CSI 5155 Assignment1

Yu Sun(8472921)

October 2020

1 Question 1

1.1 Building Models

In this question, I constructed four basic classifiers: Decision Tree, Naive Bayesian, Support Vector Machine(Linear), and K-nearest neighbor. I used all 20 features to predict whether a client will subscribe a term deposit, with One-Hot encoding for category features. Besides, I split the dataset with 67% for training and 33% for testing.

For Decision Tree model, I set the max_depth=6 and $min\_samples\_leaf$=100 in order to prevent the over-fitting problem. The tree model is visualised below, and the different color represents the different classes.

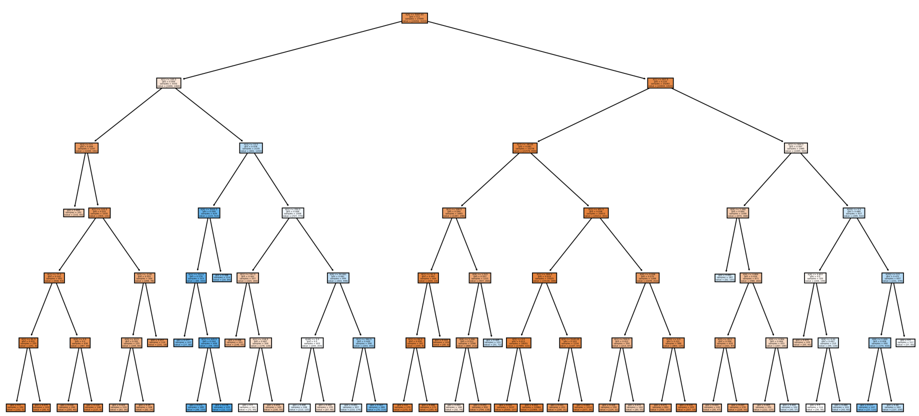

Figure 1: Decision Tree Visualisation

For Naive Bayesian model, I used the GaussianNB. The visualisation of decision boundary and the means for features of the different classes are below:

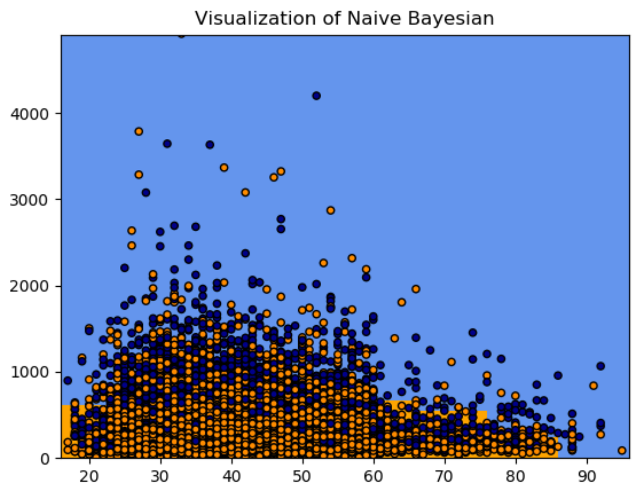

Figure 2: Naive Bayesian Visualisation

Table 1: Naive Bayesian Features' mean and Std

	Mean		Std	
	Class0('no')	Class1('yes')	Class0('no')	Class1('yes')
age	39.904	40.7319	190.7738	190.7738
duration	220.9005	550.1424	160344.1119	160344.1119
campaign	2.6514	2.0342	2.539	2.539
pdays	984.7722	792.2491	162582.9213	162582.9213
previous	0.1315	0.4905	0.7385	0.7385
emp.var.rate	0.2524	-1.2361	2.6259	2.6259
cons.price.idx	93.6052	93.3581	0.4599	0.4599
cons.conf.idx	-40.5995	-39.8424	37.7931	37.7931
euribor3m	3.8138	2.1219	3.0294	3.0294
nr.employed	5176.2535	5094.8012	7731.3527	7731.3527
job_admin.	0.2493	0.2952	0.2081	0.2081
job_blue-collar	0.2373	0.1412	0.1213	0.1213
job_entrepreneur	0.0352	0.0264	0.0258	0.0258
job_housemaid	0.0267	0.0222	0.0218	0.0218
job_management	0.0703	0.0709	0.0659	0.0659
job_retired	0.035	0.087	0.0795	0.0795
job_self-employed	0.0355	0.0306	0.0297	0.0297
job_services	0.1002	0.0715	0.0665	0.0665
job_student	0.0176	0.0593	0.0558	0.0558
job_technician	0.1616	0.1576	0.1328	0.1328
job_unemployed	0.0232	0.0309	0.03	0.03

job_unknown	0.0082	0.0071	0.0071	0.0071
marital_divorced	0.1124	0.1031	0.0926	0.0926
marital_married	0.61	0.545	0.248	0.248
marital_single	0.2758	0.3497	0.2275	0.2275
marital_unknown	0.0018	0.0023	0.0023	0.0023
education_basic.4y	0.1028	0.0925	0.084	0.084
education_basic.6y	0.0582	0.0409	0.0393	0.0393
education_basic.9y	0.1501	0.0986	0.089	0.089
education_high.school	0.2346	0.2204	0.1719	0.1719
education_illiterate	0.0003	0.0006	0.0007	0.0007
education_professional.course	0.1261	0.1273	0.1112	0.1112
education_university.degree	0.2863	0.369	0.2329	0.2329
education_unknown	0.0415	0.0506	0.0481	0.0481
default_no	0.7776	0.9069	0.0845	0.0845
default_unknown	0.2224	0.0931	0.0845	0.0845
default_yes	0.0001	0.0	0.0001	0.0001
housing_no	0.453	0.4415	0.2466	0.2466
housing_unknown	0.0249	0.0242	0.0237	0.0237
housing_yes	0.5221	0.5343	0.2489	0.2489
loan_no	0.8233	0.8205	0.1473	0.1473
loan_unknown	0.0249	0.0242	0.0237	0.0237
loan_yes	0.1518	0.1553	0.1313	0.1313
contact_cellular	0.611	0.8379	0.1359	0.1359
contact_telephone	0.389	0.1621	0.1359	0.1359
month_apr	0.0567	0.1144	0.1014	0.1014
month_aug	0.1517	0.1395	0.1201	0.1201
month_dec	0.0028	0.0184	0.0181	0.0181
month_jul	0.1804	0.1447	0.1238	0.1238
month_jun	0.1303	0.1231	0.108	0.108
month_mar	0.008	0.0606	0.057	0.057
month_may	0.3498	0.1814	0.1486	0.1486
month_nov	0.1006	0.0906	0.0824	0.0824
month_oct	0.0111	0.0722	0.067	0.067
month_sep	0.0086	0.0551	0.0521	0.0521
day_of_week_fri	0.1928	0.1811	0.1484	0.1484
day_of_week_mon	0.2074	0.1824	0.1492	0.1492
day_of_week_thu	0.2066	0.2272	0.1756	0.1756
day_of_week_tue	0.1943	0.2056	0.1634	0.1634
day_of_week_wed	0.1989	0.2037	0.1623	0.1623
poutcome_failure	0.0995	0.1289	0.1124	0.1124
poutcome_nonexistent	0.8879	0.68	0.2177	0.2177
poutcome_success	0.0126	0.1911	0.1547	0.1547

For Support Vector Machine, I used SVC with Linear kernel and C=20, The visualisation of decision boundary and the weights of features are below:

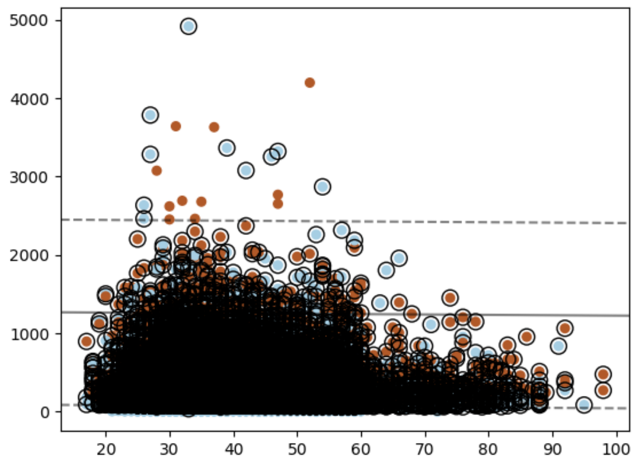

Figure 3: Support Vector Machine Visualisation

Table 2: SVC Weights of features

Feature	Coef	Feature	Coef
age	-0.0125	education_university.degree	-0.0102
duration	0.0272	education_unknown	-0.0032
campaign	-0.0154	default_no	0.0061
pdays	-0.0	default_unknown	-0.0061
previous	-0.0045	default_yes	0.0
emp.var.rate	-0.0741	housing_no	0.0201
cons.price.idx	-0.0227	housing_unknown	-0.0024
cons.conf.idx	0.005	housing_yes	-0.0177
euribor3m	0.0041	loan_no	0.0051
nr.employed	0.0044	loan_unknown	-0.0024
job_admin.	-0.0103	loan_yes	-0.0027
job_blue-collar	0.0007	contact_cellular	-0.0065
job_entrepreneur	-0.006	contact_telephone	0.0065
job_housemaid	0.0358	month_apr	0.0
job_management	-0.0005	month_aug	-0.0162
job_retired	-0.0054	month_dec	0.0
job_self-employed	0.0	month_jul	-0.0087
job_services	0.0097	month_jun	-0.0071
job_student	-0.0125	month_mar	0.0036
job_technician	-0.0114	month_may	0.0

job_unemployed	0.0	month_nov	-0.0021
job_unknown	0.0	month_oct	0.031
marital_divorced	-0.0017	month_sep	-0.0005
marital_married	0.0255	day_of_week_fri	-0.014
marital_single	-0.0238	day_of_week_mon	-0.01
marital_unknown	0.0	day_of_week_thu	-0.0045
education_basic.4y	-0.0072	day_of_week_tue	0.0289
education_basic.6y	-0.0032	day_of_week_wed	-0.0004
education_basic.9y	0.0003	poutcome_failure	-0.0045
education_high.school	0.0272	poutcome_nonexistent	0.0045
education_illiterate	0.0	poutcome_success	0.0
education_unknown	-0.0032		

For KNN, I used *n_neighbors*=3, The visualisation of decision boundary is below:

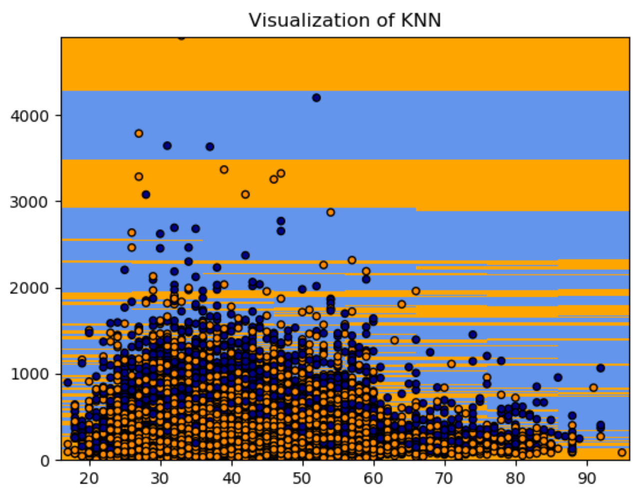

Figure 4: KNN Visualisation

1.2 Confusion Matrices

For the four models, I calculate the confusion matrices, recall rate, precision and accuracy after predicting with test data. The results are summarised below:

Table 3: Confusion Matrix of Decision Tree

		Prediction class		
		p	**n**	**total**
actual class	**p′**	TP 727	FN 810	P′
	n′	FP 393	TN 11663	N′
	total	P	N	

Table 4: Confusion Matrix of Naive Bayesian

		Prediction class		
		p	**n**	**total**
actual class	**p′**	TP 807	FN 730	P′
	n′	FP 1080	TN 10976	N′
	total	P	N	

Table 5: Confusion Matrix of SVM

		Prediction class p	Prediction class n	total
actual class	$\mathbf{p'}$	TP 468	FN 1069	P'
	$\mathbf{n'}$	FP 257	TN 11799	N'
	total	P	N	

Table 6: Confusion Matrix of KNN

		Prediction class p	Prediction class n	total
actual class	$\mathbf{p'}$	TP 717	FN 820	P'
	$\mathbf{n'}$	FP 576	TN 11480	N'
	total	P	N	

Table 7: Recall, Precision and Accuracy of the models

Model	Recall	Precision	Accuracy
Decision Tree	0.5185	0.6427	0.9129
Naive Bayesian	0.5250	0.4276	0.8668
SVM	0.3044	0.6455	0.9024
KNN	0.4664	0.5545	0.8973

1.3 ROC Curves

For each model, the ROC Curve list below:

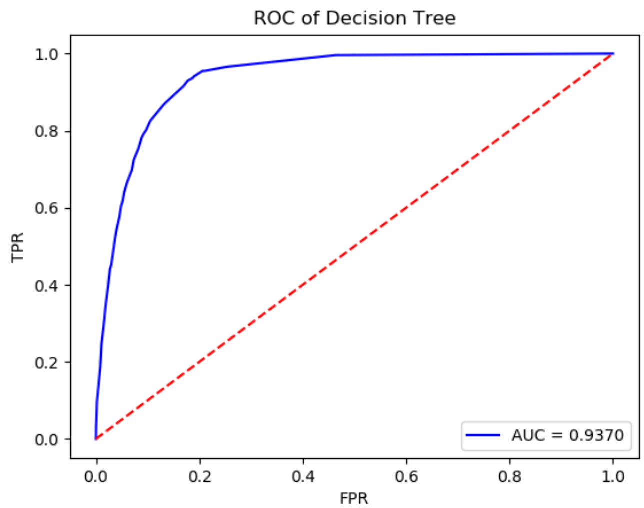

Figure 5: Decision Tree ROC

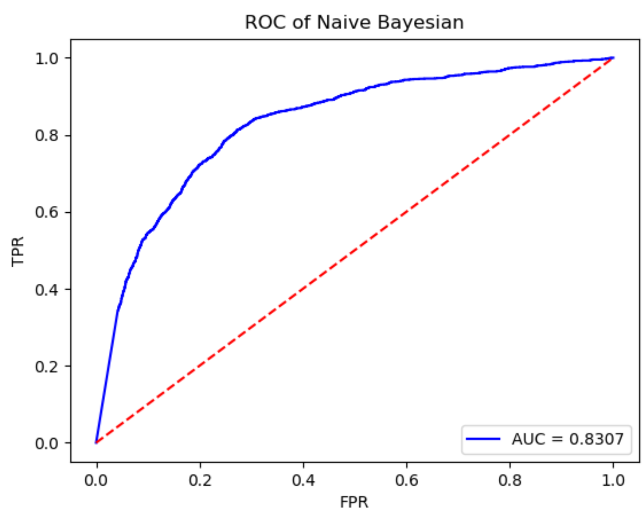

Figure 6: Naive Bayesian ROC

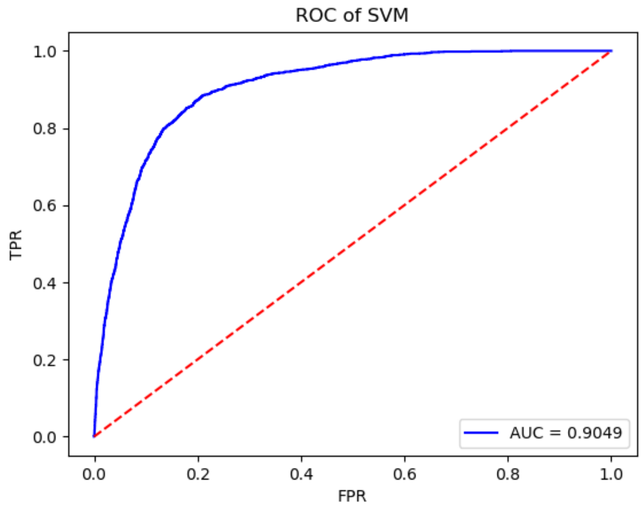

Figure 7: SVM ROC

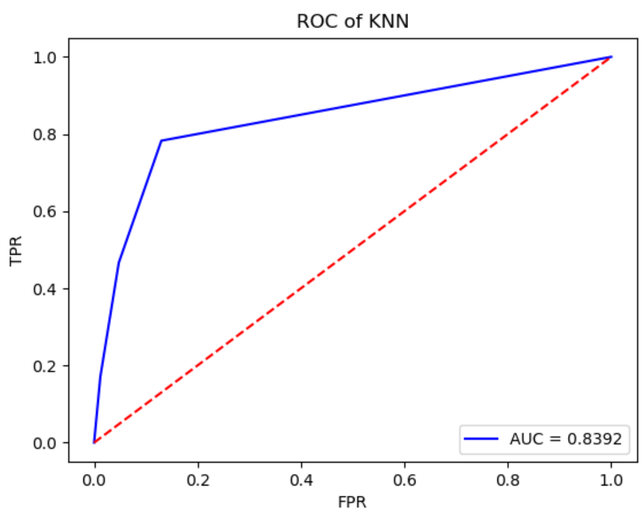

Figure 8: KNN ROC

1.4 Discussion

I noticed that the dataset has some imbalance issue with class ”yes” only accounting 10%. For this type of problem, the accuracy is not a good matrix to evaluate the model, because the learner intends to predict the class of majority samples to get high accuracy score. And we usually care more about whether the minority samples can be predicted correctly. To deal with this issue, the recall and precision are involved. Recall means how many samples' classes are predicted correctly in all samples, and precision means how many samples' classes are predicted correctly in the samples which have the same predicted class. Besides, we can calculate the AUC value through ROC curve, which also indicates the performance of the classifier. Higher AUC value means the model is better. So after constructing the four models and comparing the above matrices of each model, I found that Decision Tree is the best one with highest AUC value. Although the precision of SVM is high, the recall is extremely low.

2 Question 2

For this question, the target is column "poutcome" and features are other 19 columns except "y". Similarly, I used One-Hot encoding for category features.

2.1 Model

I build the tree model with the *max_depth*=5 and *min_samples_leaf*=100. Then I calculate the confusion matrix for each class.

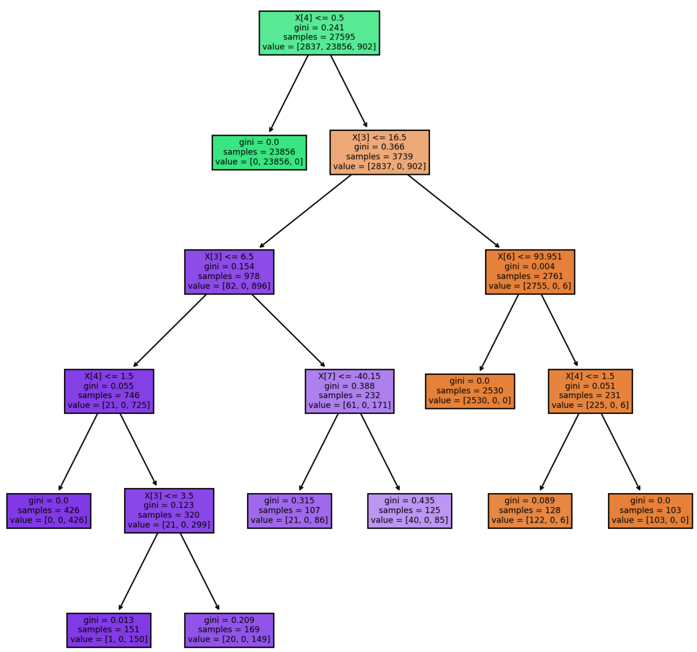

Figure 9: Decision Tree Visualisation

Table 8: Confusion Matrix of class ”failure”

		Prediction class p	Prediction class n	total
actual class	$\mathbf{p'}$	TP 1370	FN 45	P'
	$\mathbf{n'}$	FP 6	TN 12172	N'
	total	P	N	

Table 9: Confusion Matrix of class ”nonexistent”

		Prediction class p	Prediction class n	total
actual class	$\mathbf{p'}$	TP 1886	FN 0	P'
	$\mathbf{n'}$	FP 0	TN 11707	N'
	total	P	N	

Table 10: Confusion Matrix of class "success"

		Prediction class p	Prediction class n	total
actual class	p′	TP 465	FN 6	P′
	n′	FP 45	TN 13077	N′
	total	P	N	

Table 11: Recall, Precision for each class

Class	Recall	Precision
Failure	0.9681	0.9956
Nonexistent	1.0	1.0
Success	0.9872	0.9117

2.2 ROC Curve

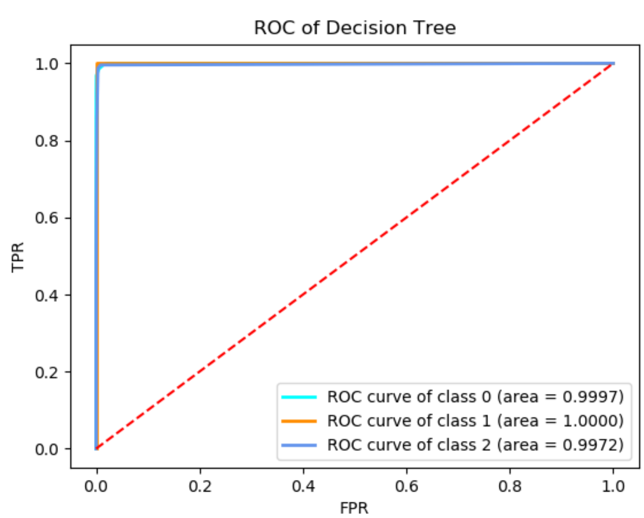

Figure 10: Decision Tree Visualisation

2.3 Calculating AUC

One-vs-One ROC AUC scores:

- 0.998974 (macro)
- 0.999872 (weighted by prevalence)

2.4 Discussion

Different from the binary classification in Question 1, we need to analyse each class in detail for multi-class problem. Because some classifiers may get the good result for whole samples, but in some class's sample which we are concerned more may get the bad behaviour. After comparing the output of each class, I noticed that the learner with high AUC value for all classes, and it is best for class "Nonexistent" with 0 error in testset. The recall of "success" is higher that "failure", while the precision is a little bit lower.

3 Appendix

The code is available on GitHub:`https://github.com/YuSun09/CSI5155`