DTS201TC Coursework Report

Jiayuan Zhu, Yaqi Yu, Zhiyi Zhao

I. Introduction

Through the literature research in recent years, it is found that hyperspectral image(Salinas dataset) classification is already a active research field. In this report, supervised classification is used to classify hyperspectral images using spectral information. Three methods are discussed in the report: K-Nearest Neighbor Classifier, Support Vector Machines and Random Forest. This report discusses the classification of 17 classes on this dataset, rather than 16, retaining the background class, even though the latter was tested to achieve a higher accuracy rate. Since the latter is not an end-to-end model, it requires human intervention after the dataset is obtained by machines before it can be put into the model, which is not industrially practical.

II. Data

A. Dataset Description

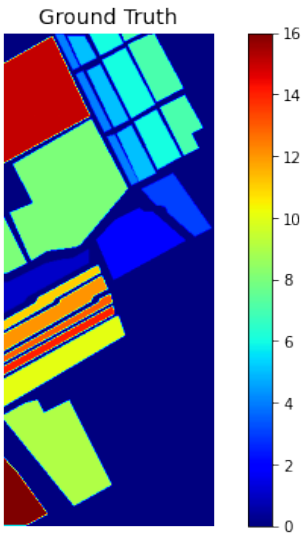


Fig. 1: Ground Truth of The Dataset

This data set comes from *Hyperspectral Remote Sensing Scenes - Grupo de Inteligencia Computacional (GIC) (ehu.eus)*. which is collected by ARVIRIS sensors. This project converts the data set that was originally a dictionary data type into an array that can be classification. The data set is different from the RGB 3-bands of ordinary images, and it has 204-band. The samples covered are 512 lines multiplied by 217. This data set also contains a salinas groundtruth with 17 classes, as shown in Figure 1. All variables are real numbers. Table I shows the percentage of samples in each class. Figure 2 uses a pie chart to show the proportion of samples more intuitively. It can be seen from the chart that the 0-label background class accounts for 51% of the data set, so the classification task must consider whether the background class needs to be eliminated.

TABLE I: Random Search Result

Label	Class	samples	percentage(%)
0	Background	56975	51.28
1	Brocoli_green_weeds_1	2009	1.81
2	Brocoli_green_weeds_2	3726	3.35
3	Fallow	1976	1.78
4	Fallow_rough_plow	1394	1.25
5	Fallow_smooth	2678	2.41
6	Stubble	3959	3.56
7	Celery	3579	3.22
8	Grapes_untrained	11271	10.14
9	Soil_vinyard_develop	6203	5.58
10	Corn_senesced_green_weeds	3278	2.95
11	Lettuce_romaine_4wk	1068	0.96
12	Lettuce_romaine_5wk	1927	1.73
13	Lettuce_romaine_6wk	916	0.82
14	Lettuce_romaine_7wk	1070	0.96
15	Vinyard_untrained	7268	6.54
16	Vinyard_vertical_trellis	1807	1.63

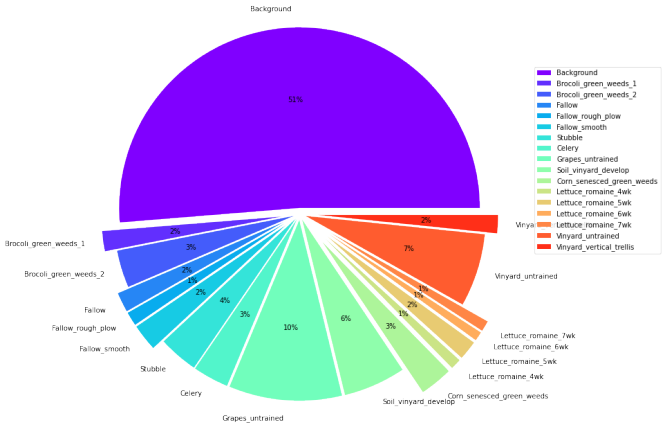


Fig. 2: The percentage of data samples in each class

B. Feature Visualization

Some of the bands in the dataset is displayed in Figure 3. Figure 4 shows data distribution of features in some pixels in the dataset.

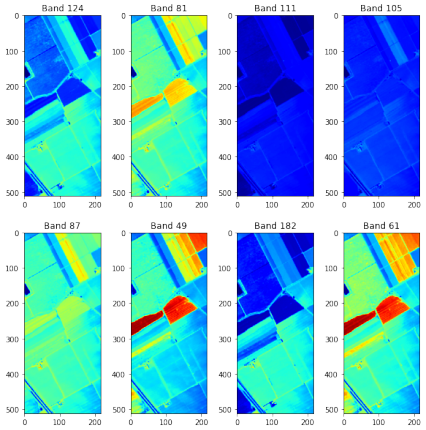


Fig. 3: Sample bands in the dataset

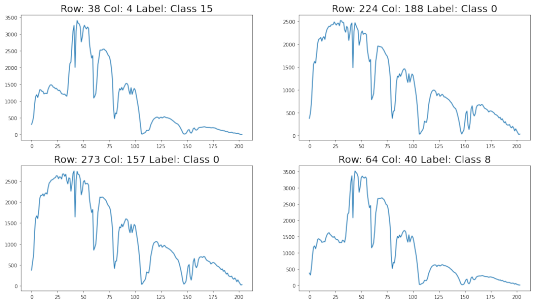


Fig. 4: Data distribution of features in sample pixels

C. Feature Spaces

Use the kernel density estimation method to compare the distribution of feature spaces in different classes. Figure 5 shows a histogram with kernel density estimates for the sample data.

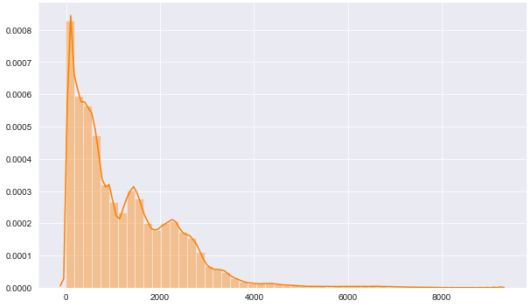


Fig. 5: KDE Histogram

Analysis shows that the data is unbalanced. Figure 6 selects the distribution of the 56-band and the 100-band in the dataset with a label of 10, and visualizes it as a scatter diagram.

Fig. 6: KDE Scatter

Features were reduced to a 2D plane using t-SNE, to show the distribution of data across the different classes, as shown Figure 7. It creates a reduced feature space where similar samples are modelled by nearby points and dissimilar samples are modelled by distant points with high probability.

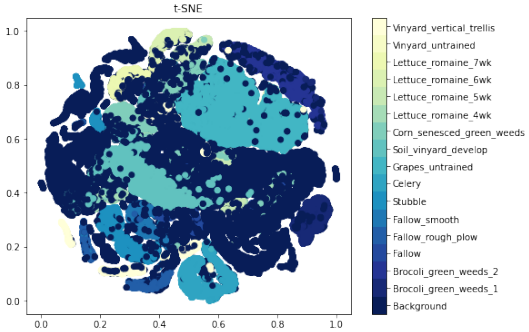


Fig. 7: Sample bands in Salinas.

D. Pre-processing

1) Standardization: To standardize the scale of features, features were transformed to a distribution with mean 0 and standard deviation 1.

$$Standardization : \frac{X_i - \mu}{\sigma}$$

2) Normalization: To generalize and unify the statistical distributivity of features, they were transformed into $[0, 1]$.

$$Normalization : \frac{X_i - X_{min}}{X_{max} - X_{min}}$$

III. Implementation

A. Feature reduction/selection

1) Principal Component Analysis: PCA is an unsupervised dimensionality reduction method, which can not only remove a large amount of redundant information from hyperspectral data, but also retain the spectral information of principal components with large variance contribution. The basis of PCA is the substantial correlation between Salinas bands. Which eliminates the correlation between the bands and determines the best linear combination of the original bands. Treat each pixel value of the dataset as a row vector: $\mathbf{X}_i = [x_1, x_2 ... x_n]$, n is the number of bands number(204).

$$m = \frac{1}{w} \sum_{i=1}^{w} [x_1, x_2 ... x_n]^T$$

m is the mean of all vectors and w is pixel vectors number. The covariance matrix of bands is

$$C = \frac{1}{w} \sum_{w}^{i=1} (\mathbf{X}_i - m)(\mathbf{X}_i - m)^T$$

$$C = ADA^T$$

Select the first K row of A^T matrix and multiply with $\mathbf{X}_i$ to get the most important feature band of the original data set.

2) Independent Component Analysis: The purpose of ICA is different from that of PCA. ICA makes the data as independent as possible, which is more suitable for the high-dimensional nature of this dataset. A mixture of random variables $x_1, x_2, ...x_N$ as a linear combination of random variables $p_1, p_2, ...p_N$. Denote $\mathbf{X} = [x_1, x_2, ...x_n]$, $\mathbf{P} = \left[p_1, p_2, ...p_n\right]$. To find the independent components, the unmixing matrix A^{-1}. Denote $\mathbf{X} \in \mathbb{R}^{d \times N}$ is dataset variables. Independent components are obtained:

$$ICA(\mathbb{X}) = \mathbb{P}_{n \times N} = A^{-1}_{n \times d} X_{d \times N}$$

N: the number of pixels in each band; d: the number of bands; n: the number of materials in the dataset.

B. Evaluation for Pre-processing & Feature reduction

This report provides two preprocessing methods, standardization and normalization, and two dimensionality reduction methods, PCA and ICA. However, not all classifier models use these preprocessing methods. Due to time constraints, it was not possible to adjust the parameters of the model and the corresponding methods and parameters of the preprocess individually. Simple tests of different models and different preprocessing methods were implemented on the training set before further tuning of the parameters, with all parameters of models were set to default. 5-fold cross validation in training dataset was used in the tests.

TABLE II: Pre-processing Selection

Model	Pre-processing	Train Accuracy	Test Accuracy
KNN	Standardization	0.9184	0.8841
	Normalization	0.9193	0.8867
SVM	**Standardization**	0.8776	0.8767
	Normalization	0.8447	0.8444
RF	**Standardization**	0.9957	0.9298
	Normalization	0.9957	0.9297

TABLE III: Feature Reduction

Model	Feature Reuction	n_components	Test Accuracy
KNN	PCA	128	0.7362
		64	0.8860
		32	0.8861
	ICA	128	0.8866
		64	0.8107
		32	0.8641
	None	-	0.8867
SVM	PCA	128	0.8765
		64	0.8678
		32	0.8674
	ICA	128	0.8997
		64	0.9074
		32	0.9070
	None	-	0.8767
RF	PCA	128	0.8881
		64	0.9037
		32	0.9116
	ICA	128	0.8611
		64	0.8940
		32	0.9060
	None	-	0.9298

Based on the results of these tests on the cross validation data, as shown in Table II, Table III, for each model, the corresponding pre-process and feature reduction method, as shown in Table IV. The parameters of the mods are further fine-tuned as explained later. As shown in Table III, feature reduction didn't perform well. This is because the number of samples of **background class** accounts for 51%, and the data dimension containing this class is relatively small, which does not cause curse of dimensionality.

TABLE IV: Pre-processing & Feature reduction Selection

Model	Pre-processing	Feature reduction
KNN	Normalization	-
SVM	Standardization	-
		ICA (32, 64, 128)
RF	Standardization	-

C. K-NearestNeighbor (KNN)

1) Model description: KNN is one of the simplest classifiers, which can be applied to the classification of Salinas image. KNN presumes that all the neighbors make equal contributions to the classification of the testing point, and KNN used *Euclidian* distance metric, which assume the data is homogeneous. The distance formula is as follows:

$$D(x, y) = \sqrt{\sum_{k=1}^{n}(x_k - y_k)^2}$$

Let N traning data spliting from Salinas dataset is $\mathbf{X} = [X_1, ..., X_N]$, the k nearest neighbors of X_i, $\mathbf{X_i} = [X_{i1}, ..., X_{ik}]$. And testing data spliting from Salinas dataset is denoted as X_t with x_0 (random testing point), the k nearest neighbors of testing data with labels $[y_1, y_2, ..., y_k]$. Assume classification classes in the data set is $|L| = [L_1, ..., L_C]$, C is the number of classes. The KNN model finds the k nearest neighbors of a testing point x_0 in the train data and assigns the testing point to the most frequently occurring class of its K neighbors. The classification of x_0 by majority voting rule is:

$$e^* = \arg \max_{e=1,...,C} \sum_{i=1}^{k} \delta(y_i, e)$$

δ is the Kronecker function.

2) Parameters:

- ***n_neighbors***: Number of neighbors. A small value of k means that prediction with a smaller training data set in the domain will have a small training error approximation error and an increased generalisation error. In other words, a smaller value of k means that the overall model becomes complex and prone to overfitting; conversely, a larger value of k means that the overall model becomes simple and prone to underfitting.
- ***weights***: Weight function used in prediction, with $uniform$ or $distance$. $Uniform$ means that equal weights are given to different points, $distance$ means that different points

are given different weights due to different distances from the neighbour.

- ***algorithm***: Algorithm used to compute the nearest neighbors, with $ball\_tree$, kd_tree or $brute$. $Brute$ is to calculate the distance between the input instance and each training instance and select the first k nearest neighbours for majority voting, this is a simple implementation but does not work well when the training set or feature dimension is large. The kd_tree and $ball\_tree$ are structures that stores instance points in a k-dimensional space for fast retrieval, eliminating the need to search for most of the data and greatly reducing computational effort.

According to the previous tests, the model over-fitted the training set when using the default parameter. Therefore, the complexity of the model needs to be reduced and the generalization capability needs to be increased. The chosen alternative values for grid search of $n_neighbors$ are $[0.1, 1, 2, 4, 6, 8]$, $weights$ are $['uniform', 'distance']$, and $algorithm$ are $['kd_tree', 'ball_tree']$. Finally, according to gird search, the model performs best when $n_neighbors$ is 8, $weights$ is $'distance'$ and $algorithm$ is $'kd_tree'$.

3) Evaluation:

TABLE V: Accuracy of KNN

Accuracy	Fold 0	Fold 1	Fold 2	Fold 3	Fold 4
Test Set	0.9062	0.9062	0.9061	**0.9068**	0.8999
Validation Set	0.9072	0.9072	0.9008	0.9027	0.9042

a) Output the accuracy of different folds: As shown in Table V, fold 3 got the best performance on the test set, with accuracy 0.9068.

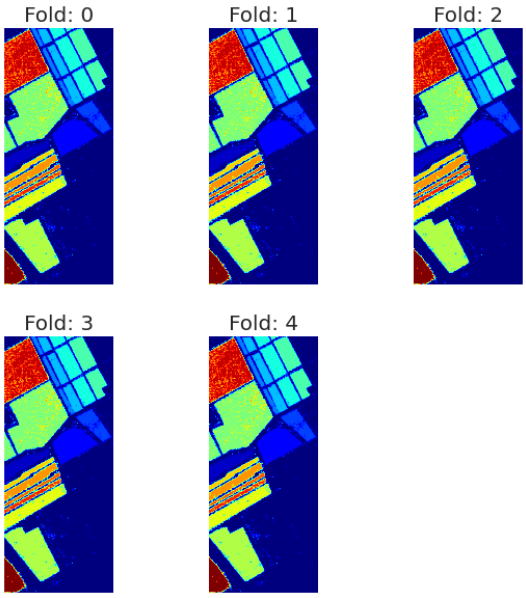


Fig. 8: Result visualization of KNN

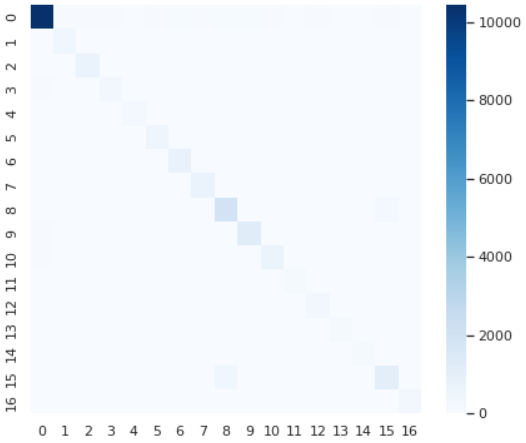


Fig. 9: Confusion Matrix of the fold with best performance of KNN

b) Classification Result Visualization: Figure 8 shows the classification result visualization of different folds of KNN. And Figure 9 shows the confusion matrix of the fold with best performance.

D. Support Vector Machines (SVM)

1) Model description: SVM is a supervised learning algorithm for binary classification. The principle is to find a hyperplane that separates two classes of dissimilar samples as far apart as possible, hence the max margin. It can also use different kernels, making it possible to partition non-linear data. Assume the training set $T = (x_0, y_0), (x_1, y_1), ..., (x_N, y_N), x_i \in \mathbb{R}^n, y_i \in$

$+1, -1, i = 1, 2, ..N$. Construct and solve the convex quadratic programming problem with the appropriate kernel function $K(x, z)$ and penalty parameter $C > 0$.

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, x_j) - \sum_{i=1}^{N} \alpha_i$$

$$s.t. \sum_{i=1}^{N} \alpha_i y_i = 0, 0 \leq \alpha_i \leq C, i = 1, 2, ..., N$$

Choose a component α_j^* of α^* that satisfies $0 < \alpha_i < C$, calculate

$$b^* = y_j - \sum_{i=1}^{N} \alpha_i^* y_i K(x_i, x_j)$$

Then, get the decision function

$$f(x) = sign\left(\sum_{i=1}^{N} \alpha_i^* y_i K(x, x_i) + b^* \right)$$

2) Parameters: The following hyperparameters are often selected for tuning for SVM.

- ***C***: Regularization parameter. The higher the C value, the higher the penalty for misclassification, the weaker the generalization ability of the model and the more likely it is to be overfitted; the lower the C value, the lower the penalty for misclassification, the stronger the generalization ability of the model and the more likely it is to be underfitted.
- ***gamma***: Kernel coefficient. It determines the distribution of the data after it is mapped to the new feature space. The larger the gamma, the less support vectors and the more prone to overfitting, and the smaller the gamma value, the more support vectors and the more prone to underfitting.
- ***kernel***: The kernel type to be used in the algorithm. The main ones are $'linear'$, $'poly'$, $'rbf'$, $'sigmoid'$. $'linear'$ is only applicable to linearly divisible data, $'sigmoid'$ is mainly used in neural networks. Both $'poly'$ and $'rbf'$ can map data from a low-dimensional space to a high-dimensional space, and the difference lies in complexity and time spent.

According to the previous tests, the model underfitted the training set when using the default parameter. Therefore, the complexity of the model needs to be increased and the generalization capability needs to be increased. The chosen alternative values of C are $[0.1, 1, 2, 4, 6]$, $gamma$ are $[0.001, 0.01, 0.1, 1, 2, 4, 6]$, and $kernel$ are $['rbf', 'poly']$. Finally, according to gird search, the model performs best when C is 2, $gamma$ is $'0.1'$ and $kernel$ is $'poly'$. Besides, ICA with n_components in $[32, 64, 128]$ was also tried in the grid search, however, the performance with ICA was lower than without, so finally abandoned the use of ICA in SVM.

3) Evaluation:

TABLE VI: Accuracy of SVM

Accuracy	Fold 0	Fold 1	Fold 2	Fold 3	Fold 4
Test Set	0.9251	0.9242	**0.9267**	0.9255	0.9263
Validation Set	0.9289	0.9252	0.9237	0.9299	0.9278

a) Output the accuracy of different folds: As shown in Table VI, fold 3 got the best performance on the test set, with accuracy 0.9267.

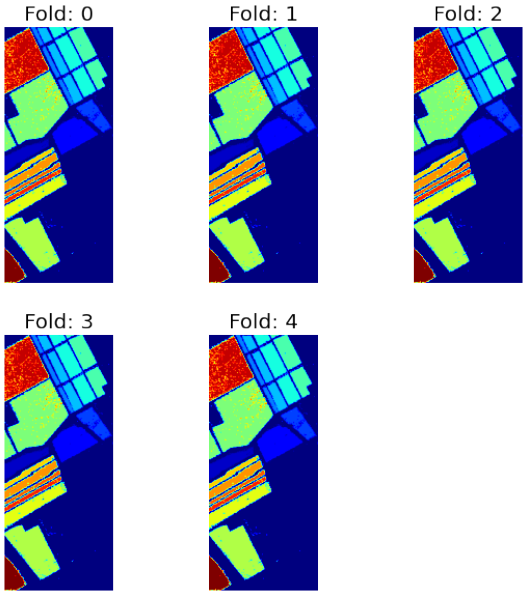


Fig. 10: Result visualization of SVM

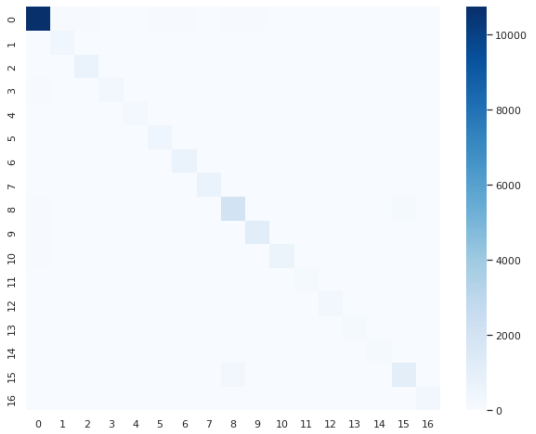


Fig. 11: Confusion Matrix of the fold with best performance of SVM

b) Classification Result Visualization: Figure 10 shows the classification result visualization of different folds of SVM. And Figure 11 shows the confusion matrix of the fold with best performance.

E. Random Forest (RF)

1) Model description: RF is a Bagging Ensemble Model built using multiple CART (Classification and Regression Tree) as base learners. More specifically, a sample set of T containing m training samples is sampled by bootstrap sampling, and then a CART is trained based on each sample set, and these decision trees are then combined. CART is a binary decision tree constructed by using the Gini index to select the division attributes. Assume that the proportion of the kth class of samples in the sample set D is $p_k(k = 1, 2, ..., |y|)$, the Gini index is defied as

$$Gini(D) = \sum_{k=1}^{|y|} \sum_{k' \neq k} p_k p_{k'} = 1 - \sum_{k=1}^{|y|} p_k^2$$

Assume that the discrete property a has V possible values $a^1, a^2, ..., a^V$ corpses. If a is used to partition the sample set D, it will result in V branch nodes, where the vth branch node contains all samples in D that take value a^v on attribute a, denoted D^v. Then, the Gini index of attribute a is defied as

$$Gini\_index(D, a) = \sum_{v=1}^{V} \frac{|D^v|}{|D|} Gini(D^v)$$

Thus, in the set of candidate attributes A, the attribute that minimizes the Gini index after division is selected as the optimal divided attribute,

$$a_* = \arg\min_{a \in A} Gini\_index(D, a)$$

2) Parameters: The following hyperparameters are often selected for tuning for RF.

- ***n_estimators***: The number of trees in the forest, namely the base learners. This value is too small and tends to be under-fitted, too large and tends to be over-fitted.
- ***max_depth***: The maximum depth of the tree. The default is None, which is the maximum depth. The smaller the value, the lower the complexity of the model and the easier it is to under-fit.

Same as KNN, the model with default values over-fitted the training data. Therefore, alternative values for $n\_estimators$ should be concentrated at less than 100 and max_depth should be assigned. The chosen alternative values of $n\_estimators$ are $[50, 60, 70, 80, 90, 100, 120, 130, 140]$, and max_depth are $[40, 60, 80, 100, None]$. Finally, according to gird search, the model performs best when $n\_estimators$ is 130 and max_depth is $None$.

3) Evaluation:

TABLE VII: Accuracy of RF

Accuracy	Fold 0	Fold 1	Fold 2	Fold 3	Fold 4
Test Set	0.9280	**0.9288**	0.9282	0.9276	0.9287
Validation Set	0.9326	0.9313	0.9306	0.9300	0.9284

a) Output the accuracy of different folds: As shown in Table VII, fold 2 got the best performance on the test set, with accuracy 0.9288.

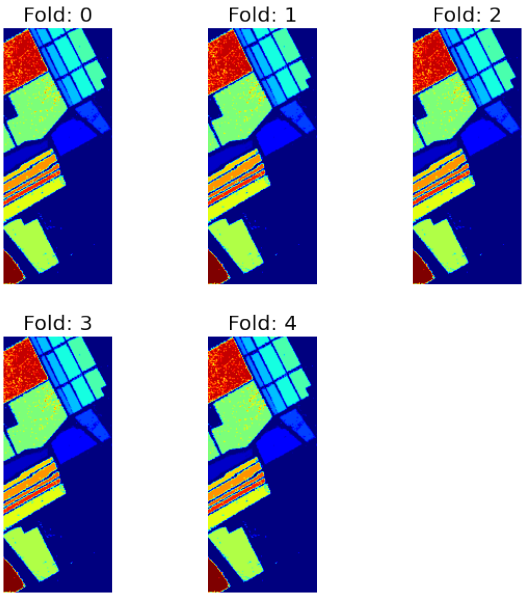


Fig. 12: Result visualization of RF

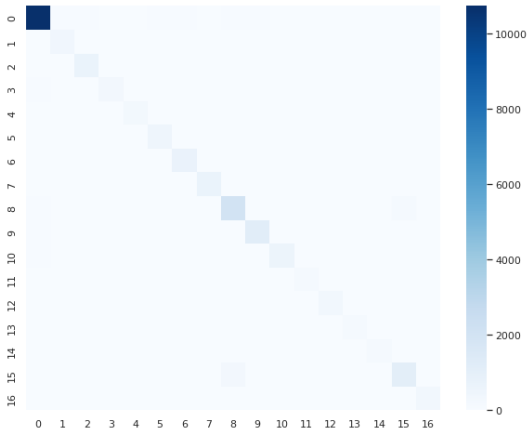


Fig. 13: Confusion Matrix of the fold with best performance of RF

b) Classification Result Visualization: Figure 12 shows the classification result visualization of different folds of Random Forest. And Figure 13 shows the confusion matrix of the fold with best performance.

F. Platform

Table 3 provides the platform infromation for running 3-models.

TABLE VIII: Model Running Platform

Model	CPU	Memory
KNN	M1 8-Core	16GB
SVM	M1 8-Core	16GB
RFz	M1 8-Core	16GB

IV. Discussion

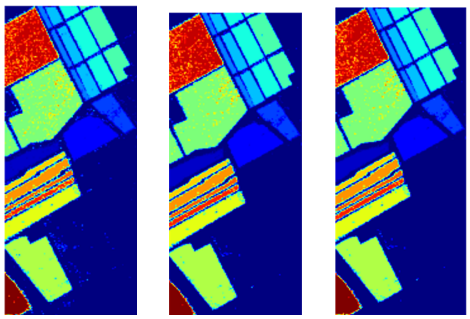
A. Model Comparison

As shown in Table IX, the accuracy comparison of the three classification models.

TABLE IX: Model Accuracy Comparison

Model	Best Accuracy
KNN	0.9068
SVM	0.9267
RF	0.9288

The classification visualization results of the three models are shown in the figure below.



(a) KNN Classification (b) SVM Classification (c) RF Classification

Fig. 14: Classification Result Comparison

This data set has labels so it can be supervised learning, and is essentially a hyperspectral image classification problem. Therefore, the use of SVM, KNN and random forest classifiers will meet the essence of this problem. And through the analysis of literature and experimental results, it is shown that in experiments on real hyperspectral data sets, the classification results obtained by the proposed method are good.[2][3][4]

B. Analysis

Through the comparison and visualization of the above classification results, it can be seen that the accuracy of the results of SVM and random forest is better than the results of KNN classification. However, due to the high computational complexity of SVM, from the perspective of algorithm time, random forest with little difference in accuracy is

a better choice. And comparing the data in this report with the results proves this assumption. This assumption can also be verified by experiments in "Overview of Hyperspectral Image Classification"[1]. The major limitations of the present study is the lack of existing research on the hyperspectral topic. Admittedly, the main limitation is sample size is small, and our team has limitations in the running platform leading to long time experiment.

C. Novel Ideas

As mentioned above, there is significant imbalance in the dataset, with **background class** taking up about half of it, which has heavily influenced the effectiveness of the training. Therefore, we developed two approaches to minimize the impact of it.

a) Weighted Classifier: Take KNN as an example, a weighted-KNN model that is more suitable for the classification of hyperspectral images, which modifies to assign different weights to the neighbors. The KNN classifier finds the K nearest neighbors of the test points in the training data, and assigns the test points for which the sum of weights chosen for the neighbors is largest.

$$e^* = \arg \max_{e=1,\ldots,C} \sum_{i=1}^{k} W_i \delta(y_i, e)$$

This changes the above decision formula, W_i is the weight of test data x_{0i}

b) Downsampling or Oversampling:

- Downsampling: Downsampling balances dataset by cleaning the original dataset. For example, edited dataset using nearest neighbours (function *EditedNearestNeighbours* in package *imblearn*) identifies and remove those data that are not fit well with neighbours.
- Oversampling: Oversampling balances dataset by generating new data based ont he distribution of the original dataset or just repeat randomly selects minority samples. For example, Synthetic Minority Oversampling Technique (SMOTE) choose a random nearest-neighbour sample b for a minority sample a, then select a random point c from the line between a and b as a new minority sample.

V. Conclusion

The Salinas data set to be classified in this paper has the characteristics of high data correlation and large data redundancy, especially the background class with a label of 0 occupies more than 50% of the data set. In view of these characteristics of the data, this report proposes two preprocessing methods, normalization and standardization, and two dimensionality reduction methods, PCA and ICA. The processed data is applied to three types of supervised classifier models: KNN, SVM and RF. In addition, the authors comparatively analyzed the classification results of excluding the data set with a label of 0 (and including the 0-label dataset). And horizontally compare the three supervised classifier models. Random forest is the best model to classify this data set. Future research can consider how to combine complementary features such as spectral pixel information while maintaining low computational complexity. Use the weighted classification model to further improve the accuracy and efficiency of spectral image classification, and to deal with more hyperspectral image data sets classification.

References

[1] Wenjing, Lv., Xiaofei, Wang. "Overview of Hyperspectral Image Classification" in Journal of Sensors, July 2020.

[2] Huang, K., Li, S., Kang, X. et al. Spectral–Spatial Hyperspectral Image Classification Based on KNN. Sens Imaging 17, 1 (2016). https://doi.org/10.1007/s11220-015-0126-z

[3] G. Mercier and M. Lennon, "Support vector machines for hyperspectral image classification with spectral-based kernels," IGARSS 2003.

2003 IEEE International Geoscience and Remote Sensing Symposium. Proceedings (IEEE Cat. No.03CH37477), 2003, pp. 288-290 vol.1, doi: 10.1109/IGARSS.2003.1293752.

[4] S. R. Joelsson, J. A. Benediktsson and J. R. Sveinsson, ”Random forest classifiers for hyperspectral data,” Proceedings. 2005 IEEE International Geoscience and Remote Sensing Symposium, 2005. IGARSS ’05., 2005, pp. 4 pp.-, doi: 10.1109/IGARSS.2005.1526129.