COMP5009 DATA MINING

Assignment Report

OCTOBER 3, 2024

Student ID: 21678145
Surname: Niu
Given name: Ben

Table of Contents

1. Summary

In this assignment of Data Mining, I tried to solve a real-world data mining problem. I used Google Colab, Python and sklearn to complete this assignment.

The dataset contains 5000 data, and I use them to train 4 models and select the best 2 models to predict 500 new data. The model prediction accuracy achieved 85%.

During this assignment, I studied how to prepare data, classify data and tune models. I learned how to identify and remove irrelevant attributes, handle missing entries and duplicates, and transform data. I learned how to handle imbalanced data and tune model parameters.

2. Methodology

2.1 Data preparation

After downloading the SQLite DB from the website, we can examine all data attributes and identify issues present in the data.

2.1.1 Data overview

The data shape is 5000 rows × 32 columns.

The name and type of columns are as follows:

Table 2-1 Data Types

Column	Type	Column	Type	Column	Type	Column	Type
Index	int64	Buyer	float64	Storage	object	Session	int64
System	float64	Insect	float64	Resource	float64	Guitar	object
Science	float64	Music	object	Writer	float64	Shopping	float64
Method	float64	Guidance	float64	Member	float64	Trainer	int64
People	float64	Power	float64	Cookie	float64	Office	float64
Estate	float64	Knowledge	int64	Virus	float64	Country	float64
Tennis	float64	Owner	int64	Moment	float64	Tension	float64
Problem	float64	Oven	float64	Driver	float64	class	int64

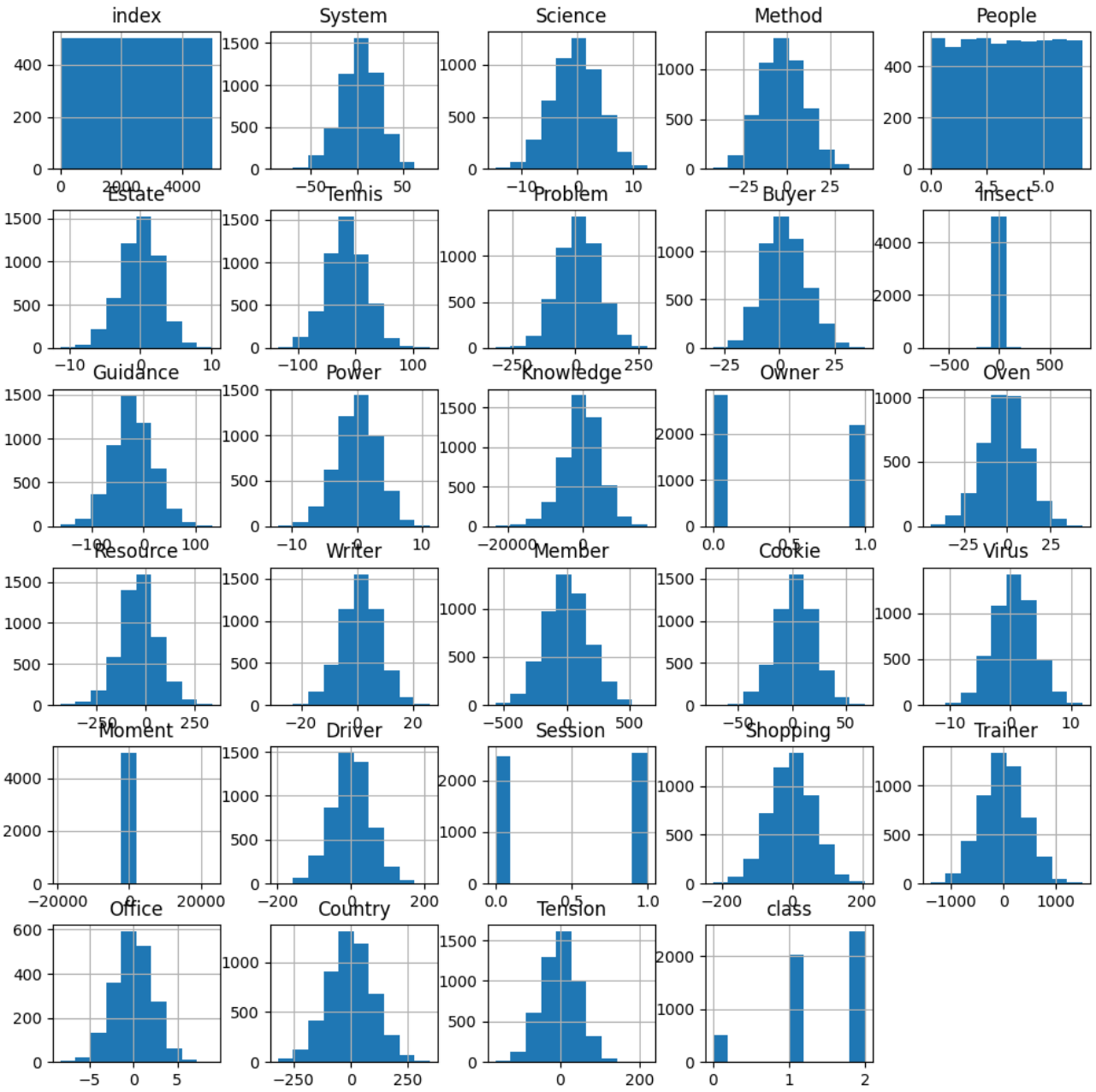

Figure 2-1 Data Histogram

From the histogram, I found that the columns “Insect” and “Moment” distribution show some outlier values. So, check the standard deviation.

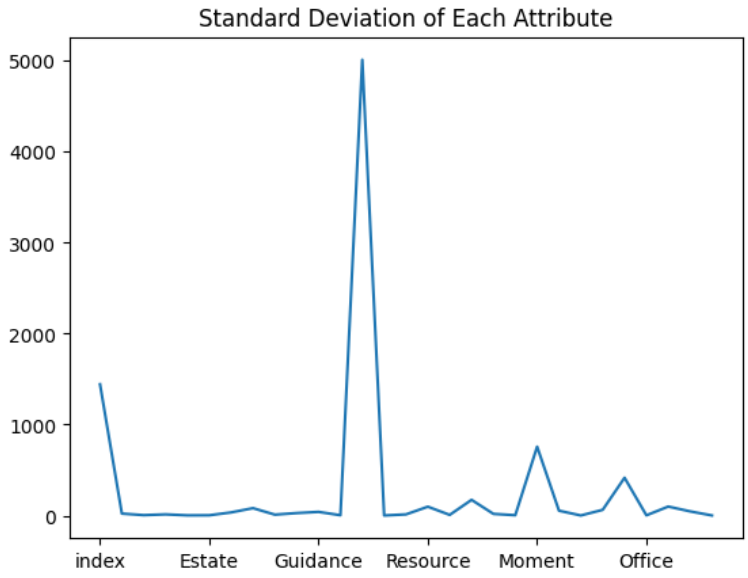

Figure 2-2 Standard Deviation

I filtered all standard deviations more than 100 except “index”.

Table 2-2 Columns with STD > 100

Column	STD
Knowledge	5012.001888066708
Moment	756.8477835397937
Trainer	415.3156156627911
Member	172.2754715353466

I used “RobustScaler” in sklearn to handle the outliers.

The distribution before handling and after handling as follows:

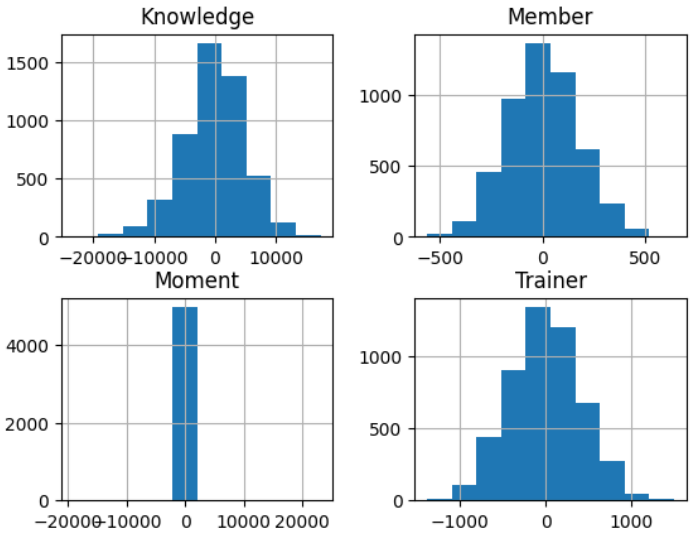

Figure 2-3 Before Handling

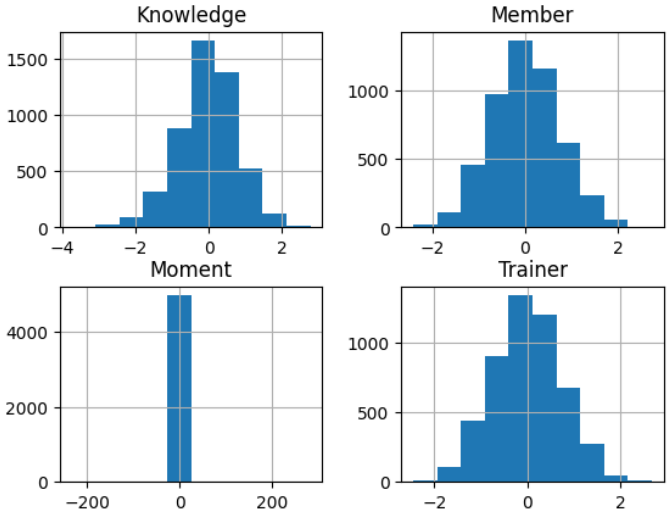

Figure 2-4 After Handling

2.1.2 Identify and remove irrelevant attributes

1. Remove the “index” column: “index” is only an order number to label the data; it is unrelated to the label.
2. Analyse correlations of columns: use “pd.corr” and “sns.heatmap” to monitor the correlations of columns.

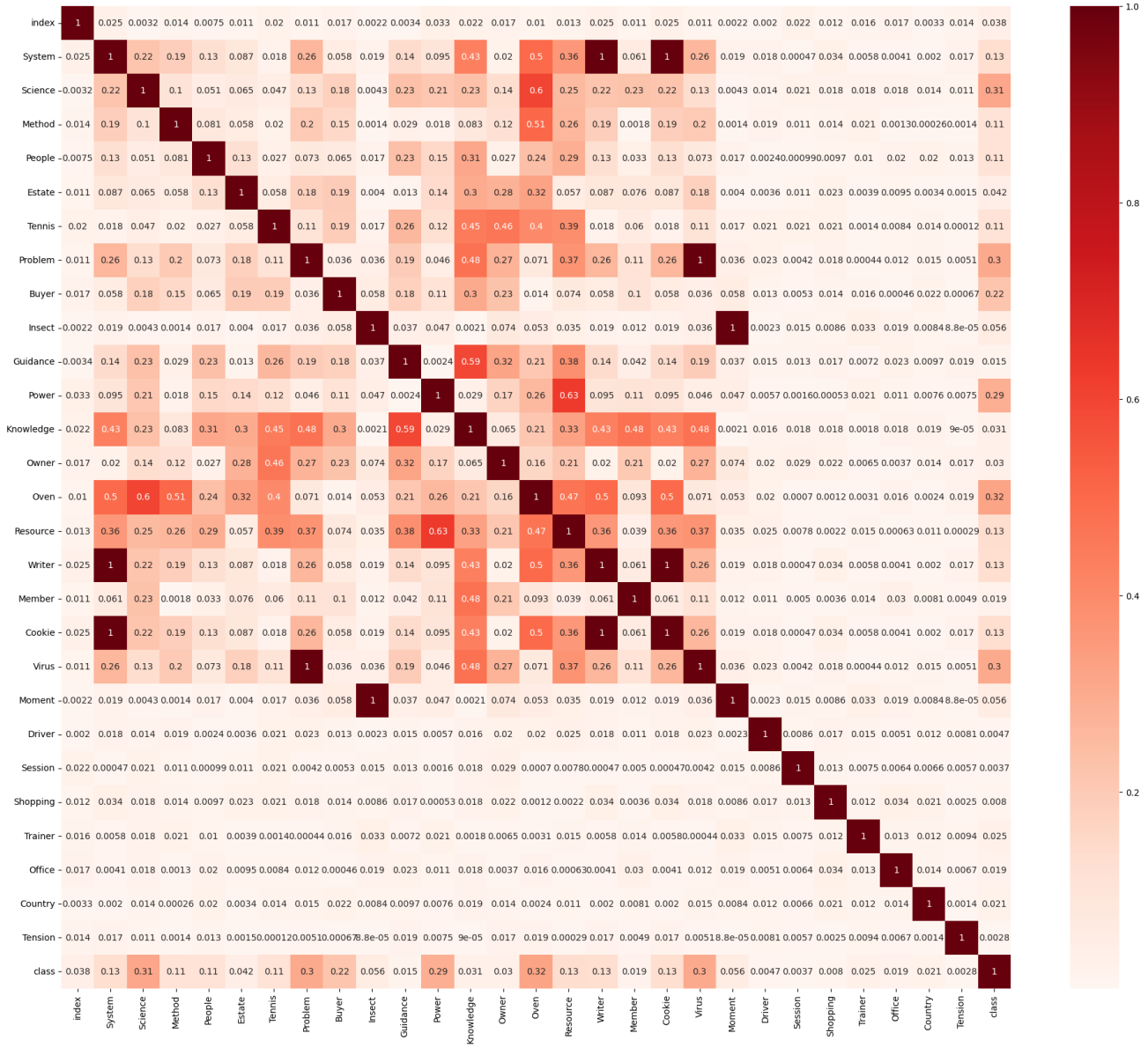

Figure 2-5 Heatmap for Correlations

I found that some columns are highly correlated. Set the correlation high threshold as 0.8 and filter all correlation items. Drop the “Writer”, “Cookie”, “Virus”, and “Moment” columns.

Table 2-3 Column Correlation > 0.8

Column 1	Column 2	Correlation
System	Writer	1.0000000000000004
System	Cookie	1.0
Problem	Virus	1.0000000000000024
Insect	Moment	0.9999999999999984
Writer	Cookie	0.9999999999999972

Continue checking the correlation of all attributes with column “class”, and filter all attributes which have low correlation (set correlation low threshold as 0.1):

Table 2-4 Columns Correlation with class < 0.1

Column 1	Column 2	Correlation
class	index	0.03766008168623011
class	Estate	0.04238407557518169
class	Insect	0.05622949087833261
class	Guidance	0.015080251130552235
class	Knowledge	0.030992414281511914
class	Owner	0.029526606860980856
class	Member	0.018503421901164586
class	Moment	0.056229490878332504
class	Driver	0.0046604714092332595
class	Session	0.0036983105394678556
class	Shopping	0.00801755216832826
class	Trainer	0.024601270464653678
class	Office	0.01856881780794045
class	Country	0.021209070634925237
class	Tension	0.00281650689440731

Drop columns of “Estate”, “Insect”, “Guidance”, “Knowledge”, “Owner”, “Member”, “Moment”, “Driver”, “Session”, “Shopping”, “Trainer”, “Office”, “Country”, “Tension”.

3. Drop the columns whose type equals “object”: “Music”, “Storage”, “Guitar”.

The dataset is reduced to 10 columns.

2.1.3 Detect and handle missing entries

Calculate the missing rate for each column:

Table 2-5 Missing Rate

Column	Missing Rate (%)	Column	Missing Rate (%)	Column	Missing Rate (%)
System	0	Tennis	1.0	Oven	22.36
Science	0	Problem	0	Resource	0
Method	0	Buyer	0		
People	0	Power	0		

1. Drop missing rate greater than 20: “Oven”
2. Fill in the missing data using the mean value for the missing rate less than 5.

The dataset is reduced to 10 columns.

2.1.4 Detect and handle duplicates

1. Check the row duplicated
   There are 50 rows duplicated, and the duplicated indexes are:

42, 91, 111, 508, 682, 763, 872, 983, 1222, 1225, 1462, 1537, 1560, 1816, 1935, 2003, 2118, 2250, 2415, 2474, 2631, 2719, 2740, 2745, 2970, 2974, 3249, 3393, 3551, 3557, 3620, 3681, 3885, 3993, 4024, 4128, 4161, 4238, 4287, 4327, 4443, 4562, 4642, 4771, 4785, 4826, 4928, 4944, 4962, 4986.

2. Check the column duplicated
   There is no duplication.

Now the data shape is 4950 rows x 9 columns.

2.1.5 Select suitable data types for attributes

Check the type of all attributes:

Table 2-6 Attributes and Type

Column	Type	Column	Type	Column	Type
System	float64	People	float64	Buyer	float64
Science	float64	Tennis	float64	Power	float64
Method	float64	Problem	float64	Resource	float64

All the types are “float64”, don’t need more handling.

2.1.6 Perform data transformation

Attributes with different scales or distributions can cause bias in the data mining application.

Table 2-7 Attributes Describe

index	System	Science	Method	People	Tennis	Problem	Buyer	Power	Resource
count	4950	4950	4950	4950	4950	4950	4950	4950	4950
mean	3.670037	-0.20402	-2.20297	3.374596	-15.1987	14.75373	2.4189	0.105462	-35.528
std	21.06388	4.25023	12.61826	1.939023	33.96617	81.05217	9.637839	3.195017	96.88005
min	-86.1559	-14.6682	-42.3718	0.001222	-135.559	-315.53	-30.2006	-12.1974	-435.234
25%	-10.4049	-3.09724	-11.1447	1.711706	-37.5888	-39.648	-4.43419	-1.99157	-98.1736
50%	4.202203	-0.15908	-2.5402	3.376959	-15.1897	15.4177	1.97323	0.183484	-35.9764
75%	18.2044	2.805944	6.456216	5.055354	7.223578	69.10684	8.977011	2.272641	25.33945
max	79.52288	12.53653	45.13045	6.722135	129.5476	284.9823	38.67819	11.2812	338.5063

Do Z-score scaling using “StandardScaler” in sklearn. After transformation, each column data mean is 0 and the standard deviation is 1.

Table 2-8 Attributes Describe after Scaling

index	System	Science	Method	People	Tennis	Problem	Buyer	Power	Resource
count	4950	4950	4950	4950	4950	4950	4950	4950	4950
mean	-2.30E-17	2.30E-17	3.45E-17	1.38E-16	-4.59E-17	0	0	0	-4.59E-17
std	1.000101	1.000101	1.000101	1.000101	1.000101	1.000101	1.000101	1.000101	1.000101
min	-4.26489	-3.4035	-3.18371	-1.7399	-3.54388	-4.07537	-3.38487	-3.85104	-4.1262
25%	-0.66827	-0.68079	-0.7087	-0.85768	-0.65926	-0.67126	-0.71113	-0.65641	-0.6467
50%	0.025267	0.010575	-0.02673	0.001219	0.000266	0.008193	-0.04625	0.024422	-0.00463
75%	0.690083	0.70826	0.686312	0.866894	0.660202	0.670662	0.680523	0.678368	0.62834
max	3.601449	2.997916	3.751563	1.72658	4.261914	3.334345	3.76256	3.498219	3.861187

We should apply the same handling for the test data.

Table 2-9 Attributes Describe for Test Data

index	System	Science	Method	People	Tennis	Problem	Buyer	Power	Resource
count	500	500	500	500	500	500	500	500	500
mean	-0.07047	0.001066	-0.00386	-0.05364	0.042565	0.042216	-0.07322	0.033123	-0.03668
std	1.035409	0.960692	0.987809	1.015752	0.953415	1.007975	0.964691	1.024279	1.064561
min	-3.36434	-2.6629	-2.94146	-1.74053	-2.41562	-3.02083	-2.86147	-3.04302	-4.46288
25%	-0.75266	-0.67892	-0.69182	-1.03642	-0.6007	-0.63384	-0.78428	-0.61892	-0.65987
50%	-0.08314	0.025394	-0.03144	-0.05048	0.058703	0.025674	-0.11451	0.092811	-0.05921
75%	0.612187	0.682995	0.656082	0.824809	0.657751	0.751336	0.522145	0.752678	0.618869
max	3.243257	3.420597	2.845443	1.706404	2.692774	2.897639	2.358577	2.778631	2.998217

2.1.7 Perform other data preparation operations

NA

2.2 Data classification

2.2.1 Class imbalance

Use “train_test_split” in sklearn to split the dataset (size: 4950) into the training dataset (3464, 70%), validation dataset (743, 15%), and testing dataset (743, 15%).

Check the class distribution in the training dataset:

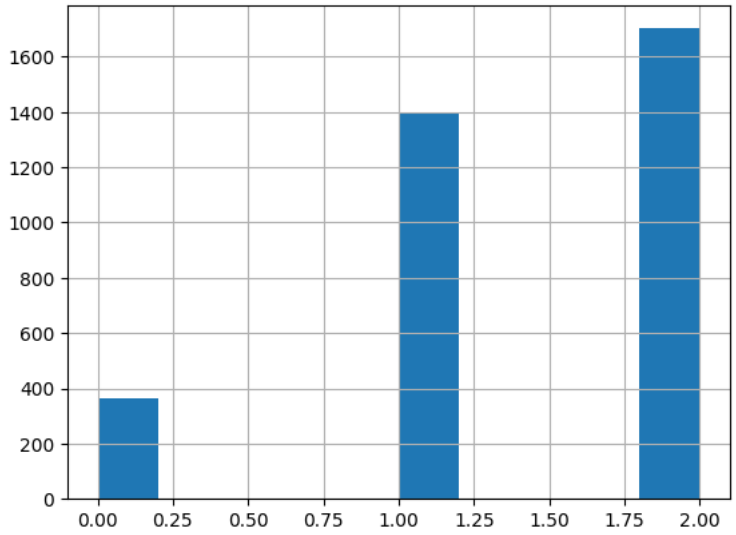

Figure 2-6 Class Distribution in Training Dataset

The dataset is imbalanced. If I use the dataset to train the model directly, I find the predictable label is very imbalanced for class “0”.

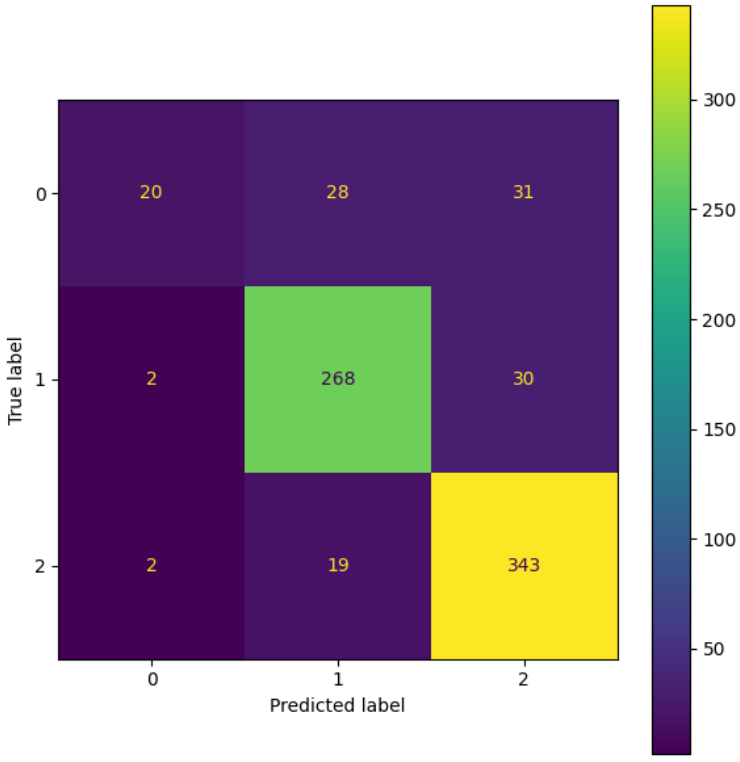

Figure 2-7 Default training data predict

I used “RandomOverSampler” in sklearn to balance the training dataset. Then check the class distribution in the training dataset again: total 5106, 1702 for each class.

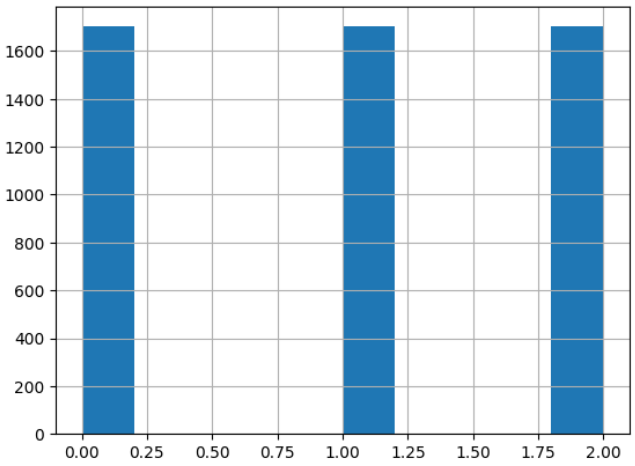

Figure 2-8 Class Distribution in Training Data after Oversampling

Using the oversampled dataset to train the model again, I found that the predictable label is balanced for class “0”.

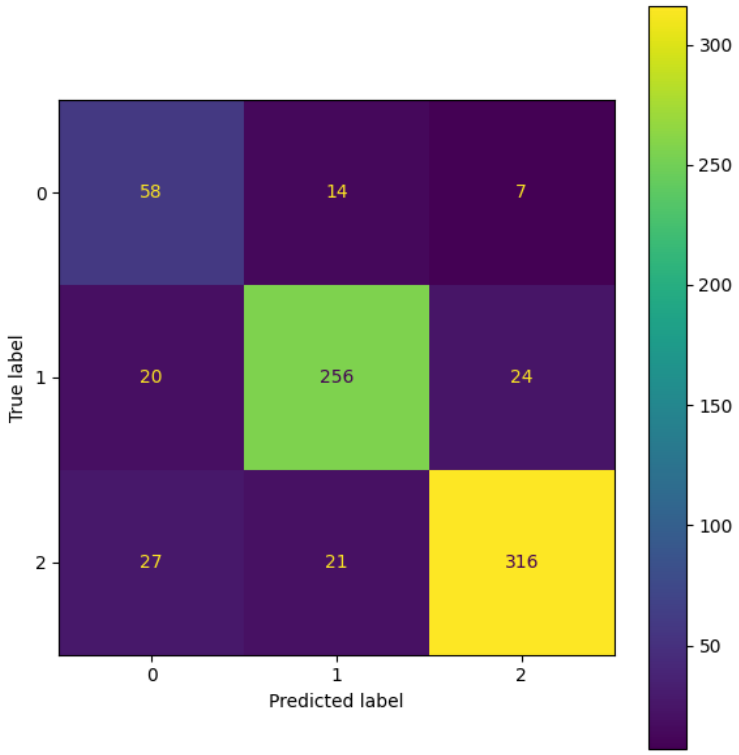

Figure 2-9 After Oversample

2.2.2 Model training and tuning

I split 15% of the data as a validation dataset; I used “PredefinedSplit” as cross-validation (cv) in “GridSearchCV” to validate the model parameters.

Because the validation dataset is imbalanced, I use “precision_score” with “weighted” average parameter as scorer in “GridSearchCV”.

I select 4 models to train and tune, using “GridSearchCV” to select the best parameters.

1. KNN

Table 2-10 KNN Tuning

Parameter	Values	Best Param	Best Score
weights	'uniform','distance'	distance	0.8703840669201928
neighbors	1,3,7,11,17,21,25,30,35,40,45,50,55,60	25	

2. DT

Table 2-11 DT Tuning

Parameter	Values	Best Param	Best Score
criterion	'gini','entropy'	gini	0.782205912087215
min_samples_split	2,3,4,5,7,10,15,20	2	
max_depth	5,7,8,9,10,11,12,13,14,15	7	

3. NB

Table 2-12 NB Tuning

Parameter	Values	Best Param	Best Score
var_smoothing	1e-10, 1e-09, 1e-08, 1e-07	1e-10	0.763941400117287

4. SVM

Table 2-13 SVM Tuning

Parameter	Values	Best Param	Best Score
kernel	'linear', 'poly', 'rbf'	kbf	0.8808395376802616
C	1.0, 2.0, 3.0, 4.0, 5.0, 6.0	4.0	
gamma	'scale', 'auto'	scale	

2.2.3 Model comparison

After getting the best parameters for each model, we must use the validation and testing datasets to score the models.

Table 2-14 Model Param and Comparison

Name	Param	Score	Validation	Test
KNN	{'n_neighbors': 25, 'weights': 'distance'}	0.870384	0.847914	0.847914
DT	{'criterion': 'gini', 'max_depth': 7, 'min_samples_split': 2}	0.782206	0.755047	0.728129
NB	{'var_smoothing': 1e-10}	0.763941	0.705249	0.666218

SVM	{'C': 4.0, 'gamma': 'scale', 'kernel': 'rbf'}	0.88084	0.87214	0.873486

So the best 2 models are SVM and KNN, and their estimated prediction accuracy are 0.873 and 0.848.

2.2.4 Prediction

I use the best 2 models to predict the 500 test samples, write the result out a table “predict” and store it in “Answers.sqlite”.

Table 2-15 Predict Model Accuracy

Predict	Model	Accuracy
Predict1	SVM	0.873
Predict2	KNN	0.848

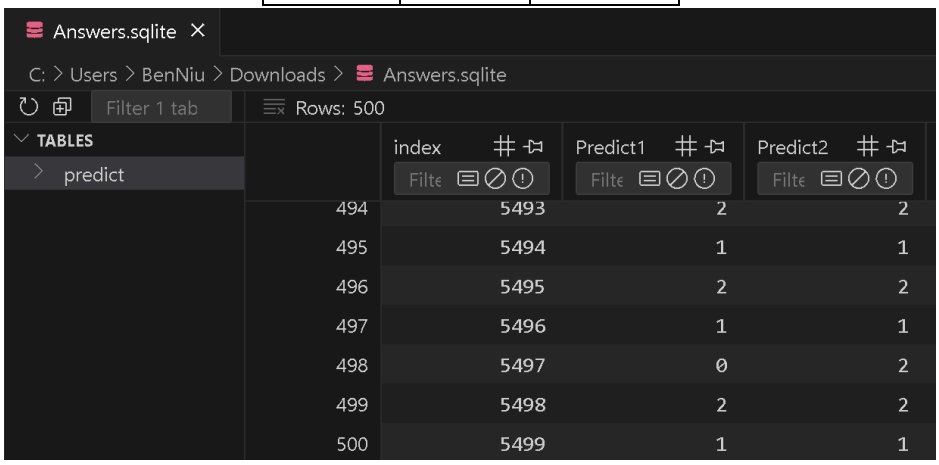

Figure 2-10 Predict Output

2.2.5 Other inventive steps

NA

3. Conclusion

I followed the instructions in the “Assignment.pdf” and practicals to complete the data preparation, classification and prediction tasks.

I tried my best to handle the data's irrelevant attributes, missing entries, duplicates and transformations, and tune the best 2 models with an accuracy of 85%. Then, the models predict 500 new data and store them in an SQLite database.

4. References

2. over-sampling — Version 0.11.0. (n.d.). Welcome to imbalanced-learn documentation!. https://imbalanced-learn.org/stable/over_sampling.html

1.4. Support vector machines. (n.d.). scikit-learn. Retrieved October 4, 2024, from https://scikit-learn.org/stable/modules/svm.html

GridSearchCV. (n.d.). scikit-learn. Retrieved October 4, 2024, from https://scikit-learn.org/dev/modules/generated/sklearn.model_selection.GridSearchCV.html

Precision_score. (n.d.). scikit-learn. Retrieved October 4, 2024, from https://scikit-learn.org/dev/modules/generated/sklearn.metrics.precision_score.html#sklearn.metrics.precisio

5. Appendices

NA