Milestone#2

Task 4: Models comparison

Table 1 : model comparison results

Classifiers	Accuracy (%)	Training time(s)	Training Memory used (MB)	Testing time(s)	Testing Memory used (MB)	Model size (MB) (.pkl format)	Developer
XGBoosting	76.88	0.13	0.40	**0.01**	~ 0	0.20	Frong
LinearSVM	77.39	**0.03**	0.70	**0.01**	~ 0	**0.01**	Punch
Random Forest	76.88	0.2	1.68	0.02	~ 0	2.67	
GaussianNaive Bayes	73.4	0.043	**0.031**	0.033	0.211	**0.01**	Wendy
Linear Logistic Regression	**77.9**	0.050	0.242	0.050	0.020	**0.01**	
SMOTEENN with StandardScaler and RidgeClassifier	74.1	0.33	1.72	0.02	~ 0	2.89	Jincheng Zhang

Experiment dataset : German dataset, contains a total of 995 instances and 20 features.

Metrics :

1. **Prediction accuracy :** Logistic regression classifier achieved the highest accuracy as 77.9%, followed by LinearSVM (77.39%). XGBoosting and Random forest achieved the same accuracy as 76.88 % . The worst is the Gaussian Naive Bayes classifier which achieved 73.4% of accuracy.

2. **Training cost (Training time & memory usage) :** Linear SVM achieved the lowest training time as 0.031 seconds, followed by GaussianNaiveBayes (0.043 seconds) and Logistic Regression (0.050 seconds). The worst is the SMOTEENN classification which achieved 0.33 second. In terms of memory usage, The best one is GaussianNaiveBayes which achieved 0.031MB likewise the worst one is SMOTEENN at 1.72MB.

3. **Inference cost (Testing training time & memory usage) :** LinearSVM,GaussianNaiveBayes and logistic regression consume a small amount of memory on testset ,accounting for 0.01MB respectively in comparison to other classifier models. In terms of training time, XGBoosting and LinearSVM have the shortest training time at 0.01second each. However, the most memory usage classifier out of six on the testset is Gaussian Naive Bayes and the classifier which takes the longest time to execute the testset is Logistic Regression.

4. **Model size :** LinearSVM,GaussianNaiveBayes and logistic regression have a minimum with model size around 0.01MB as .pkl format file. The maximum model size is the ridge classifier as 2.89MB.

Summary
The best model : Linear Logistic Regression
Selecting the best model, we first prioritize the model accuracy. The Logistic Regression model has achieved a considerable accuracy of 77.9% as compared to other classifiers' accuracy score within our analysis. Moreover, its' small-sized can achieve the highest accuracy score which is a reasonable choice.

Task 5: Measure the defined metrics (in Task#2)

"It is worse to class a customer as good when they are bad (false positive), than it is to class a customer as bad when they are good (false negative)."

According to the phase above, we decided to give our interest to false positive values. So to improve the prediction performance (1st goal) and reduce the prediction bias (2nd goal), sampling techniques such as oversampling and undersampling were applied to the dataset in our work.

1st goal -> Improve the prediction performance

Table 2 : model performance comparison after applied sampling techniques

Model	Sampling method	Accuracy (%)	Precision (%)	**Recall (%)**	**F1-score (%)**
XGBoosting	Under-sampling	67.84	71.74	67.84	68.99
	Over-sampling	73.37	73.77	73.37	73.55
	Non-sampling	**76.88**	**75.97**	**76.88**	**76.21**
LinearSVM	Under-sampling	72.36	75.47	72.36	73.27
	Over-sampling	72.86	74.29	75.38	73.93
	Non-sampling	**77.39**	**76.72**	**77.39**	**76.95**
Random Forest	Under-sampling	69.35	74.25	69.35	70.57
	Over-sampling	75.38	74.29	75.38	74.58
	Non-sampling	**76.88**	**75.67**	**76.88**	**74.85**
KNNs	Under-sampling	72.4	72.8	72.4	72.6
	Over-sampling	**76.4**	**80.6**	**76.4**	**77.3**
	Non-sampling	75.9	74.4	75.9	73.5

Model	Sampling method	Accuracy (%)	Precision (%)	**Recall (%)**	**F1-score (%)**
SMOTEENN with StandardScaler and RidgeClassifier	**Under-sampling**	**74.16**	**75.12**	**77.21**	**76.95**
	Over-sampling	75.27	73.98	75.44	73.94
	Non-sampling	75.98	73.27	74.29	75.66

2nd goal -> Reduce the prediction bias (in term of imbalance class)

Table 3 : model prediction bias comparison after applied sampling techniques

Model	Sampling method	**No. false positive**	Precision (%)	Recall (&)	**Specificity**
XGBoosting	Under-sampling	28	71.74	67.84	0.64
	Over-sampling	**21**	**73.77**	**73.37**	**0.58**
	Non-sampling	25	75.97	76.88	0.52
LinearSVM	Under-sampling	18	75.47	72.36	0.69
	Over-sampling	**14**	**74.29**	**75.38**	**0.76**
	Non-sampling	26	76.72	77.39	0.55
Random Forest	**Under-sampling**	**17**	**74.25**	**69.35**	**0.71**
	Over-sampling	30	74.29	75.38	0.49
	Non-sampling	35	75.67	76.88	0.40
KNNs	Under-sampling	32	72.8	72.4	0.80
	Over-sampling	**26**	**80.6**	**76.4**	**0.62**
	Non-sampling	37	74.4	75.9	0.21
SMOTEENN with StandardScale r and RidgeClassifier	Under-sampling	30	71.97	68.57	0.57
	Over-sampling	27	73.85	75.36	0.61
	Non-sampling	**25**	**73.69**	**76.45**	**0.54**

Summary

In terms of models' performance, the sampling techniques are mostly not effective on the models, except for two models such as KNN and RidgeClassifier.So we can conclude that sampling techniques might not be the suitable choice to improve the performance.

On the other hand, applying the sampling techniques, especially over-sampling, can reduce the prediction bias. According to table 3, the models using the over-sampling method not only can achieve lower false positive instances, but also provide higher specificity value than the non-sampling version (except for RidgeClassifier). Having lower false positive instances and higher specificity could help to achieve the business goal of the dataset.