XJTLU Entrepreneur College (Taicang) Cover Sheet

Module code and Title	**DTS201TC Pattern Recognition**
School Title	**School of AI and Advanced Computing**
Assignment Title	**Coursework (Groupwork)**
Submission Deadline	**23:59 10th Dec.**
Final Word Count	
If you agree to let the university use your work anonymously for teaching and learning purposes, please type **"yes"** here.	Yes

I certify that I have read and understood the University's Policy for dealing with Plagiarism, Collusion and the Fabrication of Data (available on Learning Mall Online). With reference to this policy I certify that:

- My work does not contain any instances of plagiarism and/or collusion. My work does not contain any fabricated data.

By uploading my assignment onto Learning Mall Online, I formally declare that all of the above information is true to the best of my knowledge and belief.

Scoring – For Tutor Use	
Student ID	**1927932, 1928735, 1928620, 1929074, 1928715**

<table>
<tr><th rowspan="2">Stage of Marking</th><th rowspan="2">Marker Code</th><th colspan="3">Learning Outcomes Achieved (F/P/M/D) (please modify as appropriate)</th><th rowspan="2">Final Score</th></tr>
<tr><th>A</th><th>B</th><th>C</th></tr>
<tr><td>1st Marker – red pen</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td rowspan="2">Moderation

– green pen</td><td>IM Initials</td><td colspan="3">The original mark has been accepted by the moderator (please circle as appropriate):</td><td>Y / N</td></tr>
<tr><td></td><td colspan="3">Data entry and score calculation have been checked by another tutor (please circle):</td><td>Y</td></tr>
<tr><td>2nd Marker if needed – green pen</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="2">For Academic Office Use</td><td colspan="4">Possible Academic Infringement (please tick as appropriate)</td></tr>
<tr><td>Date Received</td><td>Days late</td><td>Late Penalty</td><td colspan="2">☐ Category A</td><td rowspan="5">Total Academic Infringement Penalty (A,B, C, D, E, Please modify where necessary) _____</td></tr>
<tr><td rowspan="4"></td><td rowspan="4"></td><td rowspan="4"></td><td colspan="2">☐ Category B</td></tr>
<tr><td colspan="2">☐ Category C</td></tr>
<tr><td colspan="2">☐ Category D</td></tr>
<tr><td colspan="2">☐ Category E</td></tr>
</table>

DTS201TC Coursework report

(Dec. 13th, 2021)

Shengzhi Tian, Tianyue Jiang, Yang Deng, Yabin Xu, Jiawen Gao

Content

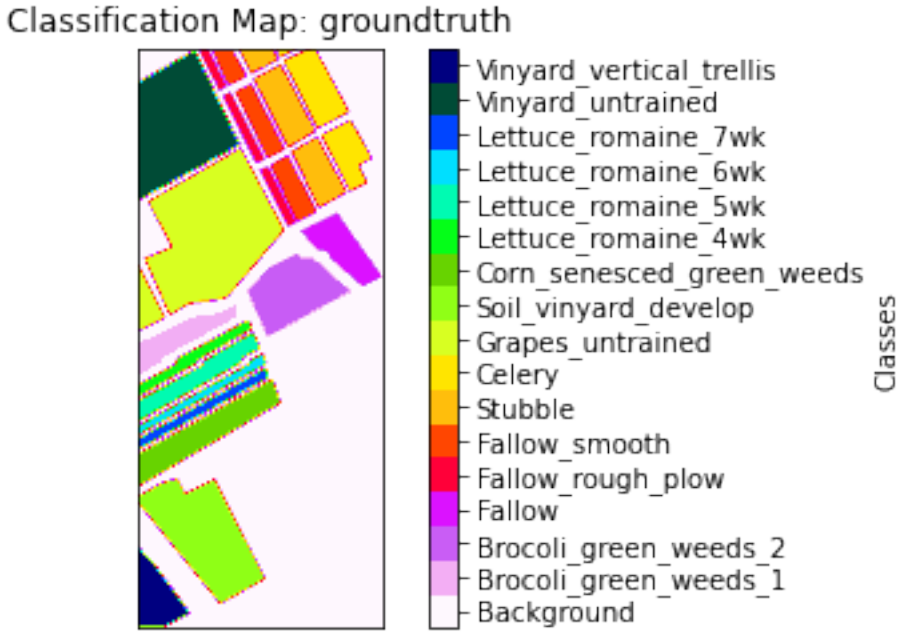
1. Introduction

Pattern recognition-based methods are highly successful in hyperspectral image analysis tasks because they are able to automatically learn the relationship between reflectance spectra and the desired information, while being robust to noise and uncertainty in spectral and ground truth measurements. The aim of this report is to classify the hyperspectral remote sensing dataset Salinas using different pattern recognition models and to evaluate each model for comparison and analysis. The models covered are Gaussian Bayes (MAP), Gaussian Bayes (MLE), Random Forest (RF), Support Vector Machine (SVM), and k-Nearest Neighbour (KNN). A survey analysis of the Salinas dataset, including a description of the pre-processing of the data, is shown in Section 2. In Section 3 the implementation of the different models is shown. In Section 4, the different models are compared and analysed. Section 5 concludes the report.

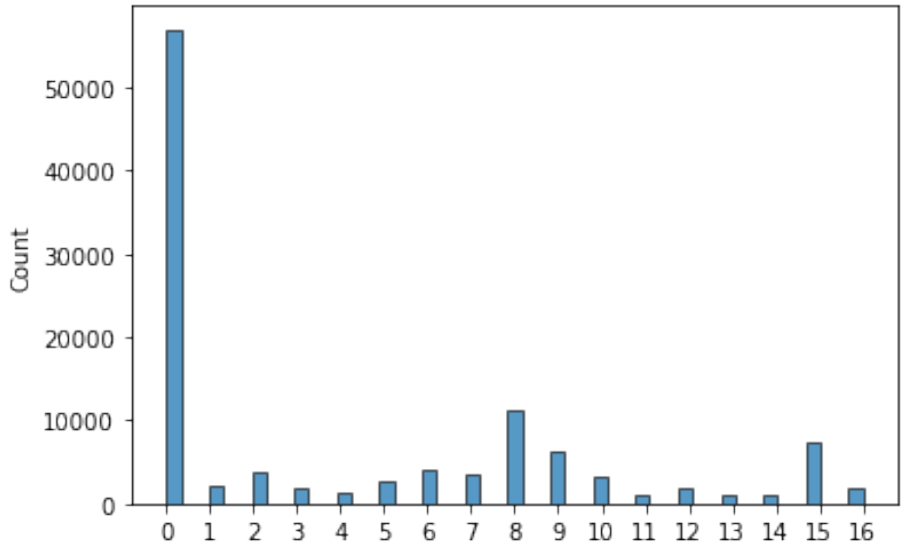
2. Data

2.1. Dataset Description

Hyperspectral image' pixels consist of a certain number of bands' value which serve as features of objects in the image. In the Salinas scene, there are 224 bands and 20 of them are neglected. There are 512*217 training examples contained in 'Salinas_corrected.mat' file, and their corresponding 16 labels contained in 'Salinas_gt.mat'. In total, the dataset consists of more than 111 k pixels of which 54,129 have been labeled with respect to 16 classes.
The classification map with labels is as followed.

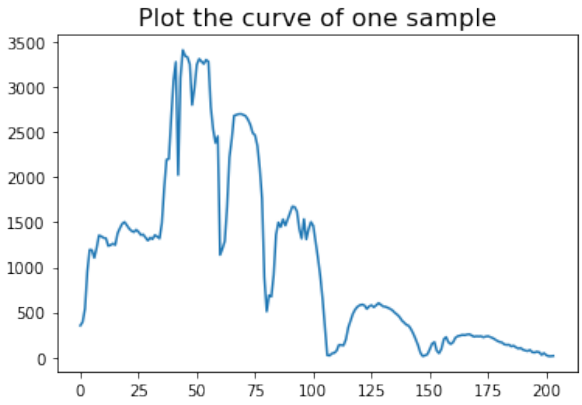


The count of pixels of different bands is as followed, it is obvious that the background has the most pixels.

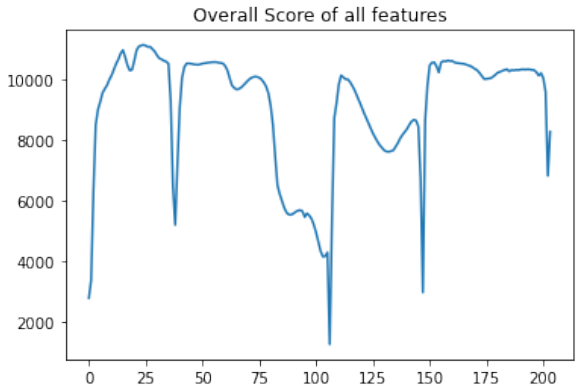


2.2. Feature Visualization

1. Plot the curve of one sample:



2. Overall Feature Score:

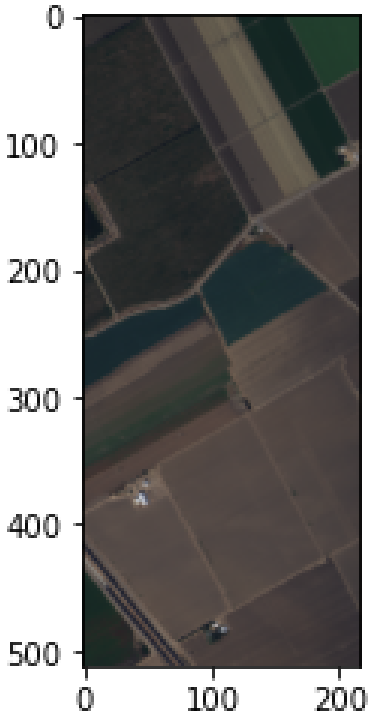


In all the 204 bands of Salinas hyperspectral image, we've plotted the overall score for each band via SelectKBest() in sklearn.feature_selection. Broadly, it can be seen that different features will have different impacts on the classification result. For example, features between 80 and 110 seems to have a low relevance compared with the features between 150 and 175. However, this technique can only be used to visualize the features rather than to conduct a feature extraction since these feature scores are too general and part of them are too similar.

3. Display a Composite Color Image

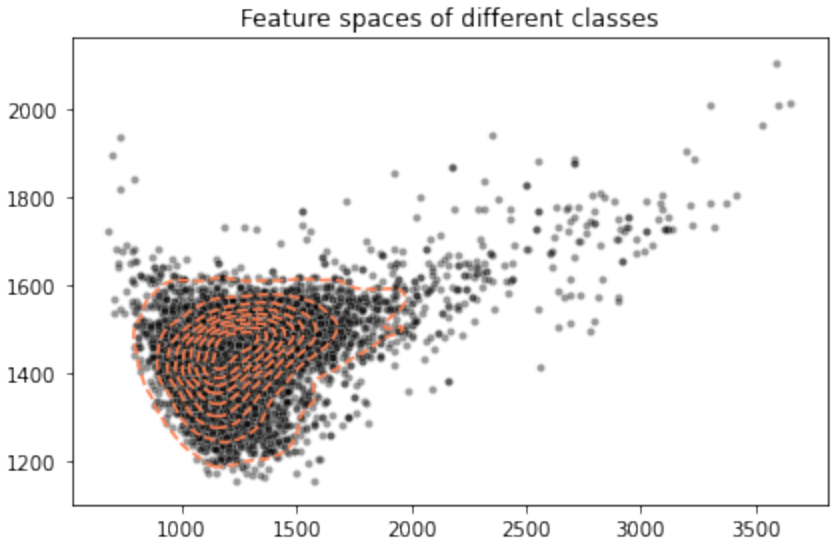
We've randomly selected three bands and displayed a rgb composite color image which is shown as follows:

```
ImageView object:
  Display bands       :  (29, 19, 9)
  Interpolation       :  <default>
  RGB data limits     :
    R: [2.0, 8336.0]
    G: [59.0, 9171.0]
    B: [86.0, 8209.0]
```



2.3. Comparison of the features space in different classes

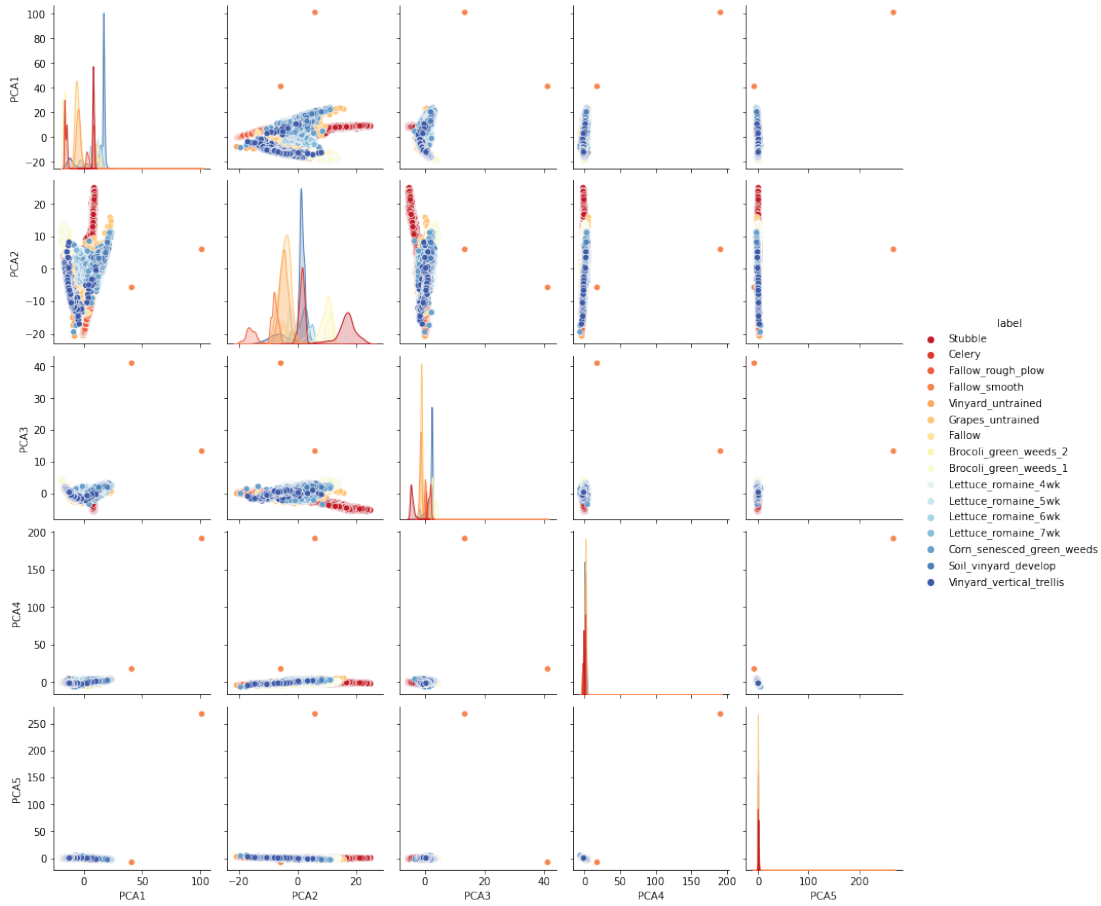
1. Feature spaces of different classes



This diagram has illustrated the feature space against band 33 and 99. Meanwhile, detailed explanation of the feature space will be illustrated in the following parts.

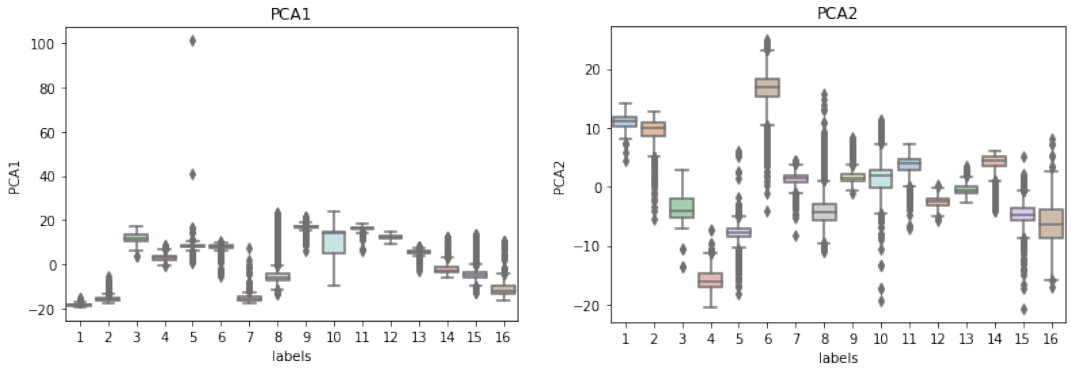
2. Pairwise Plot of Feature Space After PCA

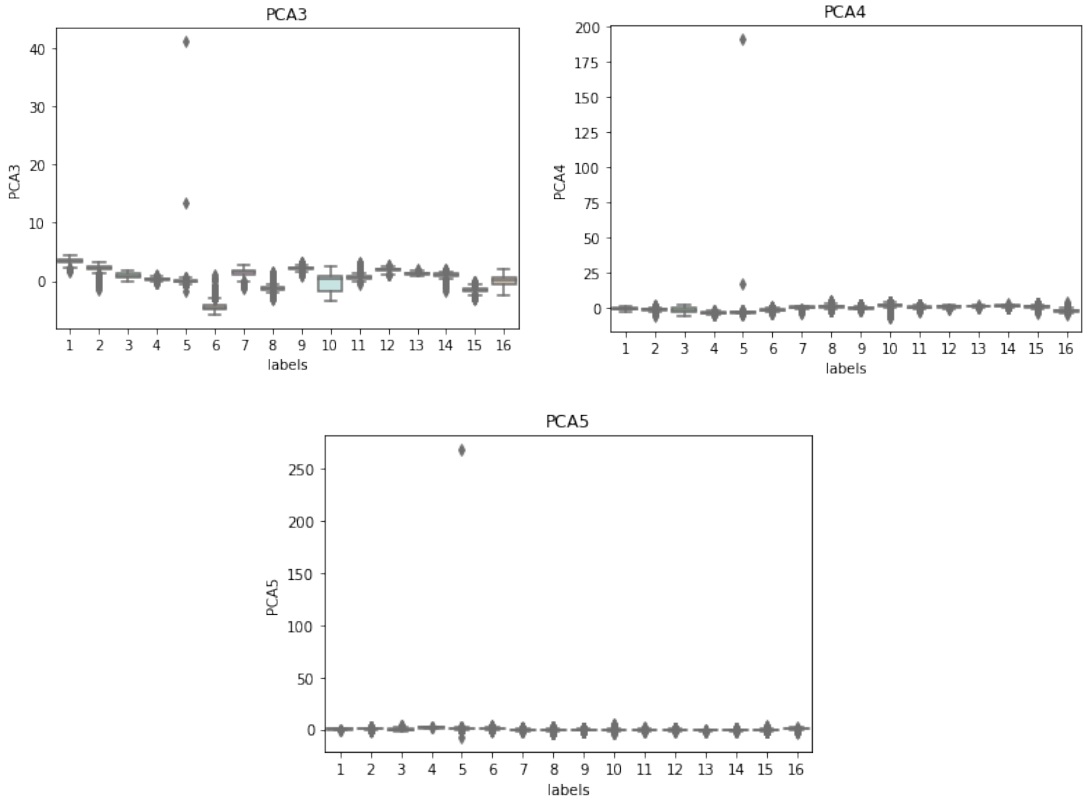
Scatter plots of each training examples against different groups of features in 2-D space are presented below. They can reveal differences between each features combinations' classification abilities. For example, the scatter plot against {PCA{1}, PCA{2}} has more separated color distribution than {PCA{1}, PCA{3}}. It can indicate that PCA{1} and PCA{2} contribute more to the classification task.



3. Boxplot of Feature Space After PCA

It can also show differences between each class's distribution depending on different features. Additionally, statistic information including mean values, quartiles and outliers can be clearly visualized.





2.4. Data Prepossessing

2.4.1. Transform the 3D-image Dataset into 2D Structural Data

Since the classification of hyperspectral image is based on each individual pixel, the original 3D-image dataset can be firstly reshaped to 2D structural data so as to enhance computational efficiency.

2.4.2. Standard Scaling

Since the objective function of a machine learning algorithm usually assumes that all features should look like a standard normal distribution, such as the rbf kernel in SVM, the dataset is highly suggested to be standardized before the model fitting process. In order to compare with standard scaler, we also tried the MinMax Scaler to normalize our data to (0, 1), however the result has shown that Standard Scaler outperformed MinMax Scaler in terms of prediction performance.

2.4.3. Remove the Background

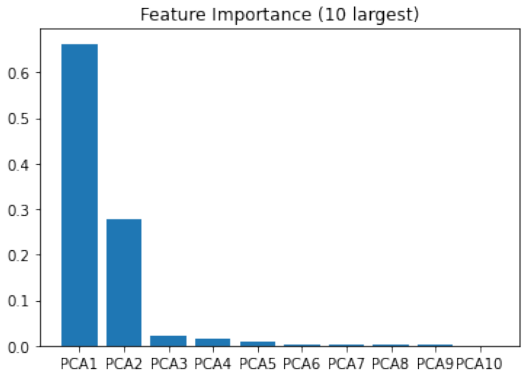
The Salinas dataset contains a large number of background image, specifically 56975

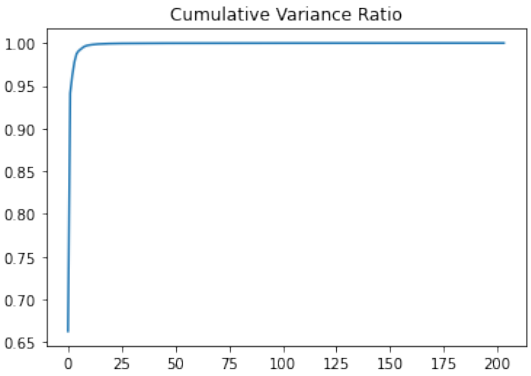
pixels, which are irrelevant to the classification scenario. Therefore, the background can be removed preliminarily to reduce the computational cost in the model implementation process.

2.4.4. Principle Component Analysis (PCA)

In the context of hyperspectral image classification, HSI usually contains a large dataset of high dimensionalities which is usually tough for some machine learning models to interpret and process. In order to solve this problem, the dimensionality of such dataset is supposed to be drastically reduced. Therefore, Principle Component Analysis, short for PCA, is a dimensionality reduction method via preserving principle features that cover the variance of the statistical information as much as possible, which can be applied to the Salinas problem here.

Therefore, as for all our five models, we've utilized the PCA technique for the purpose of feature extraction and dimension reduction. Basically, the variance of data that each principle component captures can be illustrated in the following two figures:





Therefore, we've set a threshold for the cumulative variance ratio and select 5 principle components which cover 99.0% of the data variance, and thus the shape of Salinas dataset (without background) can be reduce drastically to (54129, 5).

3. Models

3.1. Gaussian Bayes (MAP)

3.1.1. Model Description:

The Naive Bayes Classifier, derived from classical mathematical theory, uses bayes theorems to predict the likelihood that a sample of an unknown category will fall into

each category, choosing one of the most likely categories as the final category for that sample. In the simple Bayesian classification model, it will establish a function that obeys the normal distribution for the characteristic vectors of each category, and given the training data, the algorithm will estimate the vector mean and variance matrix for each category and then make predictions based on these. In Bayesian classification, the Gaussian model is suitable for dealing with continuous type feature variables for prediction. When using this model, the features are assumed to belong to a Gaussian distribution and then the mean and variance of the features are calculated based on training samples so that the prior probability of each attribute value under this feature can be obtained. The parameter estimation of this model is Maximum a Posteriori estimation (MAP), it is to estimate the parameter in order to maximum the $P(x|y_i)P(y_i)$.

3.1.2. Model Principles:

According to Bayes Theory:

$$P(y_i|x) = P(x|y_i)P(y_i)/P(x)$$
$$P(y_i|x) = \frac{P(x|y_i)P(y_i)}{\sum_1^m P(x|y_i)P(y_i)}$$

In this dataset, we have 16 classes, and we are going to calculate all p(y1|x) to p(y16|x), and get the max of p(yi|x) as the prediction of the label.

$$y = \arg\max_i P(y_i|x)$$

As the $P(x)$ is the same between different $P(y_i|x)$, therefore, it can be ignored when maximum the $P(y_i|x)$.

$$P(y_i|x) \propto P(x|y_i)P(y_i)$$

In Gaussian Naïve Bayes algorithm for classification, the likelihood of the features is assumed to be Gaussian.

$$P(y_i|x) \propto \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_i-\mu)^2}{2\sigma^2}\right) P(y_i)$$

The prior probability of each label can be calculated by number of (y_i)/ number of total labels.
The likelihood function can be calculated by the function multivariate_normal in the library of scipy.stats
Finally, after getting all the predicted labels, we are going to calculate the accuracy.

3.1.3. Model Implementation:

Feature selection:
We use PCA to select the feature and reduce the dimension. It is showed that when the n_component is 5, the 99.9% variance of data has been captured.

Parameters:
The parameters mean and covariance of data and prior probability of the label are calculated according to the train data.

```
mu_class = np.mean(item,0)
cov_class = np.cov(item, rowvar=False)
prior_i = (item.shape[0]*item.shape[1])/len(label)
```

Platform:
This model is implemented on the personal CPU
CPU: 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz 2.80 GHz
RAM: 16 GB

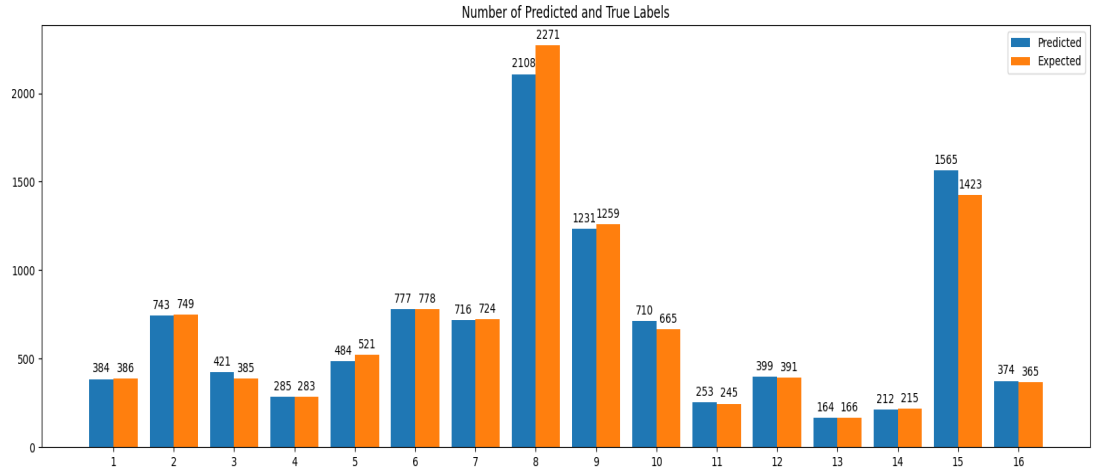
Parameter estimation:
The parameter estimation is MAP.

3.1.4. Evaluation:

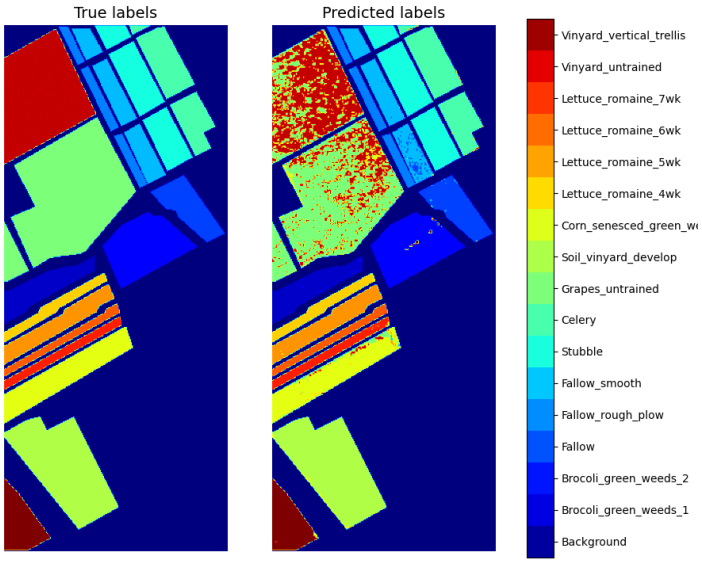
Accuracy:
we compute the final accuracy through cross validating. And get the mean of accuracy output by different validations.

	precision	recall	f1-score	support
1	1.00	0.99	1.00	386
2	1.00	0.99	0.99	749
3	0.91	0.99	0.95	385
4	0.99	1.00	1.00	283
5	0.99	0.92	0.96	521
6	1.00	1.00	1.00	778
7	1.00	0.99	0.99	724
8	0.81	0.75	0.78	2271
9	1.00	0.97	0.98	1259
10	0.83	0.89	0.86	665
11	0.95	0.98	0.97	245
12	0.98	1.00	0.99	391
13	0.99	0.98	0.99	166
14	0.98	0.96	0.97	215
15	0.66	0.73	0.69	1423
16	0.96	0.98	0.97	365
accuracy			0.89	10826
macro avg	0.94	0.95	0.94	10826
weighted avg	0.90	0.89	0.89	10826

Classification visualization:



It can be seen that the 8 and 15 label are predicted with higher error.



Cross Validation:

```
The accuracy after cross validation is: 0.8934948090030801
```

3.2. Gaussian Bayes (MLE)

3.2.1. Model Description

The Naive Bayes Classifier is derived from classical mathematical theory and uses Bayes' theorem to predict the likelihood of a sample of an unknown category falling into each category, selecting the most likely category as the final category for that sample. In a simple Bayesian classification model, a function obeying a normal distribution is created for the eigenvectors of each category, and given training data,

the algorithm estimates the vector mean and variance matrix for each category and makes predictions accordingly. In Bayesian classification, the Gaussian model is suitable for dealing with continuous type feature variables for prediction. When using this model, the features are assumed to belong to a Gaussian distribution and then the mean and variance of the features are calculated based on training samples so that the prior probability of each attribute value under this feature can be obtained. Maximum likelihood estimation (MLE) is a common estimation method used by the frequency school to evaluate the model parameters given the observed data, i.e.: "the model is fixed, the parameters are unknown". The maximum likelihood estimation is used to obtain the mean and variance of the Gaussian distribution in the hypothesis.

3.2.2. Model Principles

According to Bayes Theory:

$$P(y_i|x) = P(x|y_i)P(y_i)/P(x)$$
$$P(y_i|x) = \frac{P(x|y_i)P(y_i)}{\sum_1^m P(x|y_i)P(y_i)}$$

In this dataset, we have 16 classes, and we are going to calculate all p(y1|x) to p(y16|x), and get the max of p(yi|x) as the prediction of the label.

$$y = \arg\max_i P(y_i|x)$$

In Gaussian Naïve Bayes algorithm for classification, the likelihood of the features is assumed to be Gaussian.

$$P(y_i|x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_i-\mu)^2}{2\sigma^2}\right)$$

The parameters are estimated using MLE. Assume that the data $x_1, x_2, \ldots, x_n$ is a set of independent identically distributed samples, $X = (x_1, x_2, \ldots, x_n)$. Then the MLE estimate of θ can be derived as follows:

$$\begin{aligned}\hat{\theta}_{MLE} &= argmax P(X;\theta) \\ &= argmax P(x_1;\theta)P(x_2;\theta)\ldots P(x_n;\theta) \\ &= argmax log \prod_{i=1}^{n} P(x_i;\theta) \\ &= argmax \sum_{i=1}^{n} log P(x_i;\theta)\end{aligned}$$

In the calculation process, to facilitate the calculation, often take the logarithm of the likelihood function first, and then find the derivative to calculate the extreme value point; if it is impossible to find the derivative, to use the principle of great likelihood to solve.

Once all the parameters have been estimated, the conditional probability $P(x|y_i)$ for a given sample can be calculated, and $P(y_i)P(x|y_i)$ can then be calculated, after which the sample category can be determined and the model prediction completed.

3.2.3. Model Implementation

Parameters

The parameters mean and covariance of data are calculated according to the train data.

```
mu_class = np.mean(item, 0)
cov_class = np.cov(item, rowvar=False)
```

MLE estimates of the mean and covariance of the Gaussian distribution. Derive the joint probability function (or joint density) of the sample from the overall distribution. The likelihood function $L(\theta) = \prod_{i=1}^{n} P(x_i; \theta)$ is obtained by considering the independent variable in the joint probability function (or joint density) of the sample as a constant, and the parameter θ as the independent variable. The estimate of the mean $\mu_{i,c}$ is the mean of all X_i in the sample category c. The estimate of the covariance $\sigma^2{}_{i,k}$ is the covariance of all X_i in the sample category c. For a continuous sample value, its probability distribution can be found by bringing it into a Gaussian distribution.

Platform

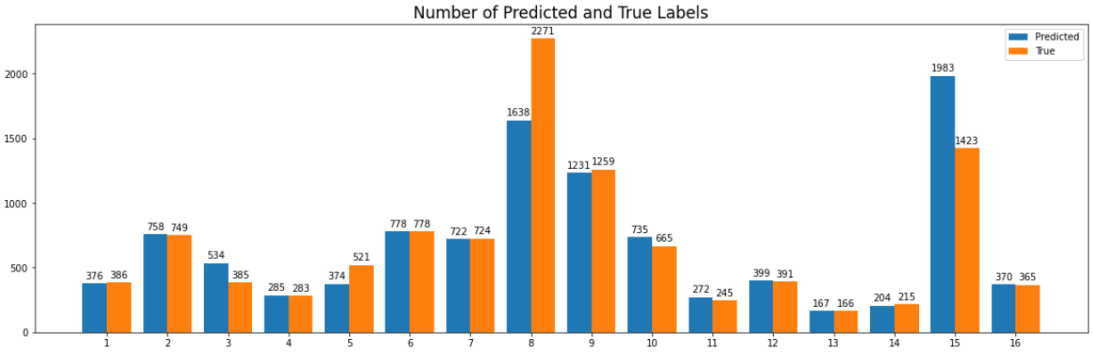
CPU: AMD Ryzen 7 5700U with Radeon Graphics 1.80 GHz
RAM: 16.0 GB

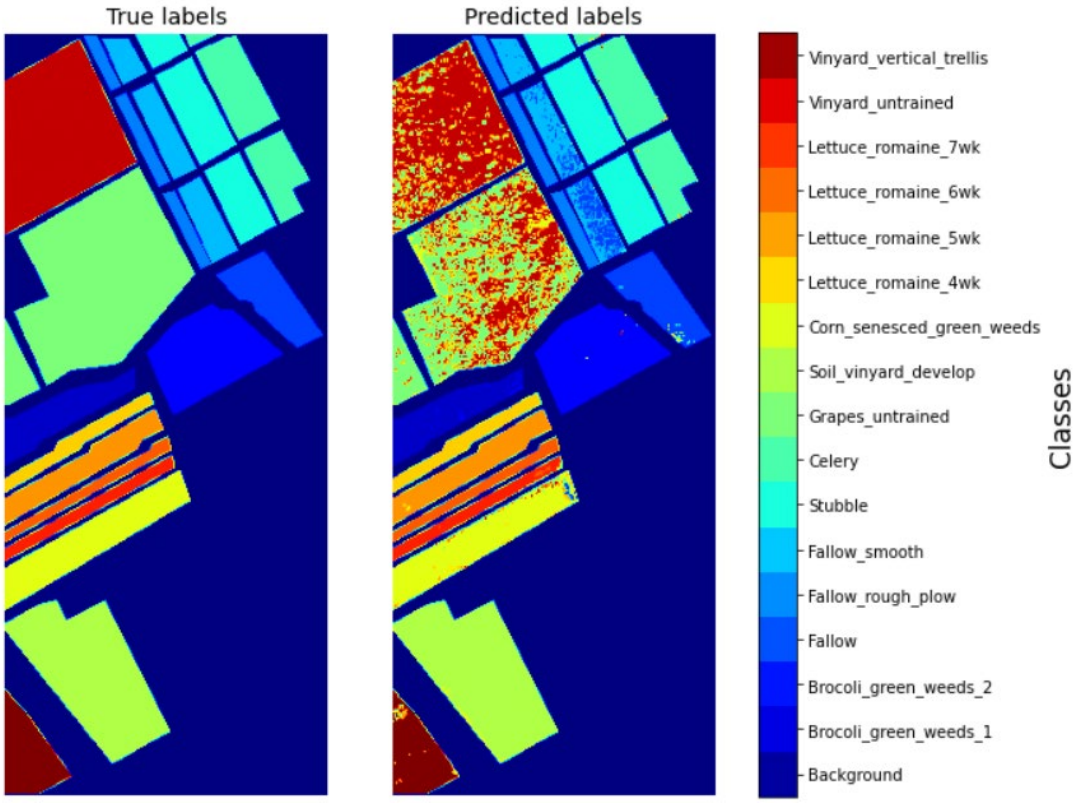
3.2.4. Evaluation

Accuracy:

	precision	recall	f1-score	support
1	1.00	0.97	0.98	386
2	0.99	1.00	0.99	749
3	0.71	0.98	0.82	385
4	0.99	1.00	1.00	283
5	1.00	0.72	0.84	521
6	1.00	1.00	1.00	778
7	0.99	0.99	0.99	724
8	0.82	0.59	0.69	2271
9	1.00	0.97	0.98	1259
10	0.80	0.88	0.84	665
11	0.89	0.98	0.93	245
12	0.97	0.99	0.98	391
13	0.97	0.98	0.97	166
14	0.98	0.93	0.95	215
15	0.57	0.80	0.66	1423
16	0.95	0.97	0.96	365
accuracy			0.86	10826
macro avg	0.91	0.92	0.91	10826
weighted avg	0.88	0.86	0.86	10826

Classification result visualization:





Thorough model validation

```
Cross validation scores: 85.77498614446702
```

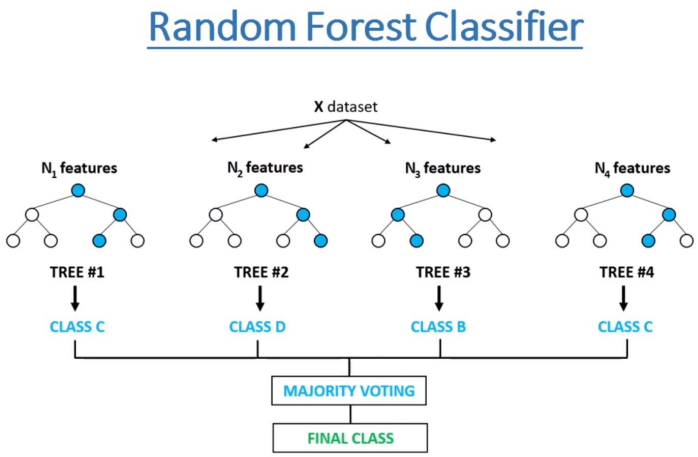
3.3. Random Forest

3.3.1. Model Description

Generally, A random forest classifier is a supervised machine learning method constructed from decision tree algorithm that is widely used in classification scenarios. The random forest applies a general technique called 'bagging' to wrap massive decision trees into an ensemble where each individual tree will make a decision for

the incoming sample's class and the class with the most votes become the model's final prediction. Since all individual trees are trained by random selected subsets of the training data with replacement (bootstrapping technique), the correlation of predictions between different trees will be relatively low so that the generality of the model can be enhanced.

Specifically, as for an individual decision tree in the forest, it is a hierarchical tree structure that is constructed by a randomly chosen subset of features. The split of each node is based on a measurement of the likelihood of an incorrect classification of its subnodes, called Gini Impurity, which will detailed illustrated in the next section. In addition, unlike tradition decision trees, the trees in a random forest have no need to be pruned and grow to the largest since the randomness of random forest has guaranteed that this model is hard to overfit.



3.3.2. Model Principles

For each tree in the forest, a single parent node will have a test function to classify the incoming data. Basically, the test function for splitting the node associates with Gini impurity, which measures the homogeneity in a data sample while a sample is homogeneous if all data are from the same class. Mathematically, Gini impurity can be denoted as:

$$Gini\ impurity = \sum_{i}^{n} p_i\,(1 - p_i) = 1 - \sum_{i=1}^{n} p_i^2$$

$$(n\text{: } number\ of\ classes, pi\text{: } success\ probability\ of\ each\ class)$$

The best split at each node is chosen via minimizing the weighted sum of Gini impurities of its subnodes:

$$Weighted\ Gini\ impurity = \sum_{i}^{n} \omega_i * p_i\,(1 - p_i)$$

$$(\omega_i: class\ weight, corresponding\ to\ observations)$$

As for the Salinas problem here, since a hyperspectral image usually has a large dataset with high dimensionalities (bands), the random forest classifier can be appropriately applied in this context due to its randomness of feature selection in training each individual tree. In addition, situations such as an unbalanced dataset can usually happen due to different spatial information when classifying hyperspectral images. Therefore, the random forest classifier can be used in such scenarios since it can adjust the weight of unbalanced classes when calculating the weighted Gini impurity so as to enhance greater performance.

3.3.3. Model Implementation

Parameter Explanation:
Basically, two specific parameters are chosen in this Random Forest Classifier:

n_estimators: This parameter refers to the number of decision trees in this forest. A higher number of estimators can increase the model's stability of performance and usually it is also positively related to the performance.

max_features: This parameter refers to the number of features that an individual tree in the forest randomly selects from the training data. Generally, selecting more features in a tree will increase its individual performance while, in turn, results in a higher correlation between different trees, reducing the overall model generalization ability, which may affect the model's classification performance.

Therefore, these two parameters have a fairly distinct impact on the performance of the Random Forest model, and thus we've used the GridSearchCV() in sklearn.model_selection to choose the best parameter according to the cross validation score.

The parameter grid that we want to conduct a grid search on is shown as follows:

```
params_grid = {
    'n_estimators': [50, 100, 200, 300],
    'max_features': [3, 4, 5]
}
```

A 5-fold cross validation will be used to score each pair of these parameters and the result is shown as below:

	param_max_features	param_n_estimators	mean_test_score
0	3	50	0.922384
1	3	100	0.922107
2	3	200	0.922661
3	3	300	0.922500
4	4	50	0.920791
5	4	100	0.921507
6	4	200	0.922523
7	4	300	0.922153
8	5	50	0.919890
9	5	100	0.920075
10	5	200	0.920814
11	5	300	0.921045

Therefore, according to the mean cross validation score of each pair, the best parameter for our model is {'max_features': 3, 'n_estimators': 200}, and thus we've chosen this pair of parameters in the model implementation.

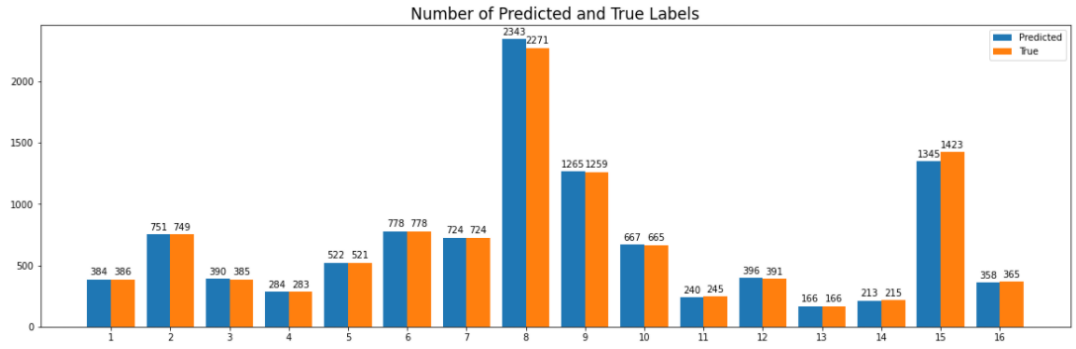
Platform:
CPU: Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz 2.59 GHz
RAM: 16GB

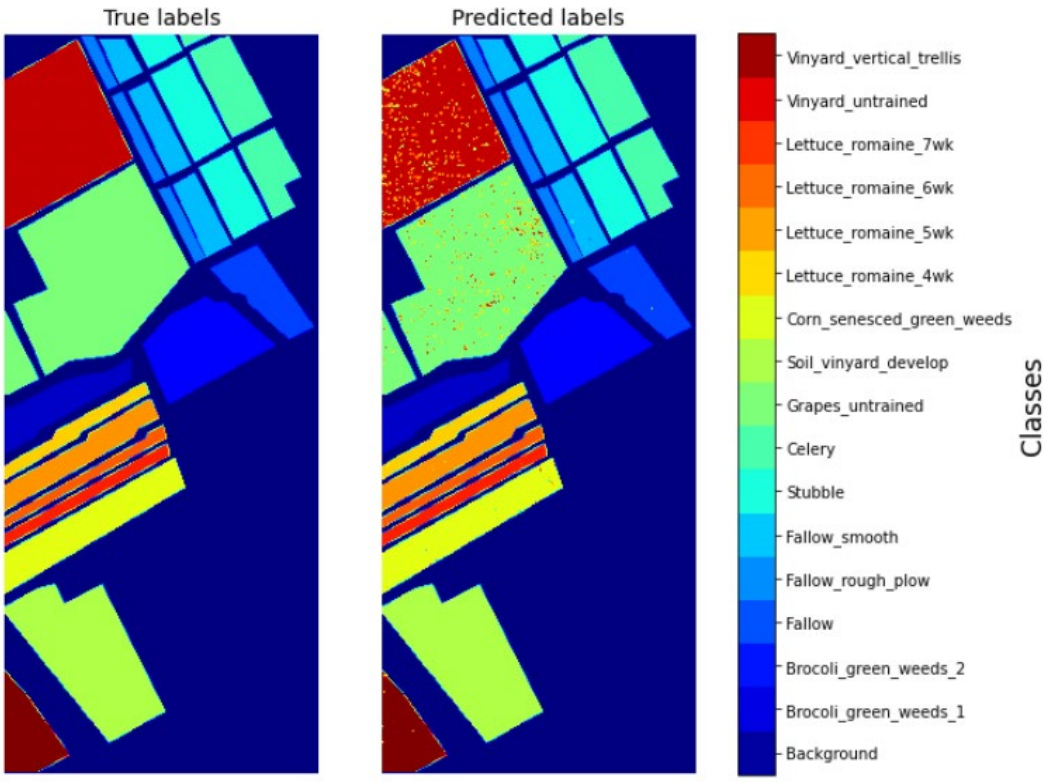
3.3.4. Evaluation

Accuracy:

	precision	recall	f1-score	support
1	0.99	0.99	0.99	386
2	0.99	1.00	1.00	749
3	0.98	0.99	0.99	385
4	1.00	1.00	1.00	283
5	0.99	0.99	0.99	521
6	1.00	1.00	1.00	778
7	1.00	1.00	1.00	724
8	0.83	0.86	0.84	2271
9	0.99	1.00	0.99	1259
10	0.95	0.95	0.95	665
11	0.97	0.95	0.96	245
12	0.98	0.99	0.99	391
13	0.99	0.99	0.99	166
14	0.99	0.98	0.99	215
15	0.77	0.73	0.75	1423
16	0.98	0.96	0.97	365
accuracy			0.93	10826
macro avg	0.96	0.96	0.96	10826
weighted avg	0.93	0.93	0.93	10826

Classification visualization:





Cross Validation:

```
Fitting 5 folds for each of 12 candidates, totalling 60 fits
Best parameters:  {'max_features': 3, 'n_estimators': 200}
Mean cross validation score of best estimator:  0.9226612888387941
```

3.4. SVM

3.4.1. Model Description

SVM is supervised learning model developed at AT&T Bell Laboratory. In the classification problem contexts, it is implemented as a non-probabilistic model which intends to construct a proper hyperplane separating the feature space into different regions.

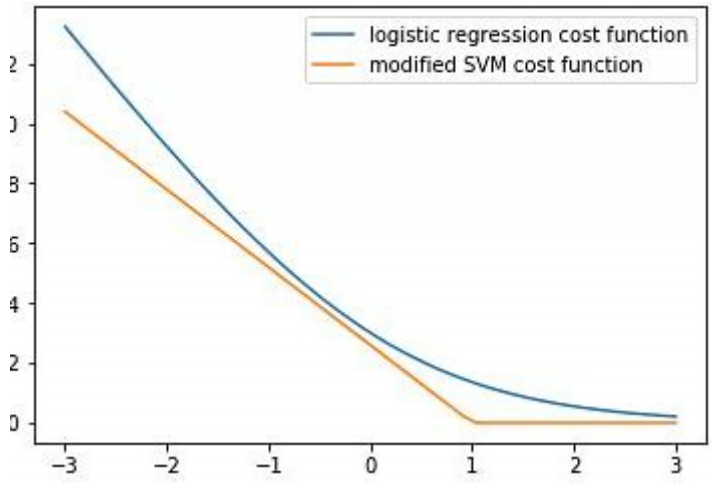
3.4.2. Model Principles

Optimization objectives
The cost function of SVM:

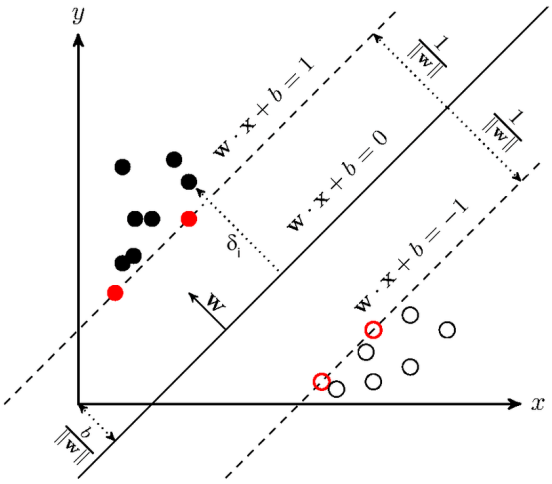
$$(\theta)_i = C\big(ycost_1(\theta^T X) + (1-y)cost_0(\theta^T X)\big) + \frac{1}{2}\sum \theta^2$$

where θ_i denotes the parameter for the classifier that classify the i[th] class.

A simple plot of $cost_1\,(\theta^T X)$:



Based on the above information, it is discovered that the cost function will penalize the misclassification of the input variable X which belong to the i_{th} class unless $\theta^T X >= 1$. To further explain this statement, a geometrical description of $\theta^T X$ is illustrated below



where $\theta^T X = d\|w\|$. The term d represents the distance between the targeted point and the hyperplane. Thus, SVM tries to find a hyperplane maximizing the margin that bounds the points in the feature space.

Kernel:

To perform SVM as a non-linear classifier, the kernel functions are used to map original features in to a new feature space where a hyperplane operates as a linear classifier. Their overall implementation is shown below:

$$X = K(x, l)$$

The term l is regarded as a 'landmark'. Its similarity between the input variable x is measured by the kernel function, which produces the new feature X. One commonly used kernel function is Gaussian kernel, which is illustrated below:

$$X = \exp\left(-\frac{\|x - l\|}{\partial^2}\right)$$

It is applied into classification of the Salinas data considering the training data may be not linear separable.

3.4.3. Model Implementation

Choosing parameters:

It is necessary to consider the trade-off between the variance and bias when choosing proper values of the regularization parameter C in the cost function. If C is too small, there might exist problems of underfitting with high bias. If C is too large, the problem of overfitting may occur with high variance. Therefore, the trade-off serves as the principle of choosing the parameter C. Its original value is set to the default. The model's performances on the test set were observed each time the value of C is increased in order to avoid the overfitting. Finally, it is found that the model performs well on the test set if C is set to around 100.

Platform:

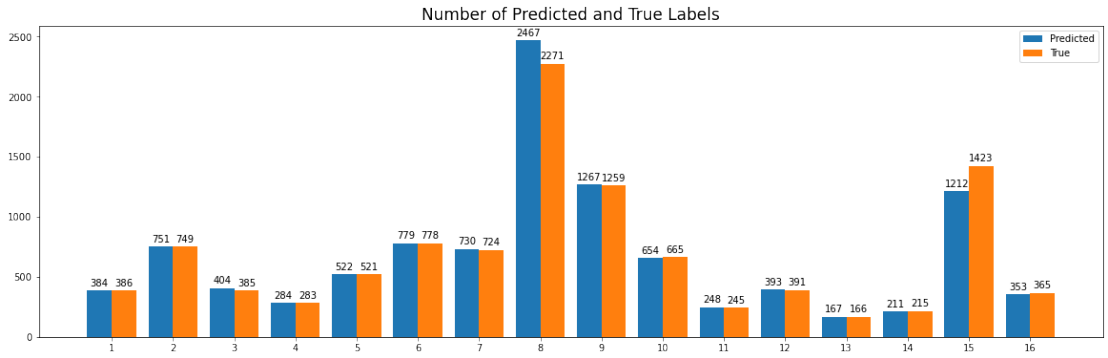
CPU: Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz 1.99 GHz
RAM: 8.00 GB

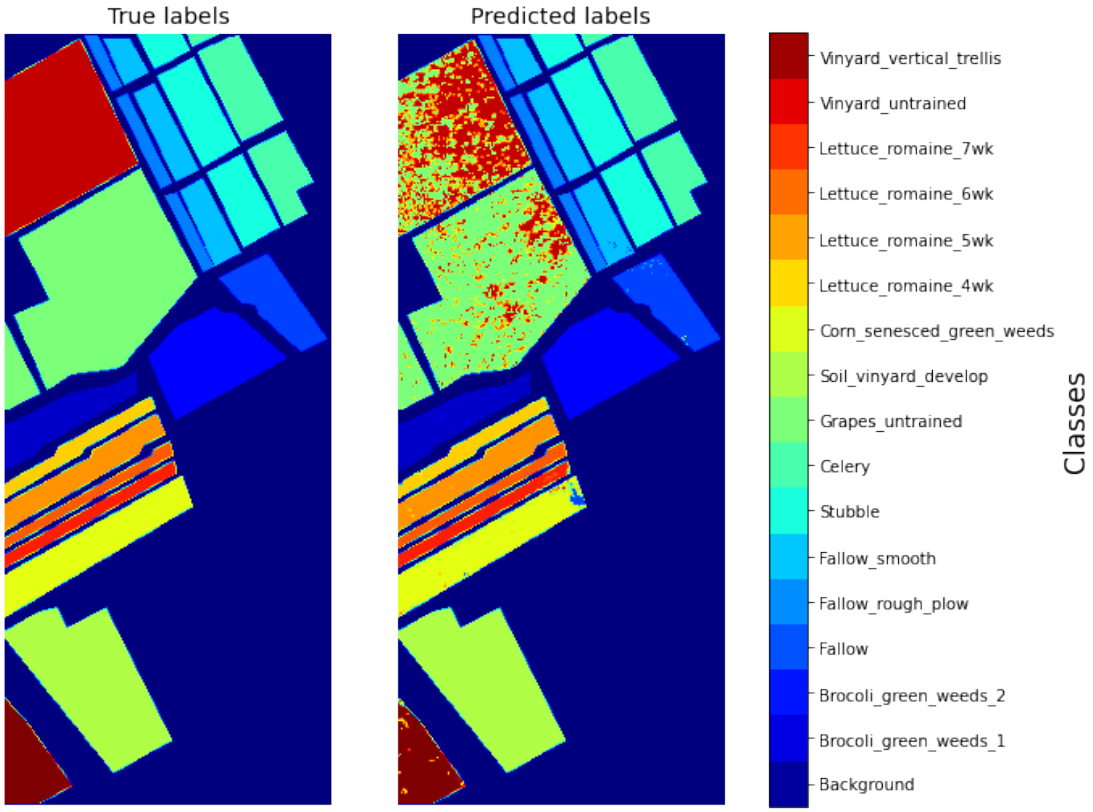
3.4.4. Evaluation

Accuracy:

	precision	recall	f1-score	support
class1	0.99	0.99	0.99	386
class2	0.99	1.00	1.00	749
class3	0.94	0.99	0.97	385
class4	1.00	1.00	1.00	283
class5	0.99	0.99	0.99	521
class6	1.00	1.00	1.00	778
class7	0.99	1.00	0.99	724
class8	0.77	0.84	0.80	2271
class9	0.99	1.00	0.99	1259
class10	0.93	0.91	0.92	665
class11	0.96	0.97	0.96	245
class12	0.99	0.99	0.99	391
class13	0.99	0.99	0.99	166
class14	0.99	0.97	0.98	215
class15	0.72	0.61	0.66	1423
class16	0.99	0.95	0.97	365
accuracy			0.90	10826
macro avg	0.95	0.95	0.95	10826
weighted avg	0.90	0.90	0.90	10826

Classification result visualization:





3.5. KNN

3.5.1. Model Description

KNN is one of the machine learning algorithms. Identify the known training sample, and then train to get the unknown sample classification. For unknown samples, K nearest neighbors in the training set are found according to the calculated distance. KNN algorithm is a kind of supervised learning, which is known to classify K nearest samples. According to the category of the majority of the k nearest samples, which category is judged. Since KNN mainly relies on the surrounding samples, rather than the method of discriminating the class domain to determine the category, KNN is

more suitable for the samples to be divided with more overlap or crossover in the class domain compared with other methods.

3.5.2. Model Principles

1. Calculate the distance between test data and each training data;

   KNN calculates adjacent pixel labels of Salinas hyperspectral images.

2. In KNN, the distance between objects is calculated as the dissimilarity index between objects.

   In two dimensions, use the Euclidean distance formula, the distance between two points A and B

   $$\sqrt{(a_1 - b_1)^2 - (a_2 - b_2)^2}$$

   Extend to multiple dimensions

   $$D = \sqrt{\sum_{i}^{n} (a_i - b_i)^2}$$

3. Sort according to increasing distance.

4. Select K points with the smallest distance;

   In the problem, there were 16 label classes, and 5 of the k most similar (that is, the closest neighbor of the feature space) were selected for classification.

5. Determine the occurrence frequency of the category of the first K points;

   P(X) the density function of the n sample.

   $$P_n(x) = \frac{k_n/n}{V_n}$$

6. K-Nearest Neighbor Density Estimation. Sort each observation by its distance from x, $R_k(x)$ denotes the distance from x to its k^{th} nearest neighbor point.

   $$P_n(x) = \frac{k}{n} \frac{1}{V_d \cdot R_k^d(x)}$$
   $$= \frac{k}{n} \frac{1}{Volume\ of\ a\ d - dimentional\ ball\ with\ radius\ being\ R_k(x)}$$

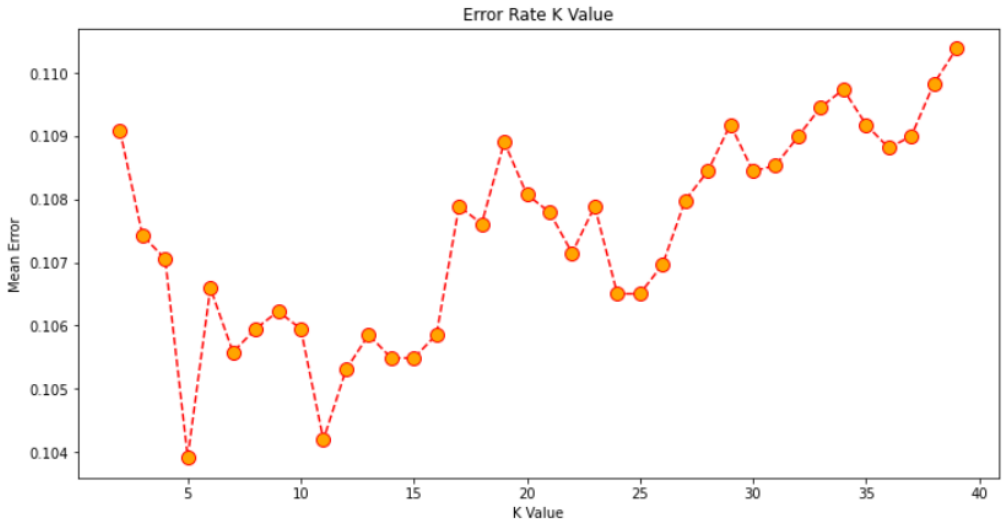
7. The category with the highest frequency in the first K points is returned as the predicted classification of the test data.

3.5.3. Model Implementation

The parameter:

X=None Find the nearest neighbor target sample

n_neighbors=None The number of nearest neighbor samples of the target sample



Calculating error for K values between 2 and 40. We can find K = 5 are more propriate. The error rate for spotting K goes down and then goes up, because there are more samples around.

Platfom:

CPU: Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz 1.99 GHz

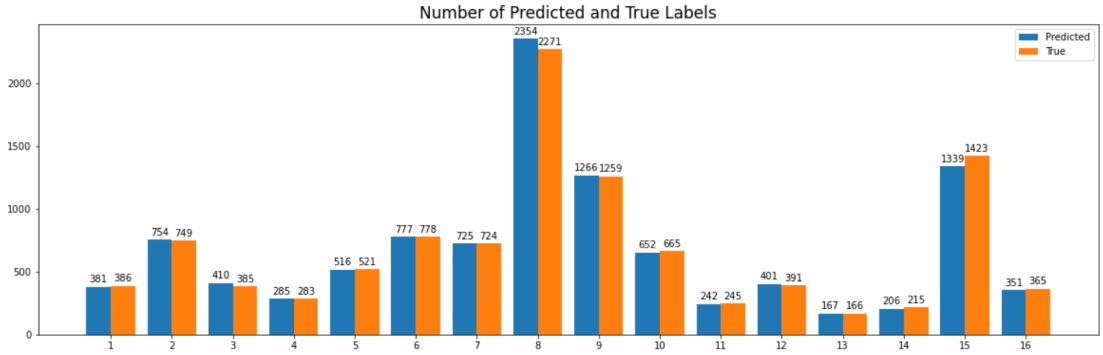
RAM: 8.00 GB

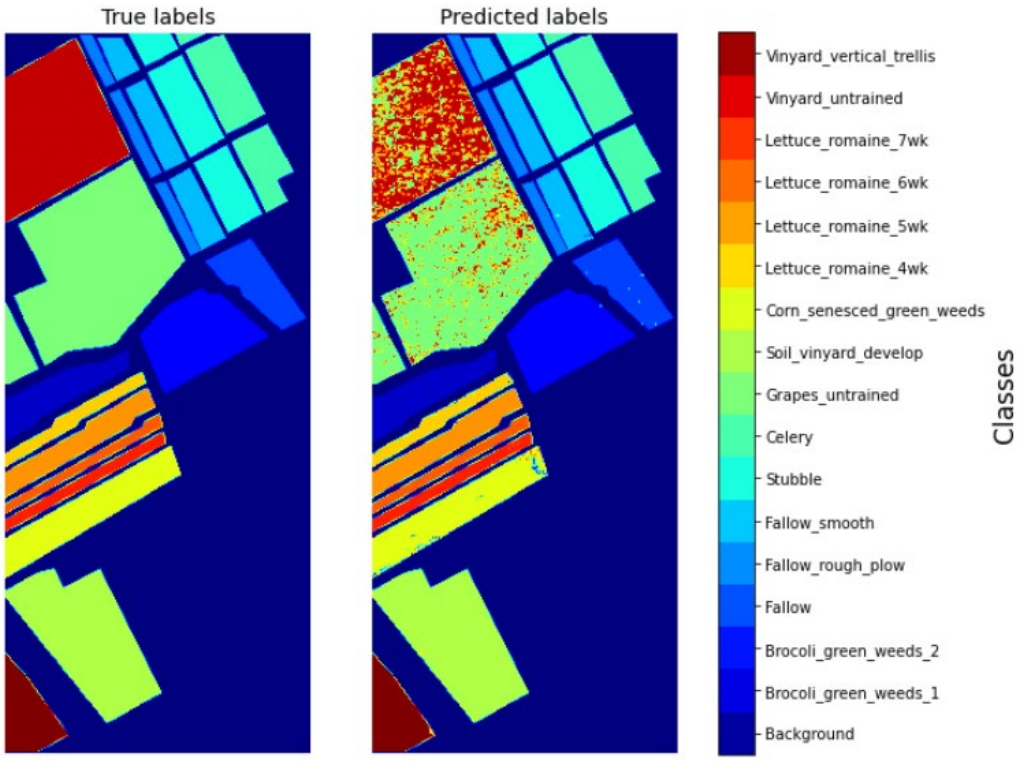
3.5.4. Evaluation

Accuracy

	precision	recall	f1-score	support
1	1.00	0.99	0.99	386
2	0.99	1.00	1.00	749
3	0.93	0.99	0.96	385
4	0.99	1.00	1.00	283
5	0.99	0.98	0.98	521
6	1.00	1.00	1.00	778
7	0.99	0.99	0.99	724
8	0.77	0.80	0.78	2271
9	0.99	1.00	0.99	1259
10	0.93	0.91	0.92	665
11	0.96	0.95	0.96	245
12	0.97	1.00	0.98	391
13	0.98	0.99	0.98	166
14	0.99	0.94	0.96	215
15	0.67	0.63	0.65	1423
16	0.99	0.95	0.97	365
accuracy			0.90	10826
macro avg	0.95	0.94	0.95	10826
weighted avg	0.90	0.90	0.90	10826

Classification result visualization:





Thorough model validation:

```
The accuracy after cross validation is: 0.8924324218384395
```

4. Comparison and analysis

4.1. Models Comparison

4.1.1. Performance:

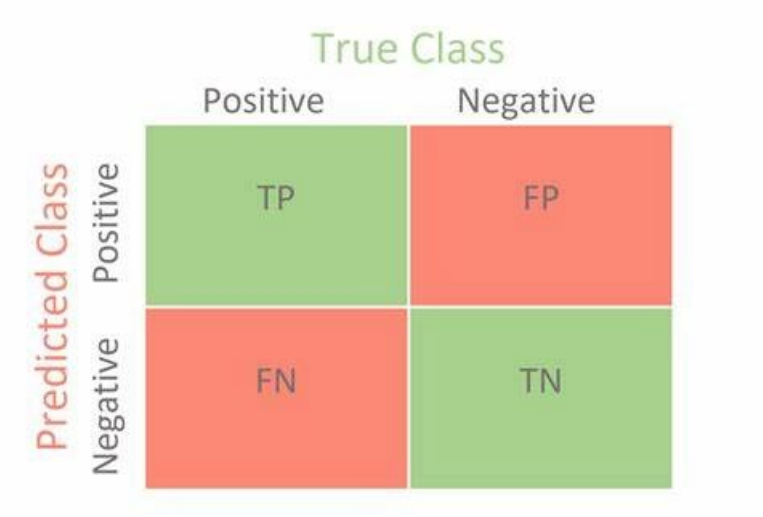
Confusion Matrix

True Positives (TP): which represents the number of samples whose true value is positive and whose classifier determines to be positive (predicted).

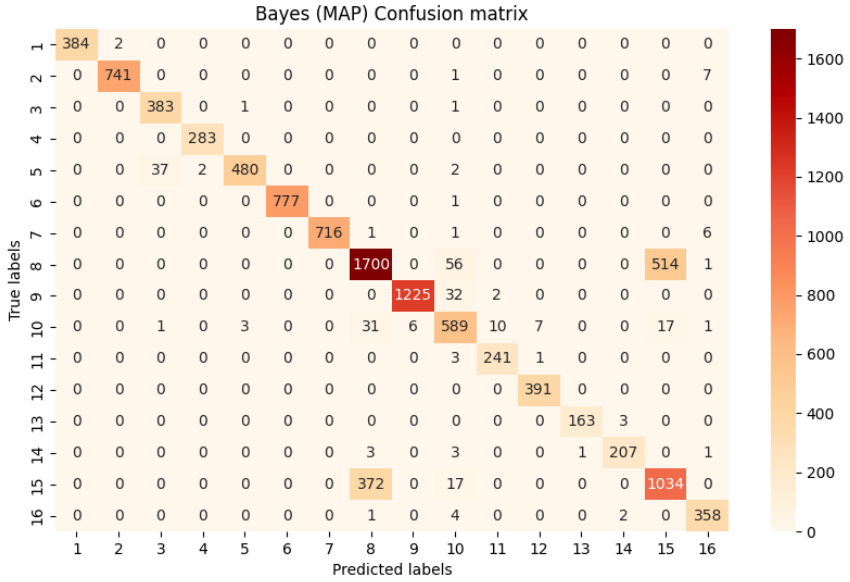
False Positives (FP): which represents the number of samples whose true value is negative and whose classifier determines it to be positive (predicted).

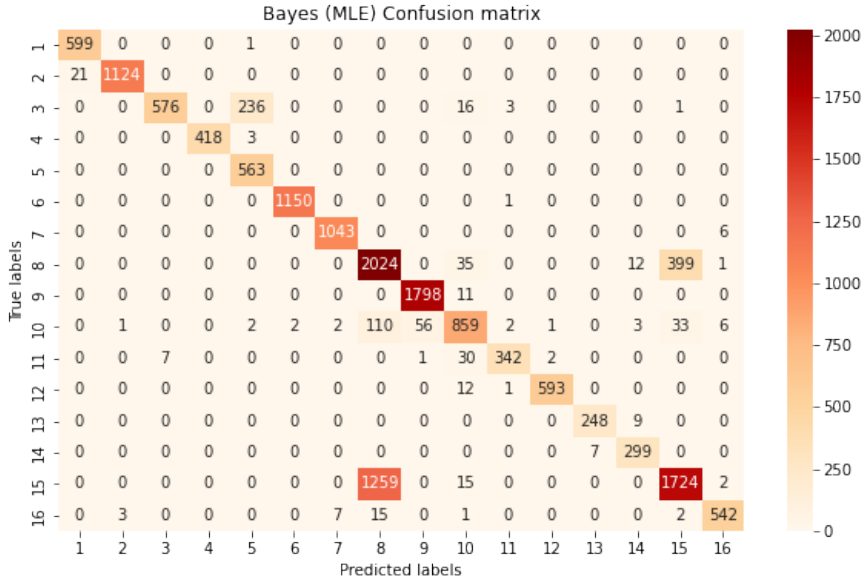
False Negatives (FN): which represents the number of samples whose true value is positive but determined by the classifier to be negative (predicted).

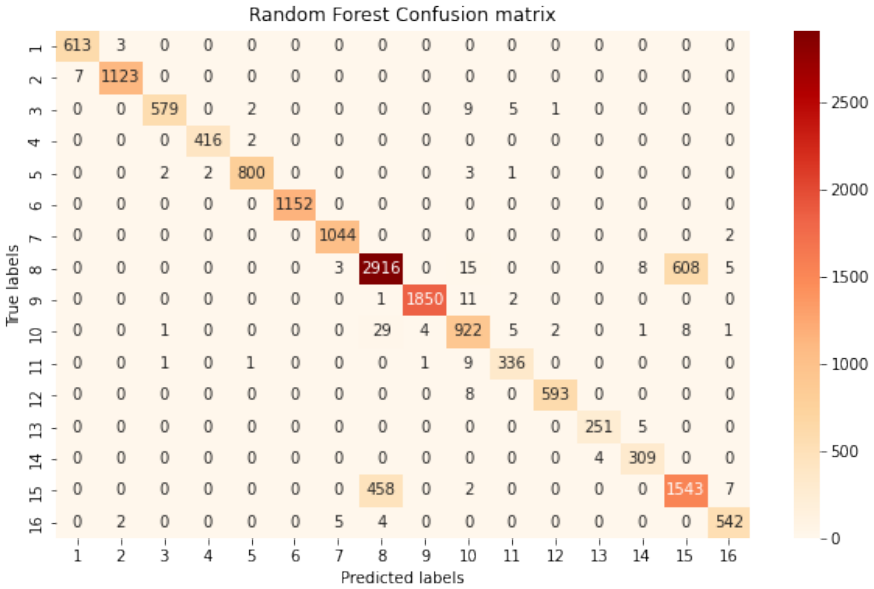
True Negatives (TN): which represents the number of samples whose true value is negative and whose classifier determines it to be negative (predicted).

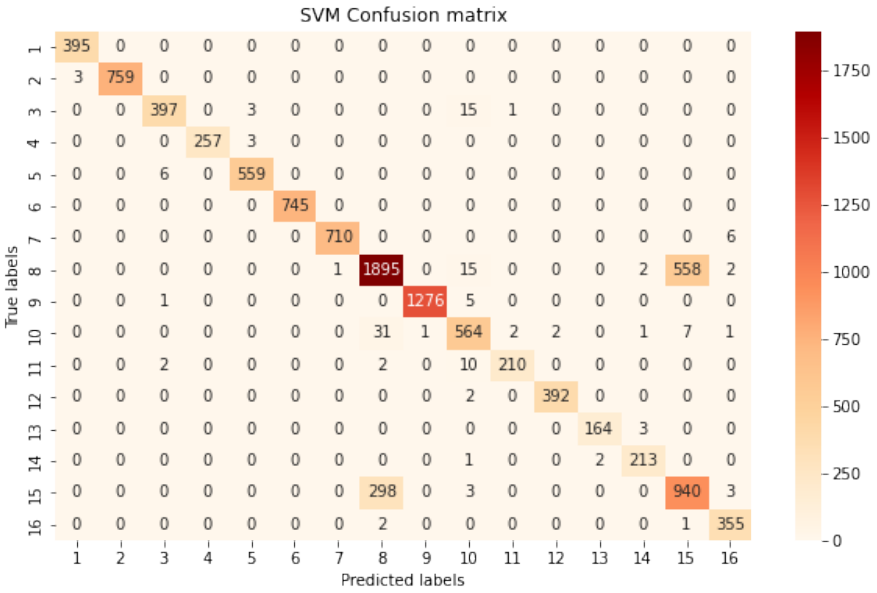


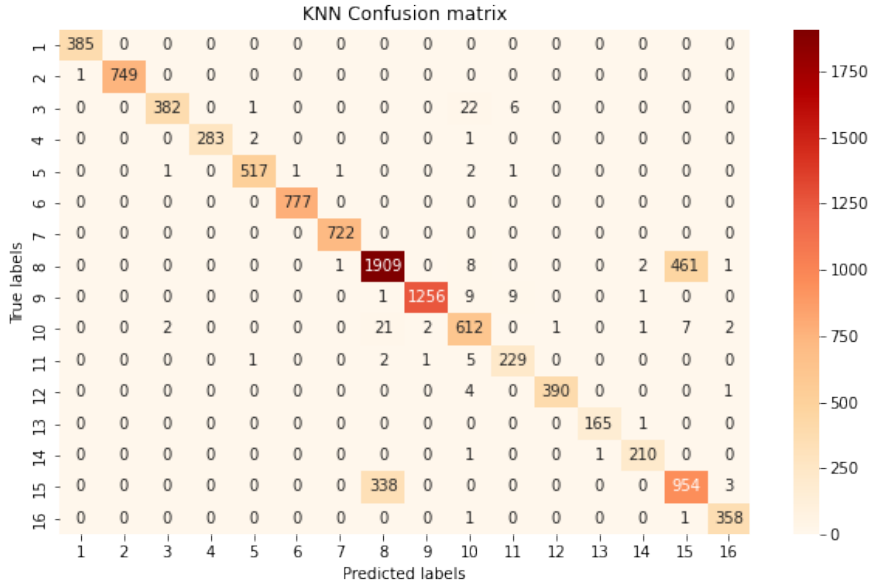
The confusion matrix of the five model is showed as below. It is evident that the 5 and the 15 label are classified with a higher error rate in every models. This may be because of the brands of the spectrum are very similar.




Bayes (MLE) Confusion matrix
True labels
Predicted labels
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
2000
1750
1500
1250
1000
750
500
250
0


Random Forest Confusion matrix
True labels
Predicted labels
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
2500
2000
1500
1000
500
0

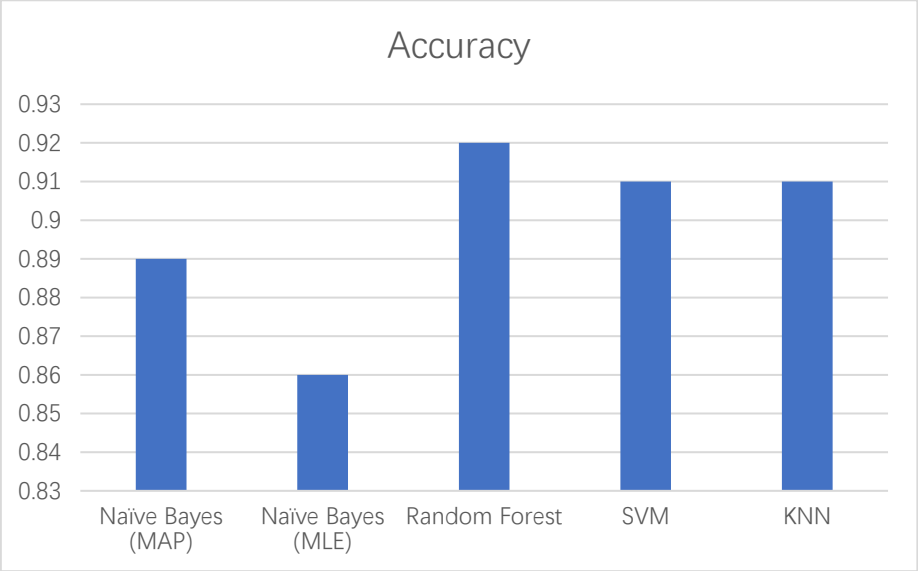




Accuracy:

Accuracy is the percentage of the total that predicts the correct sample size,

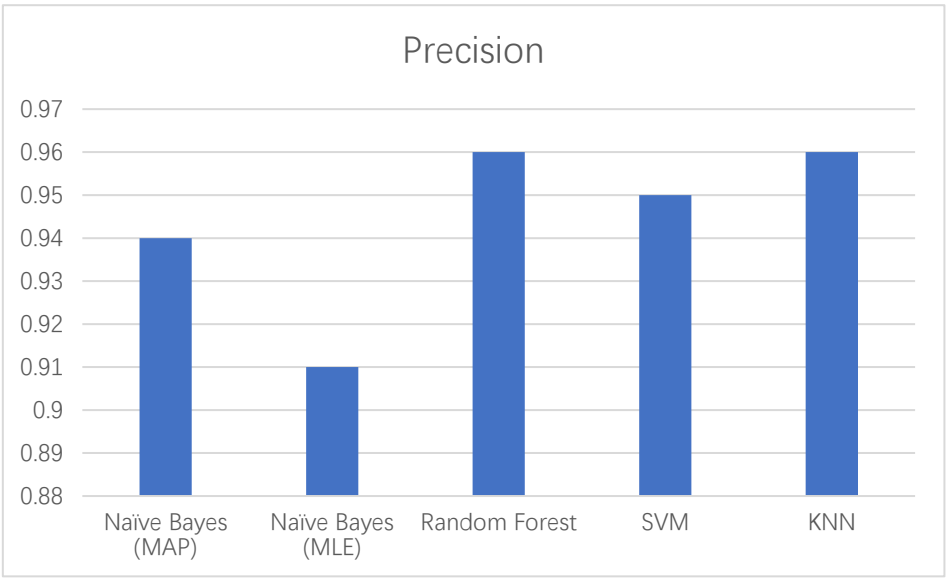
$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP}$$



Precision:

Also known as the accuracy rate, it is an evaluation index for the forecast results. In the results of the model prediction as positive samples, the percentage of the real positive sample is as follows:

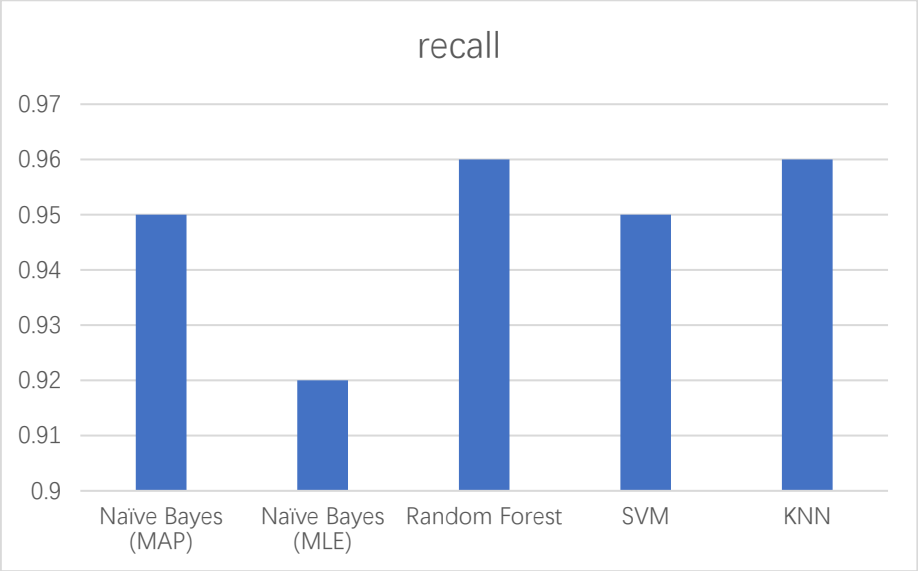
$$Precision = \frac{TP}{TP + FP}$$



Recall:

Also known as the check-up rate, it is an evaluation index for the original sample. In an actual positive sample, the percentage of the sample that is predicted to be positive. The specific formula is as follows

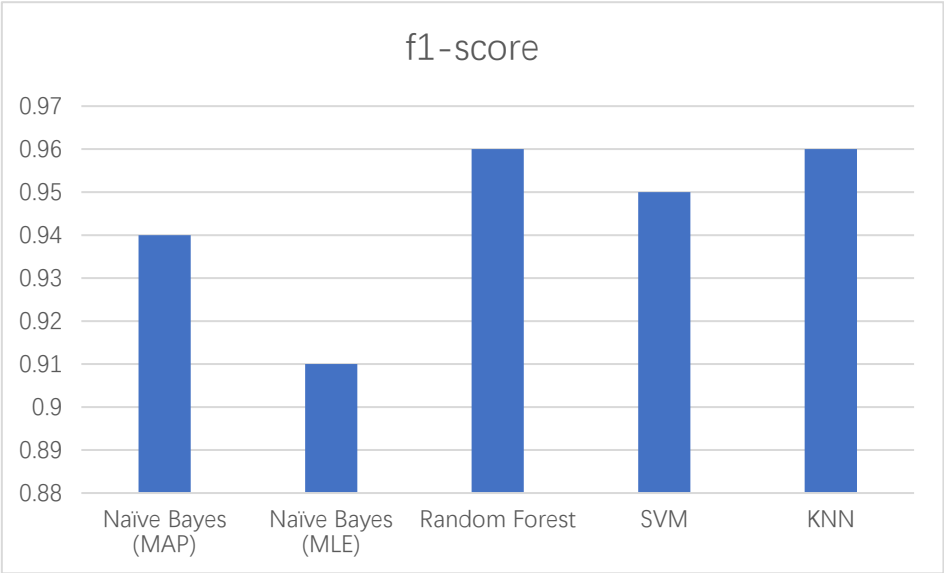
$$Recall = \frac{TP}{TP + FN}$$



F1 Score:

F1 score is a way of combining the precision and recall of the model, and it is defined as the harmonic mean off the model's precision and recall.

$$F_1 = \frac{2}{\frac{1}{recall} \times \frac{1}{precision}} = 2 \times \frac{precision \times recall}{precision + recall} = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$



Time complexity:

The time complexity of Bayes Model is very low, as the theory of Bayes is very simple. While the Random Forest also has a low time complexity, because it can be parallelized. And the time complexity of SVM is the most complexed because of the kernel functions.

Models	Time Complexity
Naïve Bayes (MAP)	O(n*d)
Naïve Bayes (MLE)	O(n*d)
Random Forest	O(n*d)
SVM	O(n^2)
KNN	O(K*n*d)

N is the number of training samples, d is the dimension of data, and K represents the number of neighbours.

4.1.2. Reasons

Reasons for Bayes Classifier (MAP)

a. The time complexity of the Bayesian model is very low, and because of its simple algorithmic principles, the model runs very fast.
b. Although the algorithm principle of Bayesian classifier is very simple, its classification efficiency is very stable, especially when the quantity of samples is small.
c. The theory of Bayes Classifier is based from the Mathematical principle of the Bayes Theory, the classifier is easy to understand and use. For example, in the case of text classification and the filter of junk email.

Reasons for Bayes Classifier (MLE)

a. The plain Bayesian model originates from classical mathematical theory and has a solid mathematical foundation as well as a stable classification efficiency.
b. It has a high speed for large numbers of training and queries. Even with very large training sets, there are usually only a relatively small number of features per item, and training and classification of items is simply a mathematical operation of feature probabilities.
c. Performs well on small data sizes, can handle multiple classification tasks, and is suitable for incremental training (i.e. it can be trained on additional samples in real time).
d. Less sensitive to missing data, the algorithm is simple and conditional probabilities are easy to evaluate.
e. Interpretation of results is easy to understand

Reasons for Random Forest

In all, there are four reasons in terms of either the model's working principle or practical use for choosing random forest classifier in the context of hyperspectral image (HSI) classification problems:

a. Firstly, according to Xu, Ye and Nie [1], random forest classifier can have fairly satisfactory accuracy in classifying multiclass data of high feature dimensions since each individual trees is trained by a randomly selected subset of features,

which is highly associated with the problem of HSI classification where a hyperspectral image usually has large number of bands (i.e., high-dimensional dataset).

b. Secondly, random forest classifier does not suffer from the overfitting problem. As the number of trees increases, the performance of random forest classifier does not decrease since this classifier utilizes the bagging technique and each individual tree is trained by randomly selected samples and the final prediction will be the one that is most voted among all the trees, which greatly enhance the model's generalization ability.
c. Thirdly, the random forest classifier can deal with unbalanced dataset which can be usually seen in the context of a HSI classification problem. Specifically, since the split of nodes in a decision tree is principally based on the weighted sum of Gini impurity, random forest classifier can adjust the weight of unbalanced classes according to the input data frequencies so as to enhance prediction performance [2].
d. Lastly, in terms of the practical use, a random forest classifier can be run parallelly since the decision process of each individual tree is independent. Therefore, random forest can achieve high computational efficiency on modern parallel CPU or GPU to obtain the classification result.

Reasons for SVM

a. Firstly, the optimization of SVM is a convex. This indicates that it is likely to find the global minimum (or closed to it).
b. Secondly, leaving the margin bounding the training examples can reduce the generalization error. Video classification can be used for practical use. Support vector machine (SVM) is used to build video classifier, and Corel video database is simulated and tested. The results show that ga-SVM improves the accuracy of video classification compared with other video classification algorithms [3].
c. Thirdly, nonlinear mapping is the theoretical basis of SVM, and kernel function can be used to map to high-dimensional space.
d. Lastly, the algorithm is simple and has good "robustness". The addition and deletion of SVM non-support vector samples have no effect on the model. In addition, the vector sample set has certain robustness. Under certain conditions, SVM method is insensitive to kernel selection

Reasons for KNN

a. Simple and easy to use, compared with other algorithms, KNN has simple mathematical principle.
b. No input data is assumed. Compared with naive Bayes, for input data, it is assumed in advance that data obeys gaussian distribution or other distributions, KNN does not need to make such assumption.
c. High precision, good prediction effect. KNN algorithm has high accuracy for sample estimation and low classification error rate. In terms of practical use, analysis and classification of Acute Lymphoblastic Leukemia using KNN Algorithm. KNN

classifier was used to label each lymphocyte sub-image as a normal or malignant sample based on a set of measured characteristics [1].

d. Not sensitive to outliers. Since the points in the K neighborhood are judged by the majority voting method, individual outliers will not affect the majority judgment.

For example， it can be applied to a hyperspectral classification method based on joint sparse representation [2]. Determine the center pixel category through the pixels in the local area. KNN has better classification performance

4.2. Analysis

4.2.1. Assumption and Explanation

Hypothesis

The hypothesis about the reason why SVM perform better than bayes methods with higher accuracy is illustrated below:

1. SVM have higher tolerance of image noises which may result in the difficulties separating class 8 and class 15.
2. If assumption one above can be rejected, then it is possible that Bayes methods are less capable of dealing with the overlapping sample data belonging to class 8 and class 15.

Explanation of the hypothesis

Based on the heat map visualizing the confusion matrix. The first phenomenon is misclassification of class 8 and class 15, which is faced by all methods implemented. The second phenomenon is bayes methods perform the worst. It significantly contributes to the gap between overall classification accuracy of Bayes methods and SVM's. For this reason, the hypothesis focuses on underlying factors behind this phenomenon.

4.2.2. Proof

Proof of the hypothesis

There are two aspects that should be justified. Firstly, image noises have negative impacts on separating classes. Secondly, SVM is less sensitive to image noises.

The first aspect: Based on two datasets (Corel and Caltech101-600), Contato et al. [4] found that noises in images can lead to more difficulties of separating classes, which further results in the decreased performance of the model. Therefore, image noises are likely to generate difficulties of separating class 8 and class 15.

The second aspect: SVM performs a lot better when noise level is below 40%, but the

performance gap between it and bayes models become smaller as the noise level increases [5]. When the noise level is extremely high, the classification accuracy of bayes models can exceed SVM's [5]. For this reason, there are no strong evidence supporting the statement 1 in the hypothesis. This leads to the statement2: bayes methods simply have more difficulties classifying overlapping data.

4.2.3. Limitations

Basically, two limitations can be found in our implementation process of hyperspectral image classification.

Firstly, according to Vidal and Amigo [6], the instrumental noise caused by cameras can be partially removed by smoothing methods such as applying a blur on the hyperspectral image. Therefore, when preprocessing the Salinas dataset, a Gaussian blur can be firstly applied to the whole image. After implementing this on our code, the conclusion can be found that applying a Gaussian blur with sigma = 5 when prepossessing Salinas dataset will highly improve the performance of all five classifiers in our group, which can be regarded as one of the limitations of our work. The detail score for all five classifiers with or without Gaussian blur can be seen in the diagram below:

Accuracy	RF	Bayes (MLE)	Bayes (MAP)	SVM	KNN
Without Blur	0.93	0.85	0.89	091	0.91
With Blur	0.98	0.92	0.94	0.95	0.95

Secondly, other than KNN, all other models are classified based on the prediction of each individual pixel without taking into consideration that the labels of neighboring pixels are usually correlated [7]. Therefore, such pixel-wise HSI image may have difficulty in discriminating classes with high interclass variabilities, which can be enhanced by incorporating the spatial information such as the correlation among spatial related pixels into the implementation of corresponding classifiers such as random forest and SVM [8].

5. Conclusion

To sum up, five common classifiers are used to classify data Salinas, including Nave Bayes (MAP), Nave Bayes (MLE), Random Forest, SVM, and KNN. It's visibly to observe the application of machine learning in hyperspectral images. Through comparison and analysis, each classifier has advantages and disadvantages in different aspects, and there is no significant difference in performance. Group analysis showed that SVM perform better than Bayes methods with higher accuracy. From the limitation mentioned above, it is suggested that future study focus on approaches for removing

noise and improving pixel categorization capabilities.

Reference

[1] B. Xu, Y. Ye and L. Nie, "An improved random forest classifier for image classification," 2012 IEEE International Conference on Information and Automation", pp. 795-800, 2012, doi: 10.1109/ICInfA.2012.6246927.

[2] K. Fawagreh, M. M. Gaber and E. Elyan, "Random forests: from early developments to recent advancements", Systems Science & Control Engineering, vol. 2, no. 1, pp. 602-609, Sep. 2014, doi: 10.1080/21642583.2014.956265.

[3] D. Chunni, "SVM Visual Classification Based on Weighted Feature of Genetic Algorithm," 2015 Sixth International Conference on Intelligent Systems Design and Engineering Applications (ISDEA), 2015, pp. 786-789, doi: 10.1109/ISDEA.2015.198.

[4] da Costa, G. B. P., Contato, W. A., Nazare, T. S., Neto, J. E., & Ponti, M. (2016). An empirical study on the effects of different types of noise in image classification tasks. arXiv preprint arXiv:1609.02781.

[5] Abhinav Atla, Rahul Tada, Victor Sheng, and Naveen Singireddy. 2011. Sensitivity of different machine learning algorithms to noise. J. Comput. Sci. Coll. 26, 5 (May 2011), 96–103.

[6] M. Vidal and J. Amigo, "Pre-processing of hyperspectral images. Essential steps before image analysis", Chemometrics and Intelligent Laboratory Systems, vol. 117, pp. 138-148, Aug. 2012, doi: 10.1016/j.chemolab.2012.05.009.

[7] B. Patrick, K. Sina and W. Martin, "Unsupervised Feature Selection Based on Ultrametricity and Sparse Training Data: A Case Study for the Classification of High-Dimensional Hyperspectral Data", Remote Sensing, vol. 10, no. 10, Sep. 2018, doi: 10. 1564. 10.3390/rs10101564.

[8] R. Gogineni and A. Chaturvedi, "Hyperspectral Image Classification", Processing and Analysis of Hyperspectral Data, Jul. 2019, doi: 10.5772/intechopen.88925.

[9] "HSI-Classification" GitHub.com. https://github.com/drguigui1/HSI-Classification/tree/main/src (accessed Dec. 12, 2021).