5318 Assignment1 Report

1 Introduction

1.1 Overview of the data set

The data set used in the assignment is derived from the Fashion-MNIST dataset, a collection of fashion articles represented as grayscale images. The data set includes a train set which has 30000 examples and a test set which has 5000 examples. Whatever the train set and the test test, both of them have 784 features and 10 classes. To make the validation convenient, 10% of the train set is extracted as a validation set for evaluation. The purpose of this assignment is to create a machine learning classifier, using the train set to train the model and make it have the ability of sorting grayscale images into predefined categories.

1.2 Methods chosen

In this assignment, we choose various data preprocessing methods and machine learning algorithms to help build the classifier. The data preprocessing methods that are used include normalisation, missing value imputation and dimensions reduction by PCA. Moreover, the machine learning algorithms that are used include k-nearest neighbour, logistic regression, support vector machines and random forests. After using these methods to preprocess the data and train the model, we use various classification performance metrics like precision, recall, F1 score and Confusion matrix to comprehensively evaluate the model and choose the best one.

1.3 The key findings and results

By testing on the test1 data set (which only has around the first 2000 examples), we find that the model based on support vector machines has the best performance with 0.911 accuracy, 0.91 precision, 0.91 recall and 0.91 F1-score. In addition, we find that all the four models performed significantly worse in predicting class 6 thus affecting the overall performance of the model and the space for improving the total accuracy. From this phenomenon, it is clear to see that models may need additional specific methods for extracting features of the data set in order to improve their overall performance. In summary, we build the classifier based on SVM and will try to use the CNN for further refining feature extraction and enhance the accuracy for predicting the class 6 to improve the performance of the model in the future.

2 Methodology

2.1 pre-processing techniques

2.1.1 Normalisation

In the data preprocessing phase, we firstly use the normalisation method to process the data so that we can . The Min-Max scaling is “used to transform numerical features into a specific range, typically between 0 and 1” (Singh, 2023). The necessity of this method will be explained in more details from the following two aspects:

(1) Theoretical Principle: Some machine learning algorithms like KNN and SVM are sensitive to the scale of the data because both of them rely on the calculating methods which are based on distances (Singh, 2023). If we don’t normalise the data in advance, there may be a possibility that the unnormalised data will affect the learning process thus finally bringing negative impacts on the accuracy of the model.

(2) Rationale: By using the normalisation, we can prevent some specific feature from seriously affecting the model and accelerate gradient descent for optimal solutions. Additionally, normalisation can improve the accuracy of the models based on KNN or SVM and make the training process smoother.

2.1.2 Missing value imputation

In some extreme circumstances, there may be missing values in the data set after normalisation because when using the Min-Max scaling to scale the data, if the original data set has the extreme outlier, this may lead to overscaling thus bringing computational errors and missing values. In this assignment, we choose mean to realise the missing value imputation and next this method will be explained in more details from the following aspects:

(1) Theoretical Principle: According to the original data set, it is clear to see that the data are numerical, and using means to impute missing values is adapted to such this kind of data. Moreover, "it is usually combined with replacing missing values with the most common attribute value for symbolic attributes" (Kaiser, 2014). In general, mean can represent the main trend of the data set which means that using means to impute missing values may not significantly deviate the overall distribution of the data set.

(2) Rationale: The first reason for using this method is its convenience because this method can lead to lower computational costs and is easy to implement. Additionally, this method will not seriously damage the overall structure of the data set thus preventing extreme imputation values from bringing noise. Last, as the data set used in this assignment is small and medium-sized, choosing means to impute the missing values may not cause negative impacts on the performance of the model.

2.1.3 Dimensions reduction by PCA

The data set with high dimensions usually has lots of shortcomings like slower training, overfitting and unreliable classification. Therefore, we choose principal component analysis to reduce the dimensions and retain the important information of the original data set as much as possible.

(1) Theoretical Principle: The main idea of PCA is to project data with high dimensions into the space with low dimensions. The fewer dataset dimensions can keep the features that have the most impact on the original data set through the orthogonal transformation of data projection, thus realising the purpose of retaining the main feature. Maćkiewicz and Ratajczak (1993) reported that PCA created a new feature space through calculating the eigenvalues and eigenvectors of the covariance matrix of the data where each feature represented a linear combination of original features. These features are also called principal components that capture the main variance of the data and provide an effective way for dimension reduction.

(2) Rationale: For this assignment , the original data set has 784 features which means that it is the medium and high dimensional data sets. So in this case, using PCA for dimension reduction can significantly reduce the computational complexity of the model and redundancy feature thus improving the efficiency of the model. Additionally, as the original data set is a collection of grayscale images, which means that image data is highly structured. "PCA is widely used for image fusion, image compression, image segmentation, etc" (Nandi et al., 2015), this means that PCA can

process image data more effectively compared with other methods of dimension reduction.

2.2 ML algorithms

2.2.1 K-Nearest neighbour

The first algorithm we choose is the k-nearest neighbour. It is widely used as a classification method as it is easy to implement and has a good performance on the classification task (Zhang et al., 2017). This algorithm will be explained more details in the following aspects:

(1) Theoretical Principle: The main idea of the k-nearest neighbour algorithm is that if large majorities of the k nearest neighbours of one sample in the feature space belong to the same class, then the sample also belongs to the same class and has the characteristics of the samples in this class. In the classification task, this algorithm will predict the class of the samples according to the class of one or several samples of the nearest neighbour. Moreover, k-nearest neighbour is a flexible algorithm because it is a non-parametric algorithm, which means that it does not need to assume the distribution of the underlying data and can adapt to lots of situations.

(2) Rationale: For this assignment, as it needs to build a classifier, it is appropriate to choose a mature classification algorithm like k-nearest neighbour. Additionally, the training time of KNN-based models is normally short, it is simple and flexible enough to provide an advantage with low time cost and high performance on the classification tasks. Therefore, we use this algorithm for the assignment.

2.2.2 Logistic Regression

Then, the second machine learning algorithm that we choose is logistic regression. Nick and Campbell (2007) reported that the logistic regression model was "statistical models which describe the relationship between a qualitative dependent variable and an independent variable." It is explained in more details as follows:

(1) Theoretical Principle: Logistic regression is widely used in the binary classification tasks by predicting the probability of outcomes or events and it usually uses logistic function to map the output value between 0 and 1 (Kanade, 2022). Logistic function's output provides a probability value which can be thresholded to classify the data into one of two classes. The main idea of logistic regression is to firstly fit the decision boundary, then build the probability relationship between this boundary and classification to acquire the probability in the binary classification case. It is normal to use maximum likelihood estimation and Gradient descent to solve the logistic regression and get its parameters. To prevent the model from overfitting, it is usual to use regularisation to solve this problem especially for the data set with many features. Based on linear regression, logistic regression adds a logistic function mapping, which makes it an excellent classification algorithm.

(2) Rationale: For this assignment to build a classifier, logistic regression has lots of advantages. Firstly, this model has a good interpretability, it is easy to interpret the effects that different features may make on the classification results according to the weight coefficient. Moreover, it has the advantages of simple and high efficiency, it is easy to implement in the classification tasks and just needs short training time even in

the big scale data set. Last but not least, it usually has a good performance and high accuracy in the classification task, so choosing this algorithm is appropriate.

2.2.3 Support vector machines

Support vector machines are good tools for classification tasks and "they exhibit good generalisation performance on many real issues" (Yue et al., 2003). Support vector machines can deal with linear or non-linear data and improve their abilities for classification by using different kernel functions. The algorithm will also be explained in more details from the following aspects:

(1) Theoretical Principle: The main working principle of SVM is to "find the optimal hyperplane that separates data points into different classes" (Tibrewal, 2023). In the circumstance that the data is linearly divisible, SVM can find the decision boundary which maximises the intervals between classes to perform classification. The optimal purpose for this model is to maximise the intervals between classes as much as possible to improve the generalisation ability of the model. Similarly, SVM model also introduces the regularisation parameters to prevent overfitting and make the model more stable in the data set with high dimensions. Additionally, Tabsharani (2023) reported that kernel functions used in the SVM made the model "possible to map the data from the original feature space to the kernel space" and have a good ability to effectively deal with the complex problems in the high dimensions space.

(2) Rationale: For this assignment, the data set still has more than 100 features although after dimension reduction, choosing SVM is appropriate because this model has a good ability to deal with the medium and high dimensions data. Apart from that, as SVM can find the explicit classification boundary, this makes it effective on classifying those images which have subtle colour and shape differences such as the grayscale images. Last but not least, as some classes in the grayscale images have subtle visual differences, it may have a high possibility to cause the misclassification. However, as SVM model introduces some regularisation parameters, this makes it can tolerate some misclassification to some degree and maximise the performance of classification. Therefore, it is necessary to choose this algorithm.

2.2.4 Random forest

Random forest is a kind of ensemble machine learning algorithm. It creates lots of decision trees and summarises their results of prediction to improve the accuracy of the classification. This algorithm will be explained in details as follows:

(1) Theoretical Principle: The random forest uses randomisation to create a lot of decision trees and totals their results for predictions. The creating process includes "the selection of samples subset and feature subset, to guarantee the independence of each decision tree and improve classification accuracy" (Parmar et al., 2019). According to the average outputs of the many decision trees, random forest can greatly reduce the variance of the model and prevent overfitting that may be caused by a single decision tree. Random forest has a good performance on dealing with non-linear relationships and complex interactions between features.

(2) Rationale: To build the classifier for the grayscale images, random forest has the strong abilities of anti-overfitting especially in the high dimensions data. As the features are extracted randomly in the process of creating decision trees, this makes

the model based on random forest have a strong robustness and efficiently prevent some specific features from affecting the performance of the model. In summary, using this algorithm is appropriate.

3 Result and Discussion

3.1 Experimental Settings

In this section, we describe the detailed steps taken to implement each model and fine-tune their hyperparameters to achieve optimal performance.

3.1.1 Implementation Strategies

(1) Data Preprocessing:

Normalisation: Because there are big differences in the ranges of different features (range from 0 to 255) and ML models are sensitive to the scale of input features, it is essential to cast the data to the specific range, like between 0 and 1(Ali et al., 2014). We used MinMaxScaler to transform all the feature values to the range [0, 1].

Dimensionality Reduction: With 784 features per image, the dataset is quite high-dimensional. Principal Component Analysis (PCA) can reduce the risk of overfitting and improve the accuracy of training (Van Der Maaten et al., 2009). This is applied by retaining 95% of variance (n_components=0.95) from the original data and fitting the transformation to the normalised training /test data.

(2) Model Implementation:

K-Nearest Neighbour: We implement K-Nearest Neighbour (KNN) using KNeighborsClassifier. Initially, we set k=1, meaning the model considers only the single closest neighbour for classification. We chose this low value of k to find KNN's behaviour on small datasets. Then, we set Manhattan distance (p=1). During the prediction, this model computes the Manhattan distance(p) between each test sample and training sample, then classifies the test sample based on the label of the closest (k=1) training sample.

Logistic Regression: We implement Logistic Regression using LogisticRegression. Initially, we employ L2 regularisation, which adds a penalty to large weight values and thus prevents the model from fitting the training data too closely. We set c=10 to control the strength of the regularisation, allowing the model more flexibility while remaining generalisation. Then, we set solver='liblinear', which is efficient for small datasets, and max_iter=1000, which sets the maximum number of iterations for the optimization algorithm. During the prediction, this model computes the probability for each class using the logistic function, then assigns the test sample to the class with the highest probability.

Support Vector Machine: We implement Support Vector Machine using SVM. SVM uses machine learning theory to maximise predictive accuracy and the margin between the classes while allowing for some misclassifications (controlled by the C parameter)(Jakkula V, 2006). In this case, we set kernel='rbf', which is more suitable for non-linear classification. During the prediction, this model computes the decision boundary using the rbf kernel, then classifies the test sample based on which side of the boundary it falls, maximising the margin between different classes.

Random Forest: We implement Random Forest using RandomForestClassifier. Initially, we employ 80 n_estimators, which creates 80 individual decision trees. We use max_leaf_nodes=20 to limit the maximum number of leaf nodes in each tree, which can prevent overfitting and control the tree depth. We also set random_state=42 to ensure reproducibility of results. During prediction, this model combines the predictions from all the trees and assigns the test sample to the class that gets the most votes.

3.1.2 Hyperparameter Finetuning Strategies

After implementing four models, the next step was to fine-tune their hyperparameters to achieve their best performance. Hyperparameter finetuning helps find the best settings by testing different values and selecting the ones that give the best results. To achieve this goal, we used Grid search cross-validation (GridSearchCV), a method that searches through a defined parameter grid using cross-validation to find the best model for each machine learning approach. Once the parameters with the best cross-validation performance are found, the method automatically fits a new model to the entire training dataset (Alhakeem et al., 2022).

(1) For K-Nearest Neighbour, the main hyperparameter is the number of neighbours (k), which decides how many neighbours are used to make a classification. We tested values of k = [1, 3, 5, 7] to find the best number of neighbours. In terms of distance metric, we tried both Manhattan distance (p=1) and Euclidean distance (p=2) to choose the most effective distance metric for the dataset. GridSearchCV was used to test different combinations of n_neighbors and p. The cv=10 argument sets the cross-validation to 10 folds, which means the model will be tested using 10 different train-test splits of the data. By setting return_train_score=True, we ask the grid search to give both the training and validation scores, so we can compare how the model performs on the training data versus the validation data.

(2) For Logistic Regression, the main hyperparameters we tested include the ‘solver’, which decides the optimization algorithm used to find the best model. We tried two solvers: 'lbfgs' and 'liblinear'. Another important hyperparameter is the regularisation strength (C), which controls how much regularisation we apply to avoid overfitting. We tested values of C = [0.1, 1, 10, 100, 1000]. Additionally, we explored different ‘tol’ values [1e-4, 1e-3], which determine when the optimization algorithm should stop. And we also tried ‘class_weight’ = ‘None’ and ‘balanced’ to see if adjusting the weights would improve performance. GridSearchCV is used as before.

(3) For Support Vector Machine (SVM), we defined a few important hyperparameters to fine-tune the model. The C parameter controls the regularisation strength, and we tested C = [0.1, 1, 10] to find the best balance between overfitting and underfitting. The gamma parameter influences how much a single training sample impacts the decision boundary, and we tested gamma = [1, 'scale']. Finally, we explored different kernel types: ‘linear’, ‘poly’, and ‘rbf’ to see which one fits the data best. GridSearchCV is used as before.

(4) For Random Forest, we defined several important hyperparameters to fine-tune the model. The n_estimators parameter controls the number of trees in the forest, and we tested n_estimators = [80, 100] to see how the number of trees affects the model’s performance. The max_leaf_nodes parameter limits the maximum number of leaf nodes in each tree, and we tried max_leaf_nodes = [20, None] to control the depth of the trees. Finally, the criterion parameter decides how the model measures the quality of a split, and we tested 'entropy' and 'gini' to find the best one. GridSearchCV is used as before.

3.2 Model Performance

In this section, we present the results obtained from the 4 selected models and discuss their implications.

(1) The KNN model achieved an accuracy of 85.9% after hyperparameter finetuning. It performed well in predicting distinct items such as trousers and sneakers but struggled with more similar categories, like pullovers and shirts. KNN's inference time is longer than training time, which means it may not be suitable for real-time applications that require quick predictions.

The results suggest that KNN is good for categories with distinct features but is not ideal for overlapping classes. This implies that KNN may work well in scenarios where classes are easy to distinguish, but not for more complex datasets with closely related categories.

(2) The Logistic Regression model achieved an accuracy of 85.65% after hyperparameter finetuning. It worked well on trousers and bags but struggled with pullovers and shirts. Logistic Regression's training time is longer that inference time.

The results suggest that Logistic Regression performed consistently well across all classes, but its weakness in some overlapping categories. Its fast t prediction times make it an ideal choice for applications where speed is critical, like online classification.

(3) The SVM model achieved an accuracy of 91.1% after hyperparameter finetuning. It was excellent at predicting trousers and bags. Even for harder classes like pullovers and shirts, SVM still performed well.

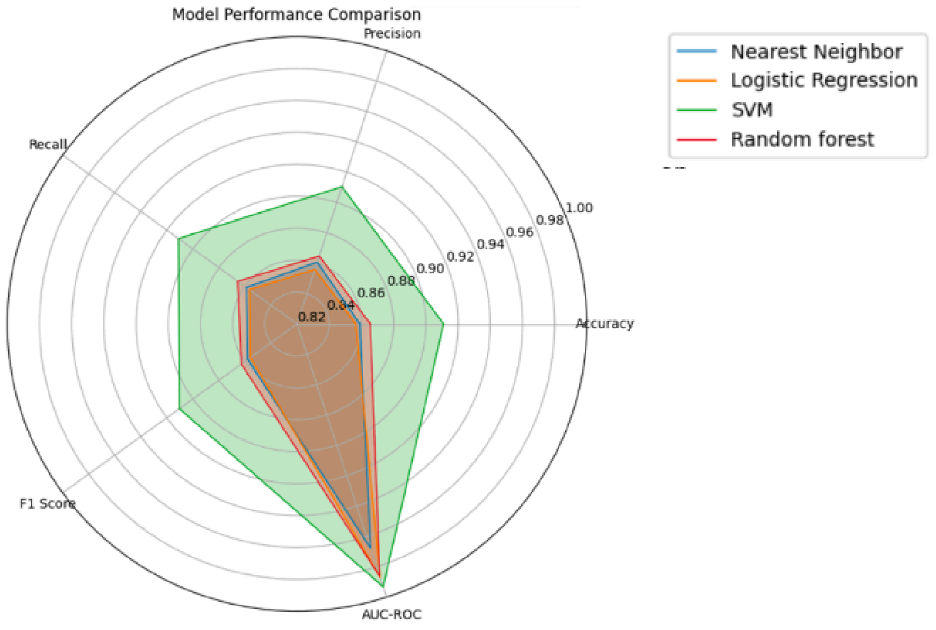
The results suggest that SVM works very well with complex, non-linear datasets because it can handle difficult class boundaries with the rbf kernel. However, the long training time could be a problem in situations where fast model updates are needed.

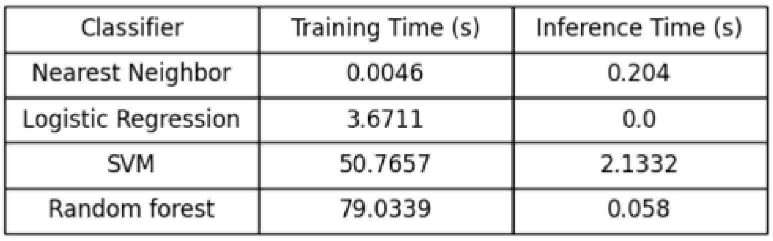
(4) The Random Forest model achieved an accuracy of 86.55% after hyperparameter fine-tuning. It performed well on trousers and bags but struggled with pullovers and shirts. Random Forest's training time is much longer than its inference time.

The results suggest that the Random Forest model shows strong predictive capabilities for most classes, however, for more overlapping or visually similar classes, the model faces some difficulties, leading to a lower accuracy for those categories.

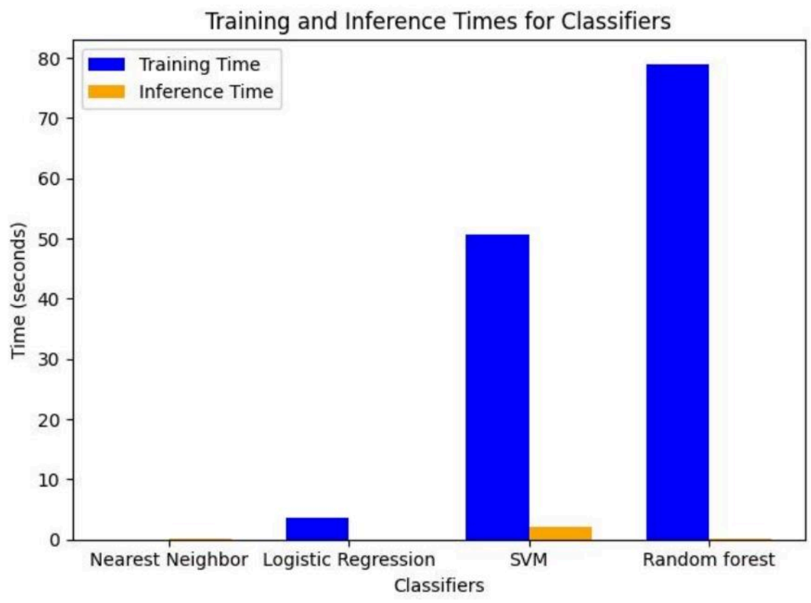
3.3 Model Comparisons

In comparing the performance of each model, we used key evaluation metrics such as accuracy, precision, recall, F1 score, AUC-ROC, training time and inference time, in the form of a radar chart, table and bar table.





Classifier	Training Time (s)	Inference Time (s)
Nearest Neighbor	0.0046	0.204
Logistic Regression	3.6711	0.0
SVM	50.7657	2.1332
Random forest	79.0339	0.058



4. Conclusion

4.1 Main finding

After testing the models on the test dataset, we found that the Support Vector Machine (SVM) model achieves the best performance. The SVM model achieves an accuracy of 91.1%, with precision, recall, and F1-score all at 0.91, which shows that this model is well-balanced and performs strongly across most classes. However, all four models, including SVM, show poor preference when predicting class 6.

4.2 Limitation of models

One major limitation of all four models is their poor performance when predicting class 6. In the results, we can observe that many samples from class 6 (shirts) were incorrectly classified as class 0 (T-shirt). This is likely because these two classes share similar features, making it difficult for the models to differentiate between them. This highlights a problem with how models handle certain similar features in the dataset which means further improvements are needed in this area.

4.3 Potential Direction

To improve the models in the future, especially their performance for class 6, we may need to explore additional methods. One potential direction is to use Convolutional Neural Networks (CNNs), which are known for their strong ability to extract complex features from data, especially image data (Li et al., 2021). By introducing CNNs, we can refine the feature extraction process, and additionally, feature engineering could help identify more distinct characteristics for classes that are difficult to separate, further enhancing model performance.

References

Alhakeem, Z. M., Jebur, Y. M., Henedy, S. N., Imran, H., Bernardo, L. F., & Hussein, H. M. (2022). Prediction of ecofriendly concrete compressive strength using gradient boosting regression tree combined with GridSearchCV hyperparameter-optimization techniques. *Materials*, *15*(21), 7432. https://doi.org/10.3390/ma15217432

Ali, P. J. M., Faraj, R. H., Koya, E., Ali, P. J. M., & Faraj, R. H. (2014). Data normalization and standardization: a technical report. *Mach Learn Tech Rep*, *1*(1), 1-6.

Jakkula, V. (2006). Tutorial on support vector machine (svm). *School of EECS, Washington State University*, *37*(2.5), 3.

Kaiser, J. (2014). Dealing with Missing Values in Data. *Journal of Systems Integration, (1804-2724), 5*(1), 42-51.

Kanade, V. (2022). *What Is Logistic Regression? Equation, Assumptions, Types, and Best Practices.* Retrieved September 12, 2024 from https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-logistic-regression/

Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2021). A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*, *33*(12), 6999-7019. https://doi.org/10.1109/TNNLS.2021.3084827

Maćkiewicz, A., & Ratajczak, W. (1993). Principal components analysis (PCA). *Computers & Geosciences*, *19*(3), 303-342. https://doi.org/10.1016/0098-3004(93)90090-R

Nandi, D., Ashour, A. S., Samanta, S., Chakraborty, S., Salem, M. A., & Dey, N. (2015). Principal component analysis in medical image processing: a study. *International Journal of Image Mining*, *1*(1), 65-86. https://doi.org/10.1504/IJIM.2015.070024

Nick, T. G., & Campbell, K. M. (2007). Logistic regression. *Topics in biostatistics*, *vol 404,* 273-301. https://doi.org/10.1007/978-1-59745-530-5_14

Parmar, A., Katariya, R., & Patel, V. (2019). *International conference on intelligent data communication technologies and internet of things (ICICI) 2018*. New York, America: Springer International Publishing.

Singh, P. V. (2023). *All About Min-max scaling.* Retrieved September 12, 2024 from https://medium.com/@poojaviveksingh/all-about-min-max-scaling-c7da4e0044c5

Tabsharani, F. (2023). *Support vector machine (SVM).* Retrieved September 13, 2024 from https://www.techtarget.com/whatis/definition/support-vector-machine-SVM

Tibrewal, T. P. (2023). *Support Vector Machines (SVM): An Intuitive Explanation.* Retrieved September 13, 2024 from https://medium.com/low-code-for-advanced-data-science/support-vector-machines-svm-an-ituitive-explanation-b084d6238106

Van Der Maaten, L., Postma, E., & Van den Herik, J. (2009). Dimensionality reduction: a comparative. *J Mach Learn Res*, *10*(66-71).

Yue, S., Li, P., & Hao, P. (2003). SVM classification: Its contents and challenges. *Applied Mathematics-A Journal of Chinese Universities*, *18*, 332-342. https://doi.org/10.1007/s11766-003-0059-5

Zhang, S., Li, X., Zong, M., Zhu, X., & Wang, R. (2017). Efficient kNN classification with different numbers of nearest neighbors. *IEEE transactions on neural networks and learning systems*, *29*(5), 1774-1785. https://doi.org/10.1109/TNNLS.2017.2673241

Appendices

In this guide, we will deliberately introduce how to set up the environment to run the code. In this appendices, we will provide 2 ways to run the code.

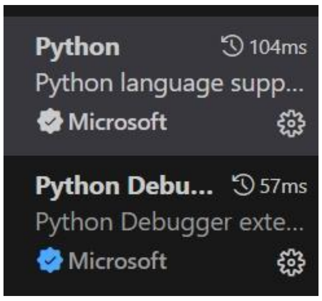
Way 1: Using VS code software to run the code on the local computer

1 Install python

The first step is to install python 3.12.1, it is easy to install this version of python on the python official website. After installing the python 3.12.1, we need to put its path into the computer environment variable to make sure the python can run normally in the following steps.

2 Install VS code

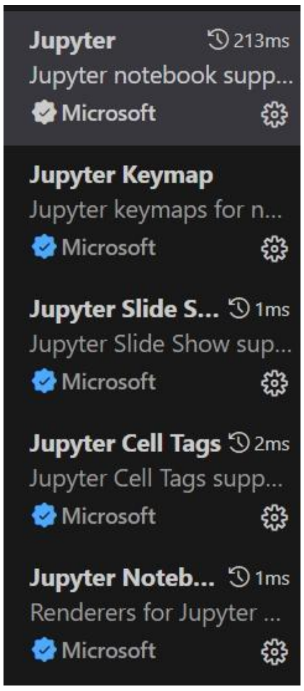
When finishing installing the python 3.12.1, we should have a compiler to run our code. Here we choose VS code as an example. We can also easily install the VS code from the official website. Then we should install the python and python debug extension of the VS code. Just like the following picture shows:



Now we have an environment that can run python code.

3 Create the training environment of machine learning

To make the VS code run the .ipynb file normally, we firstly need to install the extension about Jupyter. Just like the picture on the right shows:

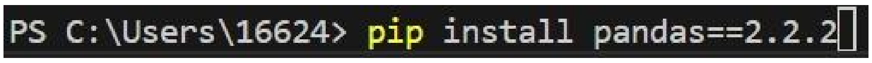


Then we should install those packages that we may need to run the code.The packages and their versions that we use are:
pandas: 2.2.2, matplotlib: 3.8.4, numpy: 1.26.4,
scikit-learn: 1.4.2.

We can use the following commands to install the packages:
pip install pandas==2.2.2, pip install matplotlib==3.8.4,
pip install numpy==1.26.4, pip install scikit-learn==1.4.2.

We just input these commands into the terminal of VS code and press the "enter" button on the keyboard, then the packages will be successfully installed on our local computer.

Just like the following picture shows (use pandas package as an example):

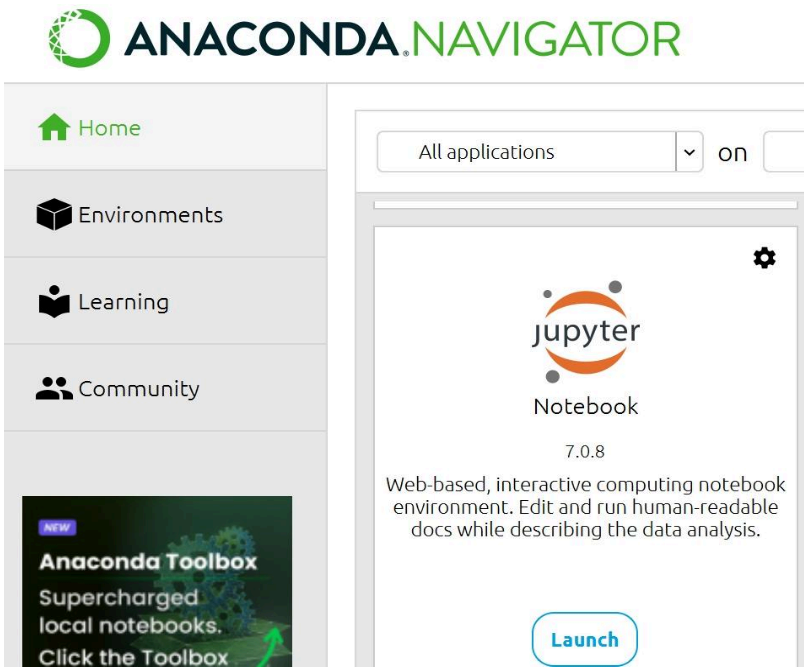


```
PS C:\Users\16624> pip install pandas==2.2.2
```

Using the steps above, we can easily use VS code to run the code normally.

Way 2: Using anaconda to launch Jupyter notebook

There is also a more convenient way to create the environment for running the code. Firstly we can install anaconda software from the official website. Then we use anaconda to launch its Jupyter notebook, just like the following picture shows:


ANACONDA.NAVIGATOR
Home
Environments
Learning
Community
NEW
Anaconda Toolbox
Supercharged local notebooks.
Click the Toolbox
All applications
on
jupyter
Notebook
7.0.8
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
Launch

In the jupyter notebook of the anaconda virtual environment, we can directly run the code normally as the anaconda default virtual environment has already had the python version and its common packages, so we don't need to install them on our own.

In the anaconda default virtual environment, the versions of python and its common packages are as follows:

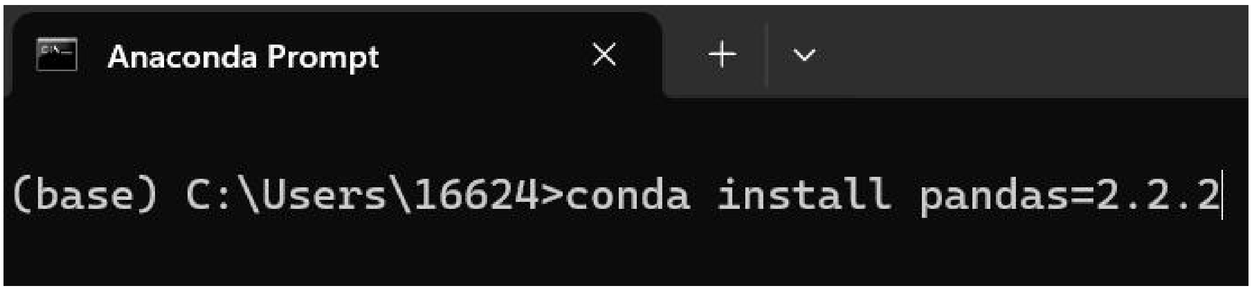
Python: 3.12.3,   pandas: 2.2.2, matplotlib: 3.8.4, numpy: 1.26.4, scikit-learn: 1.4.2.

If the default anaconda virtual environment doesn't have these packages, we can also use the anaconda prompt to install these packages manually.

To install the packages in the anaconda virtual environment, we can use the following commands:

conda install -c conda-forge python=3.12.3, conda install pandas=2.2.2,
conda install matplotlib=3.8.4,   pip install numpy==1.26.4,
pip install scikit-learn==1.4.2.

We just input these commands into the anaconda prompt and press the "enter" button on the keyboard, then the packages will be successfully installed on the anaconda virtual environment. Just like the following picture shows (use pandas package as an example):



```
(base) C:\Users\16624>conda install pandas=2.2.2
```

Using the steps above, we can easily use anaconda to launch the Jupyter notebook and run the code normally.

Hardware requirement:

To make sure that the code can run normally on the computer, it is advised that the hardware equipment can satisfy the requirements as follows:

CPU:

(1) Type: Apple M2 or higher version

(2) Number of Cores: 8 (4 performance and 4 efficiency) or higher

RAM: 16GB or higher

OS Loader Version: 8422.141.2 or higher

Chip: Apple M2 or higher