Milestone 2 Report

Task 4: Model Comparison

Table 1: Model comparison results

Model	Acc. (%)	Precision (%)	Training time (s)	Training Memory used (MB)	Testing time (S)	Testing Memory used (MB)	Model size (MB in. pkl format)	Implementer
Random Forest	85.2493	84.8882	7.50	170.73	0.84	170.73	149.63	Rew
LightGBM	87.4910	87.0745	0.59	1.98	0.08	0.43	1.006	Rew
SVM	84.6760	83.8554	16.85	15.98	12.21	0.30	0.924	B
MLP	84.4406	83.6049	40.95	1.34	0.01	0.0	0.049	B
Logistic Regression	82.3012	81.0242	0.05	2.51	0.00	0.0	0.002	Joey
K-NN	83.1201	82.6049	0.11	2.31	4.20	0.48	4.767	Joey

This experiment results in adult dataset run on Laptop CPU i7-770HQ RAM 24 GB GPU GTX 1060 (Laptop).
Performance comparison:

1. **Model performance**

In this part, LightGBM achieved the high accuracy and precision as 87.4910 and 87.0745, followed by Randon forest, MLP, SVM, K-NN and Logistic Regression.

2. **Training cost (Training runtime and memory usage)**

Logistic Regression is the best model in this section, with a training time (0.05 s) less than the MLP model (40.95 s), despite the MLP requiring less memory (1.34 MB) than Logistic Regression (2.51 MB).

3. **Inference cost (Testing runtime and memory usage)**

Logistic Regression is the top model in this section, with testing time (0 s) and memory usage (0 MB) allowing it to outperform other models.

4. **Model size**

When compared to other models, Logistic Regression is the lightest-weighted model, and the worst model in this section is the Random Forest model, which has a size of 149.63 MB while other models have a size of less than 5 MB.

Summary comparison

The best model is LightGBM because when determining the best model, we choose to prioritize performance. The LightGBM model outperforms the others, but it comes at a cost in terms of training and

inference. Logistic Regression and MLP have potential in this area, however their performance is roughly 3-5% lower than LightGBM.

Task 5: Measure the defined metrics (In Task 2)

For this part to improve overall performance, we decided to use F1-score and precision. Because we need to know who can earn more than 50K per year (True Positive).

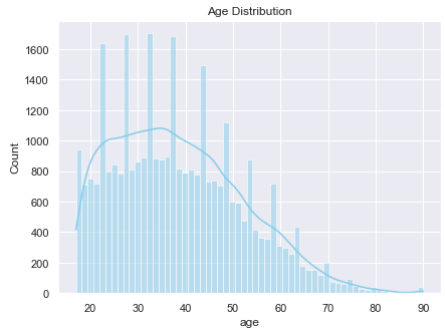
Goal -> Improve the model performance

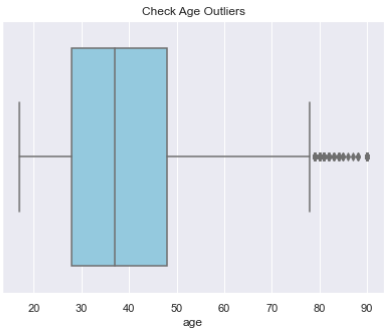
Performance metrics

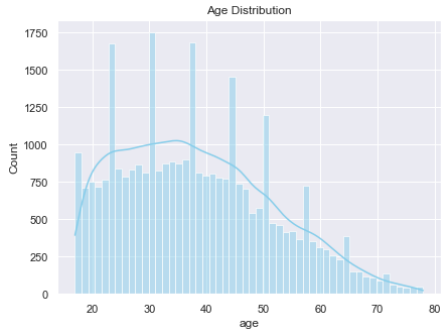
F1-Score: The model can classification between two classes well.

Precision: This score for model to focus predict in class 1.

Based on the performance of the baseline model, we chose to look over the dataset and discovered that age feature contains a high number of outliers. As a result, we remove it and retrain the model to know the difference between remove and do not remove.







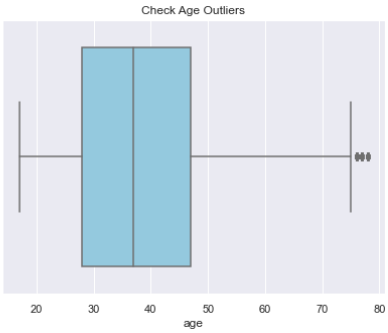


Table 2: Performance after RMO

Model		Accuracy	Recall	Precision	F1-Score
Random Forest	Baseline	85.2493	85.2493	84.7267	84.8882
	RMO	85.3372	85.3372	84.7942	84.9421
LightGBM	Baseline	87.4910	87.4910	87.0745	87.1607
	RMO	87.0339	87.0339	86.5608	86.6201
SVM	Baseline	84.6760	84.6760	83.8554	83.7191
	RMO	85.0545	85.0545	84.3385	84.0741
MLP	Baseline	84.4406	84.4406	83.6049	83.6365
	RMO	84.8240	84.8240	84.3486	84.5138
Logistic Regression	Baseline	82.3012	82.3012	81.0242	80.7996
	RMO	82.7189	82.7189	81.6092	81.1802
K-NN	Baseline	83.1201	83.1201	82.6049	82.8066
	RMO	83.7767	83.7767	83.2237	83.4188

As a result, deleting outliners from age feature helps enhance model performance except LightGBM model is not improved and is worse than baseline. However, when compared to other models, it was a good trade-off. The following stage will be to utilize sampling methods to try to improve the model based on datasets that already eliminate outliners.

Applied sampling methods to improve models

In this section, we used oversampling and undersampling in a variety of approaches before transforming the data with a standard scaler and training the models.

Table 3: oversampling results

Model	Method	Accuracy (%)	Recall (%)	Precision (%)	F1-Score (%)
Random Forest	RMO	85.3372	85.3372	84.7942	84.9421
	SMOTE	84.1852	84.1852	84.5439	84.3437
	RandomOver-sampler	83.9757	83.9757	84.2619	84.1055
	ADASYN	83.8919	83.8919	84.3010	84.0708
	BorderlineSMOTE	85.7876	85.7876	86.3587	86.0153
LightGBM	RMO	87.0339	87.0339	86.5608	86.6201
	SMOTE	85.7876	85.7876	86.3587	86.0153
	RandomOver-sampler	84.3004	84.3004	86.7840	84.9849
	ADASYN	85.0021	85.0021	85.9631	85.3531
	BorderlineSMOTE	84.6774	84.6774	85.8054	85.0773
SVM	RMO	85.0545	85.0545	84.3385	84.0741
	SMOTE	79.4407	79.4407	84.7325	80.6955
	RandomOver-sampler	79.3255	79.3255	85.2725	80.6464
	ADASYN	76.3720	76.3720	84.5020	78.0234
	BorderlineSMOTE	76.5815	76.5815	84.5015	78.2080

MLP	RMO	84.8240	84.8240	84.3486	84.5138
	SMOTE	80.2053	80.2053	84.4724	81.3066
	RandomOver-sampler	79.7865	79.7863	84.6393	80.9771
	ADASYN	80.6137	80.6137	83.7190	81.5272
	BorderlineSMOTE	79.1894	79.1894	83.7956	80.3810
Logistic Regression	RMO	82.7189	82.7189	81.6092	81.1802
	SMOTE	77.5765	77.5765	82.3115	78.8618
	RandomOver-sampler	77.9849	77.9849	82.2967	79.1930
	ADASYN	73.4290	73.4290	82.0581	75.3040
	BorderlineSMOTE	72.8530	72.8530	81.9683	74.7906
K-NN	RMO	83.7767	83.7767	83.2237	83.4188
	SMOTE	80.3729	80.3729	83.0612	81.2178
	RandomOver-sampler	78.1630	78.1630	82.4470	79.3591
	ADASYN	78.3724	78.3721	82.7001	79.5654
	BorderlineSMOTE	78.8856	78.8856	82.6506	79.9729

Table 4: undersampling results

Model	Method	Accuracy (%)	Recall (%)	Precision (%)	F1-Score (%)
Random Forest	RMO	85.3372	85.3372	84.7942	84.9421
	RandomUnder-sampler	81.5145	81.5145	85.0483	82.4604
	Tomek Links	85.3687	85.3687	85.0992	85.2129
	Edited Nearest Neighbors	82.1743	82.1743	85.6098	83.0794
	AllKNN	81.5668	81.5668	85.2422	82.5302
LightGBM	RMO	87.0339	87.0339	86.5608	86.6201
	RandomUnder-sampler	82.9179	82.9179	86.4347	83.8029
	Tomek Links	86.8873	86.8873	86.5876	86.6982
	Edited Nearest Neighbors	83.3368	83.3368	86.6204	84.1739
	AllKNN	82.8132	82.8132	86.2168	83.6877
SVM	RMO	85.0545	85.0545	84.3385	84.0741
	RandomUnder-sampler	79.0218	79.0218	85.2085	80.3799
	Tomek Links	85.0754	85.0754	84.3474	84.3261
	Edited Nearest Neighbors	81.6087	81.6087	84.8016	82.5002
	AllKNN	81.2840	81.2841	84.5132	82.1934
MLP	RMO	84.8240	84.8240	84.3486	84.5138
	RandomUnder-sampler	80.0691	80.0691	84.7628	81.2279
	Tomek Links	85.2011	85.2011	84.7228	84.8799
	Edited Nearest Neighbors	79.8597	79.8597	84.9308	81.0699
	AllKNN	81.4726	81.4726	84.5450	82.3499
Logistic Regression	RMO	82.7189	82.7189	81.6092	81.1802
	RandomUnder-sampler	77.6917	77.6917	82.2227	78.9440
	Tomek Links	82.8341	82.8341	81.7335	81.6749
	Edited Nearest Neighbors	80.4881	80.4881	81.8503	81.0104

	AllKNN	80.0901	80.0901	81.7046	80.6904
K-NN	RMO	83.7767	83.7767	83.2237	83.4188
	RandomUnder-sampler	78.4876	78.4876	83.3744	79.7476
	Tomek Links	83.6091	83.6091	83.3560	83.4695
	Edited Nearest Neighbors	80.8337	80.8337	83.7865	81.7130
	AllKNN	81.1793	81.1793	83.7933	81.9854

According to the table, most sampling approaches are ineffective on models, particularly oversampling strategies, which do not help to improve model performance and make it worse than the original. On the other hand, undersampling helps to improve the model slightly.

Only some models have been upgraded, other models have some parts that are better than the original when using the Tomek Links method mostly.

Table 5: oversampling results TP and FN bias

Model	Method	TP	FN	Precision (%)	F1-Score (%)
Random Forest	RMO	1421	860	84.7942	84.9421
	SMOTE	1596	685	84.5439	84.3437
	RandomOver-sampler	1572	709	84.2619	84.1055
	ADASYN	1590	691	84.3010	84.0708
	BorderlineSMOTE	1599	682	86.3587	86.0153
LightGBM	RMO	1476	805	86.5608	86.6201
	SMOTE	1718	563	86.3587	86.0153
	RandomOver-sampler	1909	372	86.7840	84.9849
	ADASYN	1741	540	85.9631	85.3531
	BorderlineSMOTE	1749	532	85.8054	85.0773
SVM	RMO	1226	1055	84.3385	84.0741
	SMOTE	1935	346	84.7325	80.6955
	RandomOver-sampler	1986	292	85.2725	80.6464
	ADASYN	2017	264	84.5020	78.0234
	BorderlineSMOTE	2011	270	84.5015	78.2080
MLP	RMO	1432	849	84.3486	84.5138
	SMOTE	1877	404	84.4724	81.3066
	RandomOver-sampler	1912	369	84.6393	80.9771
	ADASYN	1776	505	83.7190	81.5272
	BorderlineSMOTE	1855	426	83.7956	80.3810
Logistic Regression	RMO	1023	1258	81.6092	81.1802
	SMOTE	1779	502	82.3115	78.8618
	RandomOver-sampler	1759	522	82.2967	79.1930
	ADASYN	1903	378	82.0581	75.3040
	BorderlineSMOTE	1912	369	81.9683	74.7906

K-NN	RMO	1373	908	83.2237	83.4188
	SMOTE	1715	566	83.0612	81.2178
	RandomOver-sampler	1766	515	82.4470	79.3591
	ADASYN	1782	499	82.7001	79.5654
	BorderlineSMOTE	1752	529	82.6506	79.9729

Table 6: undersampling results TP and FN bias

Model	Method	TP	FN	Precision (%)	F1-Score (%)
Random Forest	RMO	1421	860	84.7942	84.9421
	RandomUnder-sampler	1873	408	85.0483	82.4604
	Tomek Links	1515	766	85.0992	85.2129
	Edited Nearest Neighbors	1899	382	85.6098	83.0794
	AllKNN	1891	390	85.2422	82.5302
LightGBM	RMO	1476	805	86.5608	86.6201
	RandomUnder-sampler	1949	332	86.4347	83.8029
	Tomek Links	1565	716	86.5876	86.6982
	Edited Nearest Neighbors	1947	334	86.6204	84.1739
	AllKNN	1931	350	86.2168	83.6877
SVM	RMO	1226	1055	84.3385	84.0741
	RandomUnder-sampler	1994	287	85.2085	80.3799
	Tomek Links	1294	967	84.3474	84.3261
	Edited Nearest Neighbors	1841	440	84.8016	82.5002
	AllKNN	1827	454	84.5132	82.1934
MLP	RMO	1432	849	84.3486	84.5138
	RandomUnder-sampler	1912	369	84.7628	81.2279
	Tomek Links	1443	838	84.7228	84.8799
	Edited Nearest Neighbors	1937	344	84.9308	81.0699
	AllKNN	1820	461	84.5450	82.3499
Logistic Regression	RMO	1023	1258	81.6092	81.1802
	RandomUnder-sampler	1765	516	82.2227	78.9440
	Tomek Links	1112	1169	81.7335	81.6749
	Edited Nearest Neighbors	1555	726	81.8503	81.0104
	AllKNN	1567	714	81.7046	80.6904
K-NN	RMO	1373	908	83.2237	83.4188
	RandomUnder-sampler	1844	437	83.3744	79.7476
	Tomek Links	1444	837	83.3560	83.4695
	Edited Nearest Neighbors	1771	510	83.7865	81.7130
	AllKNN	1751	530	83.7933	81.9854

The technique of sampling doesn't help much with model prediction bias. The oversampling method improves the model's ability to predict TP more than the baseline method, but it increases the FP of the

model. Undersampling improves TP more than oversampling, but only for some models. Only some models with Tomek Links improves model prediction bias. Additionally, other methods of undersampling cause the model to predict TP greater than the baseline. It's like oversampling methods.

Fairness metrics

We decided to use these matrices such as Disparate Impact (DI) and Equal Opportunity Difference (EOD) to measure model fairness.

Disparate Impact: Disparate Impact checks if a model treats different groups fairly. It looks at whether everyone has an equal chance of a favorable outcome. A value of 1 means equal treatment, while values far from 1 suggest potential unfairness.

Equal Opportunity Difference: Equal Opportunity Difference checks if the model gives everyone an equal chance to be correctly identified as positive. Smaller values indicate that both groups have similar opportunities to be correctly recognized.