DMML Assignment 1

Paarth Iyer - MCS202218

February 12, 2023

Question 1

Preprocessing of data

For `bank_X`, we do the following:

- Drop `'duration'` : As this data won't be available till the call is complete, it won't be beneficial for prediction. It also has a strong effect on the result
- Drop `'day_of_week'` : When we check the distribution of these against the result, there is little to no difference in the classes
- Label encode `'housing'` and `'loan'` using this rule : 'yes'=2, 'unknown'=1,'no'=0
- Label encode `'education'` and `'job'`
- Onehot encode the remaining categorical data

Now all the fields are numerical

Decision Tree

The model is made with these parameters : `max_depth=15,min_samples_split = 7,criterion='gini'`

These were chosen by checking the performance on multiple different parameters and random seeds, and this particular choice seems to perform marginally better.

Result : (using np seed 100)

.	precision	recall	f1-score
no	0.92	0.96	0.94
yes	0.50	0.31	0.38
accuracy	.	.	0.89

We get an accuracy of about 89%. Where as the recall for 'yes' is 31%

Naive Bayes

`GaussianNB` naive bayes classifier from sklearn turns out to perform the best of the other available models. This may be explained by some of the more influential numerical fields being approximately normally distributed.

Result : (using np seed 100)

.	precision	recall	f1-score
no	0.93	0.90	0.91
yes	0.35	0.45	0.39
accuracy	.	.	0.85

Here, the recall of 'yes' gets slightly better, but both accuracy and recall of 'no' drops. Also, the precision of 'yes' is lower, making the model more unpredictable in terms of responding positively.

Comparison

-Training times are too small to be compared

The DT gives us a higher overall accuracy and precision for the results, meaning the results may be more consistent. But the recall of 'yes' is lower than that of NB. This tells us that NB has a higher probability of predicting a 'yes' correctly but overall the NB classifier is not as accurate.

Question 2

Preprocessing of data

First, we create `bolly_y` using list comprehension and using the rule : `hit` if *revenue* > *budget* else `flop`

For `bolly_X`, we do the following:

- Drop `'Movie Names'` : These won't have any major effect as just about all of them are unique.
- Label encode `'Whether Remake', 'Whether Franchise','New Actor', 'New Director', 'New Music Director'` using 'no'=0 and 'yes'=1
- Label encode `'Release Period'` using 'Normal'=0 and 'Holiday'=1
- Onehot encode the rest of the columns (Lead Star, Director and Music Director)
- For Lead star and Music Director, we drop the Onehot columns which have less than 3 entries
- For Director, we do the same but for 2 entries.
  We do this as the people who have worked in very few number of movies won't have much of an effect in the other movies, where as the people who have worked a lot may turn up in a movie giving a potential splitting point. The 2 and 3 are selected based on the number of people and their distribution in the categories.

Decision Tree

The model is made with these parameters : `max_depth=5,min_samples_leaf = 3,criterion='gini'`
Going higher with the max depth results in lower recall. Could be a result of overfitting.

Result : (using np seed 100)

.	precision	recall	f1-score
flop	0.80	0.88	0.84
hit	0.95	0.92	0.94
accuracy	.	.	0.91

Naive Bayes

`GaussianNB` from sklearn.naive_bayes is used here

Result : (using np seed 100)

.	precision	recall	f1-score
flop	0.82	0.66	0.73
hit	0.88	0.95	0.91
accuracy	.	.	0.87

Comparison

-Training times are too small to be compared

Both models seem to give us pretty high recall for 'hit'. In terms of overall accuracy, DT seems slightly better than NB. But for the recall of 'flop', NB performs much worse than DT. This tells us that using NB to predict flops may not give us the best results.

Overall, DT seems much better in performance. Even though NB has higher recall for predicting hits, the higher overall accuracy of DT makes up for it.