INT104 Coursework 2 Report Guidance

Jing YIN (2034355)

Lab-B-Group-3

April 30, 2022

1 Introduction

In the provided table, there were data for a total of 514 students. Each student has grades for five courses. This report proposes observing and counting the dataset and classifying it using supervised or unsupervised machine learning algorithms. Specifically, data observations were first conducted to count how many students were in each major and the median, mean, maximum, minimum and standard deviation of each student's score. In addition, a principal component analysis (PCA) was performed on the data, and the optimal number of dimension reductions was selected for the later experiments. To classify the students, three algorithms, k-nearest neighbour, logistic regression and neural network, were implemented for supervised learning, and the k-means algorithm was implemented for unsupervised learning. In the experimental part, the data were divided into training and test sets according to 4:1, and the models were evaluated on the test set using a 5-fold cross-validation approach. The evaluation metrics used were those commonly used in the four classification algorithms, including classification accuracy (accuracy), precision (accuracy), recall (recall) and the f1 value (the summed average of precision and recall). In addition, PCA is combined with dimensionality reduction of the data to select the optimal number of dimensions for classification. As a result, there are eight different combinations of models. The experimental results show that the combination of PCA + LR models achieves optimal results on the test set, thus achieving the experimental objective of "classifying students into their majors according to the scores of each question".

2 Data Observation

In machine learning tasks, usually, all data contain biases for different reasons. Biases in certain data features may affect a suitable classifier for a machine learning task, so showing data bias is necessary. In the experiments, for this reason, basic statistics (e.g. mean, median, range and standard deviation) were first done on the data using the third-party library pandas, which was solved by some built-in methods of data frame, and the data was visualised using matplotlib. While these data frame methods are very commonly used, data features such as mean, median, range and standard deviation are not necessary to be features that show data deviations. Data features generally come from more complex mathematical operations such as principal component analysis (PCA), discrete Fourier transform (DCT) and non-negative matrix decomposition (NMF). In this experiment, principal component analysis (PCA) was performed in the feature analysis with the help of the PCA method from the scikit-learn library. In addition, to estimate what value of dimensionality is best achieved when PCA is reduced to different dimensions, data reduced to different dimensions are trained and tested on different models, and the optimal dimensionality is selected by evaluating the metrics on the test set. The experimental results were analysed, and the four feature values with high importance of feature values were selected, i.e. five dimensions were reduced to 4 dimensions. The extracted raw data features are then visualised data to show the deviations in their distribution. There may be deviations in the distribution for different features, which are addressed in this experiment using normalisation.

For the selection of input features, this experiment is divided into two types, one where all five features are used for classification, and the other where the features after dimensionality reduction, this is determined by tuning the parameters to determine that a dimensionality reduction of 4 is optimal and they will all be used as input to the classifier developed in the next task.

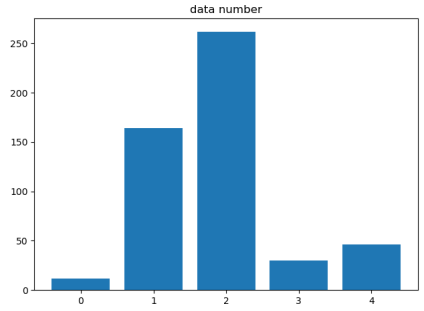
2.1 Basic statistics

(1) How many students are in each of the different majors

Code demonstrate:

```
plt.figure()
plt.title('data number')
plt.bar(['0','1','2','3','4'],[len(class_0),len(class_1),len(class_2),len(class
plt.show()
```

result demonstrate:



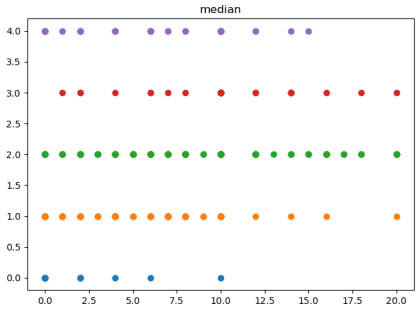
(2) Median score per student

Code demonstrate:

```
plt.figure()
plt.title('median')
```

```
a=np.array(class_0)
b=np.array(class_1)
c=np.array(class_2)
d=np.array(class_3)
e=np.array(class_4)
plt.scatter(np.median(a,-1),np.zeros(len(class_0)))
plt.scatter(np.median(b,-1),np.zeros(len(class_1))+1)
plt.scatter(np.median(c,-1),np.zeros(len(class_2))+2)
plt.scatter(np.median(d,-1),np.zeros(len(class_3))+3)
plt.scatter(np.median(e,-1),np.zeros(len(class_4))+4)
plt.show()
```

result demonstrate:



(3) Average of each student's score:

Code demonstrate:

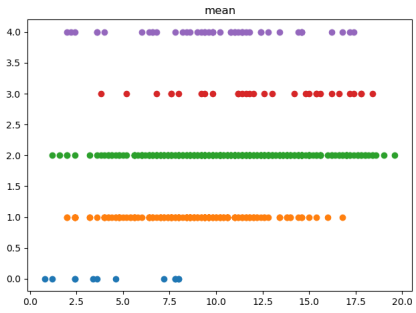
```
plt.figure()
plt.title('mean')
plt.scatter(np.array(class_0).mean(-
1),np.zeros(len(class_0)))
plt.scatter(np.array(class_1).mean(-
1),np.zeros(len(class_1))+1)
plt.scatter(np.array(class_2).mean(-
1),np.zeros(len(class_2))+2)
plt.scatter(np.array(class_3).mean(-
1),np.zeros(len(class_3))+3)
plt.scatter(np.array(class_4).mean(-
1),np.zeros(len(class_4))+4)
plt.show()
```

result demonstrate:

(4) Maximum score for each student:

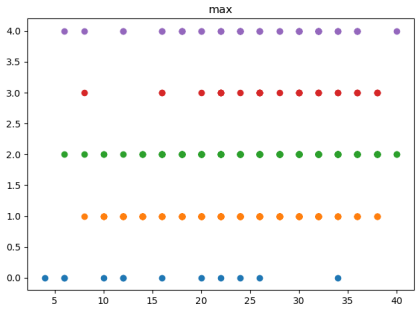
Code demonstrate:

```
plt.figure()
plt.title('mean')
plt.scatter(np.array(class_0).mean(-
1),np.zeros(len(class_0)))
```




```
plt.scatter(np.array(class_1).mean(-
1),np.zeros(len(class_1))+1)
plt.scatter(np.array(class_2).mean(-
1),np.zeros(len(class_2))+2)
plt.scatter(np.array(class_3).mean(-
1),np.zeros(len(class_3))+3)
plt.scatter(np.array(class_4).mean(-
1),np.zeros(len(class_4))+4)
plt.show()
```

result demonstrate:

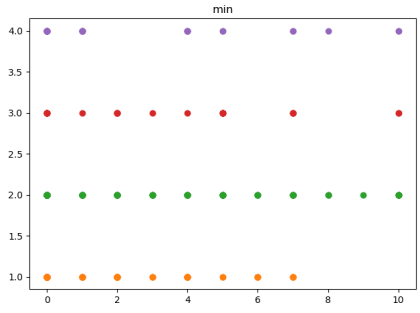


(5)The minimum value of each student's score:

Code demonstrate:

```
plt.figure()
plt.title('min')
plt.scatter(np.array(class_0).min(-
1),np.zeros(len(class_0)))
plt.scatter(np.array(class_1).min(-
1),np.zeros(len(class_1))+1)
plt.scatter(np.array(class_2).min(-
1),np.zeros(len(class_2))+2)
plt.scatter(np.array(class_3).min(-
1),np.zeros(len(class_3))+3)
plt.scatter(np.array(class_4).min(-
1),np.zeros(len(class_4))+4)
plt.show()
```

result demonstrate:

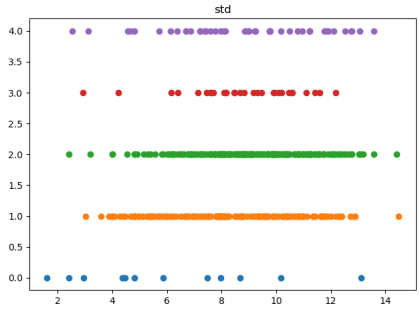


(6) Standard deviation of each student's score:

Code demonstrate:

```
plt.figure()
plt.title('std')
plt.scatter(np.array(class_0).std(-
1),np.zeros(len(class_0)))
plt.scatter(np.array(class_1).std(-
1),np.zeros(len(class_1))+1)
plt.scatter(np.array(class_2).std(-
1),np.zeros(len(class_2))+2)
plt.scatter(np.array(class_3).std(-
1),np.zeros(len(class_3))+3)
plt.scatter(np.array(class_4).std(-
1),np.zeros(len(class_4))+4)
plt.show()
```

result demonstrate:



2.2 PCA analysis

Principal Component Analysis (PCA) algorithms are often used for dimensionality reduction of data. Dimensionality reduction is a method of pre-processing high-dimensional data by keeping some of the most critical features and removing noise and unimportant features, thereby increasing data processing speed. This is usually done by calculating the covariance matrix (the degree of correlation of the current features) of the data matrix, then obtaining the eigenvectors of the eigenvalues of the covariance matrix and selecting the matrix consisting of the eigenvectors corresponding to the k features with the largest eigenvalues (i.e. the largest variance). This allows the data matrix to be transformed into a new space and enables the dimensionality reduction of the data features.

PCA essentially completes the feature selection process during the process of dimensionality reduction. Expressly, in the process of dimensionality reduction, the algorithm retains only the "principal components" of the data. At the same time, the minor components are ignored so that the data is also noise-reduced in the process of dimensionality reduction, thus achieving feature selection. In order to select the optimal number of dimension reductions, this experiment uses iterative dimensioning. It calculates the metrics on the test set to select the dimension with the best performance on the test set (see Experiments section for details).

3 Candidate classifiers

In the supervised classifier, three classifiers, KNN, logistic regression LR and neural network NN, were designed for this experiment; in the unsupervised classification, the k-Means algorithm was chosen to classify the programs registered by students based on the scores of each question in the exam. Two types of input features were used in this experiment. One was the original data from the dataset where all five features were used for classification. The other was the features after downscaling to 4 dimensions. The evaluation metrics calculated on the test set were used to determine which combination of model and features would give better performance. Critical parts of the procedure, such as the data preprocessing, the model's training process, and the inference process of the system, will be detailed in the report.

3.1 k Nearest Neighbors, kNN

The k-nearest neighbour (kNN) algorithm is a commonly used supervised learning method. The basic idea is that given a set of data, the k closest training samples in the training set are identified based on some distance measure, and then predictions are made based on the information of these k neighbours. Typically, voting is used in classification tasks. The most frequent

category marker among the k samples is chosen as the prediction result, and averaging is used in regression tasks. The average of the real-valued output markers of the k samples is used as the prediction result. This algorithm is one of the "lazy learning" methods, where there is no explicit learning process. Correspondingly, those methods that learn from the samples during the training phase are called "eager learning".

3.2 Logistic Regression, LR

The logistic regression algorithm is a generalised linear model commonly used to do dichotomous tasks and calculate the probability of predicting the corresponding class.

(1) Equation expressions for logistic regression
The loss function can be constructed for logistic regression by either the excellent likelihood estimation or the cross-entropy loss function. Firstly, in logistic regression, the sigmoid function is used as the function to be fitted to solve the problem that the regression function is severely affected by outliers, so the general equation form of logistic regression is as follows.

$y = \frac{1}{1+e^{-(wx^T+c)}}$

(2) Loss function for logistic regression
$y = \frac{1}{1+e^{-(wx^T+c)}}$, where x is the feature matrix, m is the number of features and n is the number of rows.

$$X = \begin{bmatrix} x_{11}, x_{12}, x_{13,} \dots x_{1m} \\ x_{21}, x_{22}, x_{23,} \dots x_{2m} \\ \cdot \\ \cdot \\ \cdot \\ x_{n1}, x_{n2}, x_{n3,} \dots x_{nm} \end{bmatrix}$$

And, the i-th data is $X_i = [x_{i1}, x_{i2}, x_{i3,} \dots x_{im}]$
And the model independent variable parameters are $w = [w_1, w_2, w_3 \dots w_m]$, the model intercept parameter is c. The true category of data i is used to denote the positive category and the negative category is -1 in order to comply with the derivation of the loss function in sklearn. the output of data i in the logistic regression is $\frac{1}{1+e^{-(wx^T+c)}}$. The probability of the ith data being of class 1 is expressed in terms of the probability of the ith data being of class 1 when given a set of w and c. The probability of the ith data being of class 1 can be calculated by bringing in a piece of data:
$P = (y_i = 1 \mid X_i, w, c) = \frac{1}{1+e^{-(x_1^T w+c)}}$
and the probability of belonging to category -1 is 1 - p:
$P = (y_i = -1 \mid X_i, w, c) = 1 - \frac{1}{1+e^{-(x_1^T w+c)}}$

The probability of the true class of the ith training sample is therefore $P = (X_i, w, c) = \frac{1}{1+e^{-(x_i^T w+c)\cdot y_1}}$
The goal of logistic regression modelling is to make the probability of the true category of the training sample as large as possible, i.e. to make the following equation as large as possible:

$$\prod_{i=1}^{n} \frac{1}{1+e^{-\left(x_i^\top w+c\right)\cdot y_i}}$$

The maximum value of the cumulative multiplication is equivalent to the minimum value of the cumulative result after taking the logarithm and multiplying by -1, i.e. to find the minimum value of

$$\sum_{i=1}^{n} \log\left(1+e^{-\left(X_1^\top w+c\right)\cdot y_1}\right)$$

.
At this point, the loss function for logistic regression with binary labeling at levels of 1 and -1 is introduced, and it is used to evaluate the training progress of the model.

3.3 Neural Network, NN

A neural network is a hierarchical structure based on the algorithms of a perceptron. It is built on the structure and working principles of the human brain's neurons. It is a structure of nodes linked together, called neurons, which in most cases are used to store values that represent weights, and synapses are abstracted as weighted connections that determine how the numbers will be changed and to which neurons. Although the basic unit for building a neural network is the neuron, it is divided into input, implicit, and output layers. Each layer in the implicit layer tries to recognise the various features entered by the input layer, and after a series of calculations, the output layer answers.

3.4 k-Means

The three algorithms mentioned above are all supervised learning algorithms. Another class of algorithms in machine learning is called unsupervised learning, such as the k-Means algorithm. K-Means is a clustering algorithm that automatically groups similar samples into a category, a process that does not require supervised data.

The computational steps of the K-means algorithm can be summarised in two parts: (i) calculate the distance from each object to the centre of the class cluster;(ii)

calculate a new cluster class centre based on the objects within the class cluster.

Let x denote a sample point in a cluster, u denote the centre of mass in that cluster, n denote the number of features in each sample point, and i denote each feature of the constituent points. The following distances can measure the distance from that sample point to the centre of mass.

(1) Euclidean distance:$\mathrm{d}(\mathrm{x},\mu)=\sqrt{\sum_{\mathrm{i}=1}^{\mathrm{n}}\left(\mathrm{x_i}-\mu_{\mathrm{i}}\right)^2}$
(2) Manhattan distance:$\mathrm{d}(\mathrm{x},\mu)=\sum_{\mathrm{i}=1}^{\mathrm{n}}\left(\left|\mathrm{x_i}-\mu\right|\right)$
(3) Cosine distance:$\cos\theta=\frac{\sum_1^n(x_i\cdot\mu)}{\sqrt{\sum_1^n(x_1)^2}\cdot\sqrt{\sum_1^n(\mu)^2}}$

If Euclidean distances are used, the sum of the squares of the distances from all sample points in a cluster to the centre of mass is :
Cluster Sum of Square (CSS) = $\sum_{\mathrm{j}=0}^{\mathrm{m}}\sum_{\mathrm{i}=1}^{\mathrm{n}}\left(\mathrm{x_i}-\mu_{\mathrm{i}}\right)^2$
Total Cluster Sum of Square = $\sum_{\mathrm{i}=1}^{\mathrm{k}}\mathrm{CSS_i}$

Where m is the number of samples in a cluster and j is the number of each sample. This formula is called the Cluster Sum of Square, or Inertia, and adding up the sum of the intra-cluster squares of all the clusters in a dataset gives the Total Cluster Sum of Square, also called total Inertia. The smaller the total Inertia, the more similar the samples within each cluster are, and the better the clustering effect. Thus KMeans seeks to solve for the centre of mass that minimises the Inertia. The overall sum of squares gets smaller and smaller as the centre of mass keeps changing and iterating. We can use mathematics to show that when the overall sum of squares is minimised, the centre of mass ceases to change.

4 Evaluation and selection of candidate classifiers

4.1 Five features are used for classification

(1)**Data pre-processing:** firstly, read data from the csv file using the pandas library, then normalise the data, turning each feature into a normal distribution with mean 0 and variance 1, then separate the training data from the test data.

```
# Read data

df=pd.read_csv('./CW_Data.csv', sep=',', header=0)
q1=list(df['Q1'])
q2=list(df['Q2'])
q3=list(df['Q3'])
q4=list(df['Q4'])
q5=list(df['Q5'])
cls=list(df['Programme'])

X=[ ] y=[ ] for i in range(len(q1)):
X.append([q1[i], q2[i], q3[i], q4[i], q5[i]])
y.append(cls[i])

X = np.array(X)
y = np.array(y)
print(X.shape, y.shape)

# Data standardisation

X_std = StandardScaler().fit_transform(X)

# Training test data separation

X_train,X_test,y_train,y_test=train_test_split(X_std,
y,test_size=0.2,random_state=666)
print(X_train.shape,y_train.shape,
X_test.shape,y_test.shape)
```

(2)**The training process of the model:**

In the process of defining the model, for the kNN algorithm, the number of nearest neighbours is selected as five using tuning the parameters, and the weighting method is that of calculating the distance; for the logistic regression algorithm, L2 paradigm regularisation is used to prevent overfitting; for the neural network algorithm, a four-layer neural network structure is defined, with the loss functions of the first few layers being the commonly used ReLU loss functions and the final layer being Softmax, which is used to learn the classification task. For the k-means algorithm, the number of clusters passed in is 5, as five categories exist in the original data.

```
# Supervised classification

# Define classifier
knn=KNeighborsClassifier(n_neighbors=5,weights='dis
-tance')
lr=LogisticRegression(penalty='l2')

# Model training and testing
knn.fit(X_train,y_train)
lr.fit(X_train,y_train)

knn_result=knn.predict(X_test)
lr_result=lr.predict(X_test)
```

```
knn_acc, knn_recall, knn_pre, knn_f1 = evaluate(y_test,
knn_result)
lr_acc, lr_recall, lr_pre, lr_f1=evaluate(y_test, lr_result)
with open('result.txt', 'a', encoding='utf-8')as f:
print('kNN acc:'+str(knn_acc)+',recall:'+str(knn_recall)+
',precision:'+str(knn_pre)+',f1:'+str(knn_f1))
print('\n')
print('LR acc:'+str(lr_acc)+',recall:'+str(lr_recall)+',pre
-cision:'+str(lr_pre)+',f1:'+str(lr_f1))
print('\n')

# Neural networks (tensorflow build)

model = tf.keras.models.Sequential([
tf.keras.layers.Dense(128, activation='relu'),
tf.keras.layers.Dense(64, activation='relu'),
tf.keras.layers.Dense(64, activation='relu'),
tf.keras.layers.Dense(5, activation='softmax')
])

model.compile(optimizer='adam',
loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

model.fit(X_train,y_train,validation_data=(X_test,y_test
),epochs=50)
nn_result=model.predict(X_test)
nn_result=np.argmax(nn_result,axis=1)

nn_acc,nn_recall,nn_pre,nn_f1=evaluate(y_test,nn_result)
with open('result.txt', 'a',encoding='utf-8')as f:
print('NN      acc:'+str(nn_acc)+',recall:'+str(nn_recall)
+', precision:'+ str(nn_pre)+',f1:'+str(nn_f1))
print('\n')

# Unsupervised classification

kmeans=(n_clusters=5)
kmeans_result=kmeans.fit_predict(X_test)
kmeans_acc,kmeans_recall,kmeans_pre,kmeans_f1=evalu
ate(y_test,kmeans_result)
with open('result.txt','a', encoding='utf-8') as f:
print('K-means acc:'+str(kmeans_acc)+',recall:'+str(km
eans_recall)+',precision:'+str(kmeans_pre)+',f1:'+str(k
means_f1))
print('\n')
```

4.2 Features after dimensionality reduction to 4 dimensions are used for classification

(1)**Data pre-processing:** Unlike with all features, the data is first reduced in dimension using PCA before performing the following steps.

```
# Use PCA to reduce the dimensionality, verify
the best results by tuning the reference to 4 dimensions
pca = PCA(n_components=4)
pca.fit(X_train)
X_train = pca.transform(X_train)
X_test = pca.transform(X_test)
```

(2)**Supervised classification**
with parameters defined and for the same reasons as in section 4.1.

```
# Define classifier
knn=KNeighborsClassifier(n_neighbors=5,
weights='distance')
lr=LogisticRegression(penalty='l2')

# Model training and testing
knn.fit(X_train, y_train)
lr.fit(X_train, y_train)

knn_result=knn.predict(X_test)
lr_result=lr.predict(X_test)

knn_acc,knn_recall, knn_pre, knn_f1=evaluate(y_test,
knn_result)
lr_acc, lr_recall, lr_pre, lr_f1 = evaluate(y_test, lr_result)
with open('result.txt', 'a', encoding='utf-8') as f:
print('kNN+PCA acc:'+str(knn_acc)+',recall:'+str(knn
_recall)+',precision:'+str(knn_pre)+',f1:'+str(knn_f1))
print('\n')
print('LR+PCA acc:'+str(lr_acc)+',recall:'+str(lr_recall)
+',precision:'+str(lr_pre)
+',f1:'+str(lr_f1))
print('\n')

# Neural networks (tensorflow build)

model=tf.keras.models.Sequential([
tf.keras.layers.Dense(128, activation='relu'),
tf.keras.layers.Dense(64, activation='relu'),
tf.keras.layers.Dense(64, activation='relu'),
tf.keras.layers.Dense(5, activation='softmax')
])

model.compile(optimizer='adam',
loss='sparse_categorical_crossentropy',
```

```
metrics=['accuracy'])

model.fit(X_train,y_train,validation_data=(X_test,y_test
),epochs=50)
nn_result = model.predict(X_test)
nn_result = np.argmax(nn_result, axis=1)

nn_acc,nn_recall,nn_pre,nn_f1=evaluate(y_test,
nn_result)
with open('result.txt', 'a', encoding='utf-8') as f:
print('NN+PCA acc:'+str(nn_acc)+',recall:'+str(nn_rec
all)+', precision:'+str(nn_pre)+',f1:'+str(nn_f1))
print('\n')
```

(3)**Unsupervised classification**

```
kmeans=KMeans(n_clusters=5)
kmeans_result=kmeans.fit_predict(X_test)
kmeans_acc,kmeans_recall,kmeans_pre,
kmeans_f1=evalu
ate(y_test, kmeans_result)
with open('result.txt','a', encoding='utf-8')as f:
print('K-means+PCA acc:'+str(kmeans_acc)+',recall:'+
str(kmeans_recall)+',precision:'+str(kmeans_pre)+',f1:'
+str(kmeans_f1))
print('\n')
```

4.3 Select candidate classifiers

(1) The inference process of the system
In sections 4.1 and 4.2, for the k-nearest neighbour and logistic regression algorithms, the fit function in the scikit-learn library is used to train the model; the predict function is used to cause the model to make predictions on the test set. The fit and predict functions in the TensorFlow library can also be used for the training and prediction of neural network algorithms. For k-means algorithms, the fit_predict function can be used directly to train and predict in one go. After obtaining the predicted y_pred result, the four evaluation metrics can be calculated with the actual label y_true using the following code.

```
def evaluate(y_test, y_pred):
acc = accuracy_score(y_test, y_pred)
recall = recall_score(y_test, y_pred, average='macro')
pre = precision_score(y_test, y_pred, average='macro')
f1 = f1_score(y_test, y_pred, average='macro')
return acc, recall, pre, f1;
```

(2) Experimental results
The experiments were conducted using a five-fold cross-validation strategy for training and testing. Specifically, the data were divided into five copies, and each time one of the copies was selected as the training set, the remaining copy was selected as the test set, and so on, repeated five times, and the results of the five evaluations on the test set were finally averaged.

The results of the five-fold cross-validation are shown below.

First fold:

	acc	recall	precision	f1
kNN	0.46	0.2275	0.2936	0.2272
LR	0.51	0.3026	0.2644	0.2730
K-means	0.2	0.1474	0.2709	0.1580
NN	0.5	0.3330	0.4125	0.3454
PCA+kNN	0.5	0.2474	0.3118	0.2427
PCA+LR	0.55	0.3258	0.2847	0.2961
PCA+K-means	0.1	0.0789	0.1304	0.0773
PCA+NN	0.53	0.2502	0.2172	0.2305

Second fold:

	acc	recall	precision	f1
kNN	0.56	0.2567	0.2380	0.2382
LR	0.61	0.3033	0.2727	0.2847
K-means	0.18	0.1152	0.1680	0.1143
NN	0.52	0.2631	0.2464	0.2502
PCA+kNN	0.57	0.2597	0.2389	0.2404
PCA+LR	0.63	0.3179	0.2812	0.2960
PCA+K-means	0.14	0.0689	0.1325	0.0787
PCA+NN	0.57	0.3030	0.2719	0.2797

Third fold:

	acc	recall	precision	f1
kNN	0.55	0.2802	0.2808	0.2628
LR	0.56	0.3240	0.3066	0.2821
K-means	0.3	0.1541	0.2300	0.1690
NN	0.51	0.2388	0.2398	0.2022
PCA+kNN	0.52	0.2687	0.2708	0.2487
PCA+LR	0.54	0.3100	0.2870	0.2749
PCA+K-means	0.28	0.1816	0.2731	0.1933
PCA+NN	0.53	0.3067	0.2955	0.2670

Forth fold:

	acc	recall	precision	f1
kNN	0.64	0.2909	0.3280	0.2728
LR	0.71	0.3583	0.3074	0.3308
Kmeans	0.07	0.0719	0.0999	0.0639
NN	0.62	0.2400	0.2073	0.2219
PCA+kNN	0.62	0.2747	0.2698	0.2586
PCA+LR	0.72	0.375	0.3170	0.3436
PCA+Kmeans	0.19	0.0819	0.1693	0.1049
PCA+NN	0.64	0.2352	0.2046	0.2187

Fifth fold:

	acc	recall	precision	f1
kNN	0.34	0.2437	0.2380	0.2119
LR	0.37	0.2437	0.1546	0.1883
Kmeans	0.2	0.2198	0.1871	0.1553
NN	0.33	0.2162	0.1345	0.1657
PCA+LR	0.38	0.2497	0.1578	0.1925
PCA+Kmeans	0.26	0.2637	0.2505	0.2064

(3) Selection of classifier

Averaging the results of the five-fold cross-validation shows that the PCA+LR algorithm was able to outperform the other models in all four metrics, indicating that the LR algorithm outperformed the supervised k-nearest neighbour, neural network, and the unsupervised k-mean algorithm for this task. The combination of PCA+LR outperformed the LR algorithm using all features, further illustrating the effectiveness of feature selection in this task. Therefore, a combination of

logistic regression and principal component analysis was ultimately selected as the optimal classifier for this experiment.

5 Conclusion

This experiment was conducted to classify the given student data. In the pre-processing data stage, the data is first normalised and then divided into a training set and a test set in a ratio of 4:1. A principal component analysis algorithm performed dimensionality reduction to perform feature selection. The models used include the supervised k-nearest neighbour algorithm, the logistic regression algorithm, the neural network algorithm, and the unsupervised learning model k-means algorithm. The four models were trained using the training set and then evaluated on the test set by calculating four metrics: accuracy, precision, recall and f1. Moreover, experiments were also conducted on the four models using the data after PCA for eight model combinations. The models were evaluated using a five-fold cross-validation approach. The experimental results indicated that the PCA+ combination could achieve optimal results on the test set and was therefore chosen as the optimal classifier.