-optimal projections [27].

Finally, without using dimensionality reduction methods, we can print the confusion matrix and classification report:



|  | precision | recall | f1-score | support |
|---|---|---|---|---|
| 1 | 0.96 | 0.95 | 0.95 | 1326 |
| 2 | 0.97 | 0.99 | 0.98 | 3730 |
| 3 | 0.90 | 0.81 | 0.85 | 420 |
| 4 | 0.98 | 0.97 | 0.98 | 613 |
| 5 | 1.00 | 0.99 | 1.00 | 269 |
| 6 | 0.95 | 0.91 | 0.93 | 1006 |
| 7 | 0.91 | 0.90 | 0.90 | 266 |
| 8 | 0.87 | 0.93 | 0.90 | 737 |
| 9 | 0.99 | 1.00 | 1.00 | 189 |
| accuracy |  |  | 0.95 | 8556 |
| macro avg | 0.95 | 0.94 | 0.95 | 8556 |
| weighted avg | 0.95 | 0.95 | 0.95 | 8556 |

We can see that:

Class 2 and Class 9 are well-predicted with minor errors.

Class 3 has a notable misclassification with Class 7.

Class 4 has confusion primarily with Class 7.

Classes 5 and 6 are predicted with high precision.

Class 7 is largely correct but gets confused with Class 3.

Class 8 has minor confusion with Classes 2, 4, and 9.

Overall, the model is effective, but improvements can be made in distinguishing between Classes 2 and 6.

# 3. Discussion

## 3.1 Bayes Model

**Advantages:**

(1) Simplicity

Compared to other classifiers, the Bayes theory is more straightforward. Additionally, Bayes classifiers offer simpler hyperparameter tweaking procedures than other classifiers. The model does not need to take into account intricate interactions between characteristics because of the conditional independence assumption between features. As a result, it just needs to estimate the probability distribution of one feature, significantly reducing the model's complexity and avoiding the need of excessively numerous hyperparameters.

(2) Efficiency:

In order to achieve similar performance, Bayes classifiers consume less time and computing resources. Firstly, Bayes classifiers' core mechanism relies on the independence assumption between features, which streamlines the computational process and accelerates predictions. Besides, Bayes classifiers could avoid complex linear algebra calculations. When applying Bayes' theorem, we are mainly concerned with probability and conditional probability, which usually involves calculating basic statistics such as mean, variance, probability, etc., but does not involve operations such as matrix operations.

**Drawbacks:**
(1) Assumptive Independence:
Bayesian classifiers assume that data features are mutually independent. This requirement is strict and it doesn't always hold in real-world scenarios. It means we need to do more processes before we utilize Bayes Theory.

(2) Rely on prior probability
Even though Bayes classifiers do not have complex mathematical theory derivation like SVM, Bayes classifiers strongly rely on the value of prior probability. In some cases, the choice of prior probability could significantly influence the classification performance.

(3) Over-Simplification:
It may miss complex linkages including feature interactions because it mostly depends on the feature independence assumption. As such, its inherent simplicity may render it unsuitable for ensuring complex data patterns.

## 3.2 KNN Model

**Advantages:**
(1) Simplicity:
The principle of the KNN algorithm is very simple, it has no training process and only calculates when predicting. The basic principle of the KNN algorithm is to predict the classification or output of a sample based on the classification or output of k nearest neighbors of the sample [15]. Since it does not involve a training process, the implementation of the algorithm is very intuitive. When there is a new data point that needs to be classified or predicted, simply calculate its distance from a known sample and then vote or average according to the nearest k samples.

(2) Data independent distribution:
KNN is instance-based learning, so it does not assume that data follows a particular distribution. Many traditional statistical methods require the assumption that the data follows a specific distribution, such as a normal distribution. If these assumptions are not true, the performance of the model may suffer. However, KNN does not require any assumptions about the distribution of the data. It is entirely based on the structure of the data itself, which makes KNN perform better on data that does not follow a particular distribution.

**Drawbacks**:
(1) High computational complexity:

It is necessary to compute the distance to every training sample for every prediction sample [28]. In order to identify the closest k neighbors, KNN computes the distance between each new data point and all of the training data points at the time of prediction. Predictions may become quite slow as a result, particularly if the data set is vast.

(2) High space complexity:
KNN needs to store all the training data, which can lead to insufficient memory when the data volume is large.

(3) Sensitivity:
KNN is susceptible to anomalies. A significant forecast variation may result from an outlier. Since KNN makes all of its predictions using training data, it is highly susceptible to anomalies or noisy data. This is due to the fact that an outlier, particularly if k is a tiny value, has the potential to distort vote outcomes [28].

(4) Dimensionality disaster:
As features increase, the distance between samples tends to be consistent, which makes KNN less effective on high-dimensional data [29]. Mathematically, as dimension d increases, the difference in the distance between any two points decreases, approaching a constant.

## 3.3 SVM Model

**Advantages:**

(1) High-Dimensional Data Handling: Support Vector Machines (SVM) are adept at managing high-dimensional datasets, like the PaviaU dataset, which has several spectral bands.

(2)

SVM exhibits strong performance even in situations with small sample sizes since its effectiveness is mostly dependent on support vectors rather than data volume.

(3)The Regularization Parameter offered by Support Vector Machines (SVM) aids in mitigating the risk of overfitting.

(1) Kernel Trick: SVM can employ kernel functions to address non-linear problems, which is valuable in remote sensing imagery where class boundaries may be complex and non-linear.

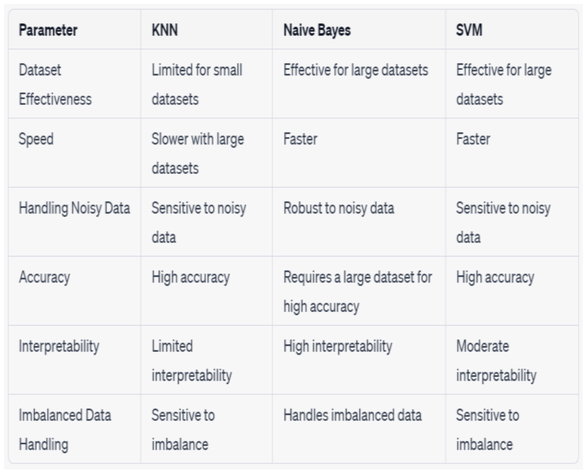
(2) Robustness: SVM is robust to noise and outliers commonly present in remote sensing data.

**Drawbacks:**

(1) Computational Complexity: SVM can require a lot of processing power, especially when dealing with huge datasets. This is especially the case when choosing the right kernel functions and parameters.
(2)
(3) Limited Interpretability: In remote sensing applications where comprehending model decisions is crucial, SVM models may be difficult to interpret.

(3) Sensitivity to Parameter Decisions: The regularization parameter and kernel selection are two parameters that have a significant impact on how well SVM performs.

Imbalanced Data: When dealing with imbalanced datasets, where certain classes have significantly more samples than others, SVM may not perform well without proper handling, such as sample weighting or resampling.



| Parameter | KNN | Naive Bayes | SVM |
|---|---|---|---|
| Dataset Effectiveness | Limited for small datasets | Effective for large datasets | Effective for large datasets |
| Speed | Slower with large datasets | Faster | Faster |
| Handling Noisy Data | Sensitive to noisy data | Robust to noisy data | Sensitive to noisy data |
| Accuracy | High accuracy | Requires a large dataset for high accuracy | High accuracy |
| Interpretability | Limited interpretability | High interpretability | Moderate interpretability |
| Imbalanced Data Handling | Sensitive to imbalance | Handles imbalanced data | Sensitive to imbalance |

## 4. Conclusion

The report successfully utilizes feature selection and dimensionality reduction techniques, specifically random forest, Linear Discriminant Analysis (LDA), and Principal Component Analysis (PCA), in the analysis of the *PaviaU dataset*. Then the data after dimensionality reduction has been used in classification models, which are Naive Bayes, K-Nearest Neighbor, and Support Vector Machine. We conduct validation on these models and choose the best dimensionality reduction methods and hyperparameters. Finally, we evaluate the models based on the confusion matrix and classification report.

Based on the whole work, we get the result that SVM and KNN could get better performance, while they consume more computing resources. Naive Bayes is the most efficient model, while it is not powerful on that dataset. Besides, we find that different dimensionality reduction method has different advantages and drawbacks. Thus, in experiments, we need to conduct more validation experiences to choose the method to use.

In future work, emphasis can be placed on enhancing the precision and efficiency of existing Bayesian, SVM, and KNN algorithms applied to remote sensing image classification. This can be achieved through algorithmic improvements and exploring integration with deep learning techniques to better leverage the unique characteristics of remote sensing data. Additionally, extending the application of these algorithms to new domains such as agriculture and urban planning, backed by real-world case studies, will showcase their practical effectiveness.

## 5.Novelty

In today's data-driven environment, using machine learning for pattern detection has become essential. We preprocess features using a Random Forest to improve model performance. This integration makes use of each algorithm's own advantages while simultaneously maximizing the potential of ensemble learning. It achieves more reliable, precise, and perceptive performance.

## 6. References

[1]
"Hyperspectral Remote Sensing Scenes - Grupo de Inteligencia Computacional (GIC)," *www.ehu.eus*. https://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes#Pavia_Centre_and_University (accessed Oct. 26, 2023).

[2]
P. Liu, K.-K. R. Choo, L. Wang, and F. Huang, "SVM or deep learning? A comparative study on remote sensing image classification," *Soft Computing*, vol. 21, no. 23, pp. 7053–7065, Jul. 2016, doi: https://doi.org/10.1007/s00500-016-2247-2.

[3]
Md. A. M. Hasan, M. Nasser, S. Ahmad, and K. I. Molla, "Feature Selection for Intrusion Detection Using Random Forest," *Journal of Information Security*, vol. 07, no. 03, pp. 129–140, 2016, doi: https://doi.org/10.4236/jis.2016.73009.

[4]
E. Scornet, "Trees, forests, and impurity-based variable importance in regression," *Annales de l'Institut Henri Poincaré, Probabilités et Statistiques*, vol. 59, no. 1, Feb. 2023, doi: https://doi.org/10.1214/21-aihp1240.

[5]
M. Chaibi, E. M. Benghoulam, L. Tarik, M. Berrada, and A. El Hmaidi,

“Machine Learning Models Based on Random Forest Feature Selection and Bayesian Optimization for Predicting Daily Global Solar Radiation,” *International Journal of Renewable Energy Development*, vol. 11, no. 1, pp. 309–323, Nov. 2021, doi: https://doi.org/10.14710/ijred.2022.41451.

[6]
J. Ye, Ravi Janardan, and Q. Li, “Two-Dimensional Linear Discriminant Analysis,” *Neural Information Processing Systems*, vol. 17, pp. 1569–1576, Dec. 2004.

[7]
G. T. Reddy *et al.*, “Analysis of Dimensionality Reduction Techniques on Big Data,” *IEEE Access*, vol. 8, pp. 54776–54788, 2020, doi: https://doi.org/10.1109/ACCESS.2020.2980942.

[8]
W.-S. Zheng, J. H. Lai, P. C. Yuen, and S. Z. Li, “Perturbation LDA: Learning the difference between the class empirical mean and its expectation,” *Pattern Recognition*, vol. 42, no. 5, pp. 764–779, May 2009, doi: https://doi.org/10.1016/j.patcog.2008.09.012.

[9]
E. K. Tang, P. N. Suganthan, X. Yao, and A. K. Qin, “Linear dimensionality reduction using relevance weighted LDA,” *Pattern Recognition*, vol. 38, no. 4, pp. 485–493, Apr. 2005, doi: https://doi.org/10.1016/j.patcog.2004.09.005.

[10]
A. Daffertshofer, C. J. C. Lamoth, O. G. Meijer, and P. J. Beek, “PCA in studying coordination and variability: a tutorial,” *Clinical Biomechanics*, vol. 19, no. 4, pp. 415–428, May 2004, doi: https://doi.org/10.1016/j.clinbiomech.2004.01.005.

[11]
R. Bellman, “DYNAMIC PROGRAMMING AND LAGRANGE MULTIPLIERS,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 42, no. 10, pp. 767–769, Oct. 1956, doi: https://doi.org/10.1073/pnas.42.10.767.

[12]
I. T. Jolliffe, “Principal Component Analysis,” *Technometrics*, vol. 30, no. 3, pp. 351–351, Aug. 1988, doi: https://doi.org/10.2307/1270093.

[13]
D. W. Aha, D. Kibler, and M. K. Albert, “Instance-based learning algorithms,” *Machine Learning*, vol. 6, no. 1, pp. 37–66, Jan. 1991, doi: https://doi.org/10.1007/bf00153759.

[14]
L. L. Cam, “Maximum Likelihood: An Introduction,” *International Statistical Review / Revue Internationale de Statistique*, vol. 58, no. 2, p. 153, Aug. 1990, doi: https://doi.org/10.2307/1403464.

[15]

I. Rish, “An empirical study of the naive Bayes classifier,” Jan. 2001.

[16]
R. O. Duda, D. G. Stork, and P. E. Hart, *Pattern classification and scene analysis. Part 1, Pattern classification*. New York ; Chichester: Wiley, 2000. Available: https://dl.acm.org/citation.cfm?id=954544

[17]
S. A. Dudani, “A Note on Distance-Weighted k-Nearest Neighbor Rules,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 8, no. 4, pp. 311–313, 1978, doi: https://doi.org/10.1109/tsmc.1978.4309958.

[18]
T. Cover and P. Hart, “Nearest neighbor pattern classification,” *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, Jan. 1967, doi: https://doi.org/10.1109/tit.1967.1053964.

[19]
Michel Marie Deza, E. Deza, and Springerlink (Online Service, *Encyclopedia of Distances*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.

[20]
J. Anthony Gualtieri and S. R. Chettri, “Support vector machines for classification of hyperspectral data,” Nov. 2002, doi: https://doi.org/10.1109/igarss.2000.861712.

[21]
G. Camps-Valls and L. Bruzzone, *Kernel Methods for Remote Sensing Data Analysis*. John Wiley & Sons, 2009. Accessed: Oct. 26, 2023. [Online]. Available: https://books.google.com/books?hl=en&lr=&id=_KhUMXQQkmQC&oi=fnd&pg=PA51&dq=J.+A.+Gualtieri

[22]
S. Wan, C. Gong, P. Zhong, S. Pan, G. Li, and J. Yang, “Hyperspectral Image Classification With Context-Aware Dynamic Graph Convolutional Network,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 59, no. 1, pp. 597–612, Jan. 2021, doi: https://doi.org/10.1109/tgrs.2020.2994205.

[23]
F. Kam, 2009. Accessed: Oct. 26, 2023. [Online]. Available: https://files.core.ac.uk/pdf/23/140536.pdf

[24]
H. Xu *et al.*, “Robustness and Regularization of Support Vector Machines,” *Journal of Machine Learning Research*, vol. 10, pp. 1485–1510, 2009, Available: https://www.jmlr.org/papers/volume10/xu09b/xu09b.pdf

[25]
A. Tharwat, “Parameter investigation of support vector machine classifier with kernel functions,” *Knowledge and Information Systems*, vol. 61, no. 3, pp. 1269–1302, Feb. 2019, doi:

https://doi.org/10.1007/s10115-019-01335-4.

[26]
F. E. H. Tay and L. Cao, “Application of support vector machines in financial time series forecasting,” *Omega*, vol. 29, no. 4, pp. 309–317, Aug. 2001, doi: https://doi.org/10.1016/s0305-0483(01)00026-3.

[27]
L. J. Cao, K. S. Chua, W. K. Chong, H. P. Lee, and Q. M. Gu, “A comparison of PCA, KPCA and ICA for dimensionality reduction in support vector machine,” *Neurocomputing*, vol. 55, no. 1–2, pp. 321–336, Sep. 2003, doi: https://doi.org/10.1016/s0925-2312(03)00433-8.

[28]
D. B. V, “Nearest neighbor (NN) norms : NN pattern classification techniques,” *IEEE Computer Society Tutorial*, 1991, Available: https://cir.nii.ac.jp/crid/1572261550010307072

[29]
K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, “When Is ‘Nearest Neighbor’ Meaningful?,” *Lecture Notes in Computer Science*, pp. 217–235, 1999, doi: https://doi.org/10.1007/3-540-49257-7_15.