Section One: Research Results

A. Classification Model Ranking:

Classification models are ranked based on algorithms.
To identify the highest scoring model and minimize computational resources, various algorithmic models are compared to evaluate the most superior algorithm for subsequent sections. The algorithms and their representative models are listed in Table 1.

Algorithm	**Representative Model**
Linear	- LogisticRegression
Tree-Based	- RandomForestClassifier
Boosting	- XGBClassifier
Distance-Based	- KNeighborsClassifier
Neural Network-Based	- MLPClassifier
Support Vector Machines	- SVC
Probabilistic	- GaussianNB
Ensemble	- Stacking (Logistic Regression, XGBClassifier, CatBoost, MLP) - Bagging (DecisionTreeClassifier, RandomForestClassifier, ExtraTreesClassifier) - Voting (Logistic Regression, XGBClassifier, SVC, KNeighborsClassifier)

Table 1: Representative models and their based-algorithms

This section tests twice using raw data and SMOTE-processed data.
The scores of representative models on four datasets are presented in Table 2.

		Dataset German				Dataset HMEQ				Dataset Taiwan			
		accuracy	f1	auc_roc	fin_score	accuracy	f1	auc_roc	fin_score	accuracy	f1	auc_roc	fin_score
Raw Data	**LogisticRegression**	**0.775**	0.6218	0.7982	65	0.8247	0.3365	0.7645	12	0.8043	0.3375	0.71	19
	RandomForestClassifier	0.755	0.5149	0.7687	33	0.9203	0.7754	**0.9737**	62	0.8098	0.4636	0.7604	54
	XGBClassifier	0.76	0.5556	0.7683	38	0.9304	0.8151	0.9686	60	0.806	0.4457	0.7541	41
	KNeighborsClassifier	0.715	0.4356	0.6757	7	0.8826	0.5858	0.9156	39	0.7915	0.4211	0.7031	15
	MLPClassifier	0.715	0.544	0.7382	18	0.8691	0.5851	0.8437	30	0.8133	0.4547	0.7666	57
	SVC	0.765	0.5688	0.7784	50	0.8372	0.4551	0.7991	21	0.8123	0.4074	0.7094	26
	GaussianNB	0.73	**0.625**	0.7811	49	0.7819	0.4037	0.7268	9	0.688	**0.4809**	0.7208	34
	StackingClassifier	0.75	0.537	**0.8037**	48	**0.9346**	**0.8274**	0.9728	67	**0.8142**	0.4558	**0.7716**	64
	BaggingClassifier	0.715	0.5128	0.7515	15	0.9086	0.7696	0.9283	46	0.7888	0.4629	0.728	35
	VotingClassifier	0.765	0.5607	0.7826	54	0.8792	0.574	0.9601	39	0.8113	0.4289	0.754	40
SMOTE	**LogisticRegression**	0.725	0.6099	**0.7973**	58	0.7408	0.4943	0.769	14	0.6885	0.4701	0.7114	21
	RandomForestClassifier	**0.725**	**0.6111**	0.7911	67	0.9346	0.8347	0.9696	50	0.7918	0.5018	0.754	52
	XGBClassifier	0.775	0.5946	0.7833	51	0.9446	**0.8565**	**0.9748**	68	0.8012	0.4747	0.7475	44
	KNeighborsClassifier	0.605	0.5031	0.6767	7	**0.9471**	0.8538	0.9258	53	0.6652	0.444	0.6842	11
	MLPClassifier	0.705	0.5354	0.7469	23	0.8364	0.6355	0.8576	28	0.7528	0.5139	0.7625	51
	SVC	0.74	0.5938	0.7671	39	0.7525	0.494	0.7717	17	0.7663	**0.5273**	0.7467	47
	GaussianNB	0.675	0.6061	0.7557	30	0.7735	0.4351	0.7332	11	0.3907	0.389	0.7081	10
	StackingClassifier	0.73	0.5424	0.7781	38	0.9396	0.8481	0.9728	59	**0.8073**	0.5122	**0.7657**	64
	BaggingClassifier	0.68	0.5152	0.7417	16	0.9077	0.7809	0.9419	38	0.756	0.4768	0.7239	32
	VotingClassifier	0.75	0.6094	0.7893	56	0.9346	0.8251	0.9571	45	0.7772	0.5165	0.7511	53

Table 2: Performances of classifiers with raw data or after smote data in three credit scoring datasets

- Fin_score = accuracy_score * 2 + f1_score * 2 + auc_roc_score *3

Fin_score is calculated based on the rank of each metric.
The rank ranges from [1:10], where 1 represents the worst and 10 the best score, as there are ten classifiers under evaluation. **Fin_score** is highlighted in yellow if **fin_score ≥ 60** and in red if **fin_score < 30**. Models are selected based on **fin_score**; for Table 2, yellow-highlighted models are given one additional point, while red-highlighted models are deducted one point. The final results are shown in Table 3.

Table 3 demonstrates that the Random Forest, XGBoost, and Stacking ensemble algorithms outperform the others.
The subsequent section focuses on utilizing tree-based models (RandomForestClassifier), boosting algorithms (XGBClassifier, CatBoostClassifier, LightGBMClassifier), and ensemble algorithms (StackingClassifier).

Model	Fin_score rank
LogisticRegression	-2
RandomForestClassifier	+2
XGBClassifier	+2

KNeighborsClassifier	-4
MLPClassifier	-3
SVC	-3
GaussianNB	-3
StackingClassifier	+3
BaggingClassifier	-2
VotingClassifier	+0

Table 3: Model selection

B. Final result

1. Comparison of Raw Data with Oversampling (OS) and Undersampling (US) Methods

The effectiveness of data balancing methods depends on the dataset.

As shown in Table 4, SMOTE, a common oversampling technique, improves model performance, particularly for the German and HMEQ datasets. However, this improvement is not observed in the Taiwan and Home Credit Default Risk datasets, suggesting that SMOTE is not universally applicable. In the subsequent section, various balancing techniques will be applied and compared to results obtained without data balancing.

		Dataset German			Dataset HMEQ			Dataset Taiwan			Dataset HomeCreditDefaultRisk		
		accuracy	**f1**	**auc_roc**	**accuracy**	**f1**	**auc_roc**	**accuracy**	**f1**	**auc_roc**	**accuracy**	**f1**	**auc_roc**
No balancing, using raw data	**RandomForestClassifier**	0.755	0.5149	0.7687	0.9203	0.7754	0.9737	0.8098	**0.4636**	0.7604	**0.9261**	0.1494	0.7026
	XGBClassifier	0.76	**0.5556**	0.7683	**0.9304**	**0.8151**	0.9686	0.806	0.4457	0.7541	0.9196	0.0999	0.7085
	CatBoostClassifier	0.755	0.5421	**0.8006**	0.9253	0.7954	0.972	0.8112	0.454	0.768	0.9202	0.0518	0.7232
	LightGBMClassifier	0.73	0.5	0.7667	0.9195	0.7828	0.9658	**0.8147**	0.4633	0.7681	0.9194	0.0317	0.717
	StackingClassifier	**0.765**	0.5524	0.7892	0.9287	0.8107	**0.976**	0.814	0.4572	**0.773**	0.9252	**0.2041**	**0.7313**
SMOTE	**RandomForestClassifier**	0.755	0.5664	0.7788	0.9312	0.8292	0.9706	0.7933	0.5032	0.7511	0.9188	0.0773	0.6747
	XGBClassifier	0.735	0.5546	0.7742	0.9362	0.8376	0.9727	0.8018	0.4819	0.7374	0.9171	0.062	0.6864
	CatBoostClassifier	**0.775**	**0.6154**	**0.8154**	0.9329	0.8246	0.9708	**0.8132**	0.4788	**0.7654**	**0.9199**	0.0413	**0.7069**
	LightGBMClassifier	0.735	0.547	0.7749	0.9262	0.807	0.967	0.8085	0.4877	0.7601	0.9197	0.0186	0.695
	StackingClassifier	**0.775**	0.6018	0.7936	**0.9388**	**0.845**	**0.9751**	0.803	**0.5034**	0.7619	0.9138	**0.1292**	0.6703
Cluster Centroid	**RandomForestClassifier**	0.59	**0.5495**	**0.7568**	**0.8926**	**0.7739**	**0.9533**	**0.5463**	**0.4296**	**0.6823**	**0.0988**	**0.1505**	0.5324
	XGBClassifier	0.56	0.5217	0.6967	0.823	0.6759	0.945	0.4537	0.4001	0.6306	0.0939	0.1501	0.5526
	CatBoostClassifier	0.59	**0.5495**	0.7413	0.8515	0.7131	0.9415	0.4557	0.406	0.6553	0.0899	0.1499	**0.5544**
	LightGBMClassifier	0.57	0.5376	0.6987	0.8163	0.6687	0.9405	0.4748	0.4111	0.6587	0.0881	0.1493	0.5471
	StackingClassifier	**0.6**	0.5455	0.7425	0.8574	0.724	0.9498	0.4535	0.4037	0.6434	0.0928	0.1498	0.5417

Table 4: Performances of classifiers with raw data and oversampling/undersampling data in four credit scoring datasets

2. Comparison of Raw Data with Hybrid Methods, Class Weight, Ensemble Methods, and Neural Networks (NN)

Table 5 presents the performance of models using raw data compared with other techniques (Hybrid Method, Class Weight, Ensemble Method, and Neural Networks):

			Dataset German			Dataset HMEQ			Dataset Taiwan			Dataset HomeCreditDefaultRisk		
Type of balancing	Balancing method	Model	accuracy	f1	auc_roc	accuracy	f1	auc_roc	accuracy	f1	auc_roc	accuracy	f1	auc_roc
No type as using raw data	No balancing, using raw data	RandomForestClassifier	0.755	0.5149	0.7687	0.9203	0.7754	0.9737	0.8098	**0.4636**	0.7604	**0.9261**	0.1494	0.7026
		XGBClassifier	0.76	**0.5556**	0.7683	**0.9304**	**0.8151**	0.9686	0.806	0.4457	0.7541	0.9196	0.0999	0.7085
		CatBoostClassifier	0.755	0.5421	**0.8006**	0.9253	0.7954	0.972	0.8112	0.454	0.768	0.9202	0.0518	0.7232
		LightGBMClassifier	0.73	0.5	0.7667	0.9195	0.7828	0.9658	**0.8147**	0.4633	0.7681	0.9194	0.0317	0.717
		StackingClassifier	**0.765**	0.5524	0.7892	0.9287	0.8107	**0.976**	0.814	0.4572	**0.773**	0.9252	**0.2041**	**0.7313**
			accuracy	f1	auc_roc	accuracy	f1	auc_roc	accuracy	f1	auc_roc	accuracy	f1	auc_roc
Hybrid methods (OS + US)	SMOTEENN	RandomForestClassifier	0.68	0.5789	0.7746	0.927	0.8161	0.9665	0.7625	0.5194	0.7655	0.5346	0.1946	0.6528
		XGBClassifier	**0.69**	**0.5867**	0.7651	**0.9354**	0.83	0.9673	0.7702	0.521	0.7519	0.6978	0.2219	0.6741
		CatBoostClassifier	0.675	0.5806	**0.7769**	0.9304	0.8184	0.9671	**0.7833**	**0.5276**	0.7648	0.697	**0.2269**	**0.684**
		LightGBMClassifier	0.67	0.5658	0.7546	0.9262	0.8087	0.9631	0.7733	0.5208	**0.7677**	**0.7056**	0.2202	0.6716
		StackingClassifier	0.67	0.5658	0.7746	0.9346	**0.8326**	**0.9711**	0.7662	0.5239	0.7646	0.6691	0.2031	0.6412
			accuracy	f1	auc_roc	accuracy	f1	auc_roc	accuracy	f1	auc_roc	accuracy	f1	auc_roc
	SMOTETomek	RandomForestClassifier	0.765	0.5766	0.789	0.9346	0.8354	0.9687	0.7948	**0.5121**	0.7557	0.9183	0.0716	0.681
		XGBClassifier	0.77	0.5893	0.7633	0.932	0.8243	0.9717	0.802	0.4857	0.7417	0.9185	0.076	0.6861
		CatBoostClassifier	**0.775**	**0.6341**	**0.8089**	0.9295	0.8166	0.9675	**0.8133**	0.4848	**0.7661**	**0.9198**	0.0389	**0.7128**
		LightGBMClassifier	0.755	0.5664	0.7856	0.932	0.822	0.965	0.8112	0.5033	0.7648	0.9196	0.0174	0.7018
		StackingClassifier	0.77	0.6034	0.7999	**0.9354**	**0.8358**	**0.973**	0.8042	0.5118	0.7617	0.9156	**0.1351**	0.6762
			accuracy	f1	auc_roc	accuracy	f1	auc_roc	accuracy	f1	auc_roc	accuracy	f1	auc_roc
Classweight (balance the weight of classes)		RandomForestClassifier	**0.78**	0.551	0.7739	0.9161	0.763	0.9704	**0.808**	0.444	0.7605	**0.9233**	0.0903	0.7162
		XGBClassifier	0.75	0.5763	0.7825	**0.9354**	**0.8406**	0.9719	0.7525	0.5045	0.7468	0.8159	0.2685	0.7017
		CatBoostClassifier	0.77	**0.6406**	**0.8142**	0.9203	0.8141	0.9694	0.7688	0.5328	0.7672	0.8212	**0.2876**	0.7285
		LightGBMClassifier	0.73	0.55	0.7602	0.9195	0.8095	0.9681	0.764	**0.5336**	0.7699	0.743	0.2562	0.7227
		StackingClassifier	0.73	0.6197	0.8038	0.927	0.8355	**0.9741**	0.7527	0.5268	**0.7722**	0.7176	0.2664	**0.7483**
Ensemble method (models that balance the data)		BalancedRandomForestClassifier	0.64	0.5663	0.7733	0.8943	0.7797	0.9649	0.7073	0.5073	0.7685	0.6528	0.2386	0.7313
			accuracy	f1	auc_roc	accuracy	f1	auc_roc	accuracy	f1	auc_roc	accuracy	f1	auc_roc
Neural Network	cWGAN for tabular data	RandomForestClassifier	0.74	0.4348	0.7647	0.9362	0.8249	0.9798	0.8135	0.4709	0.7639	0.8084	**0.223**	**0.6471**
		XGBClassifier	**0.78**	**0.551**	0.7579	0.9362	0.8296	0.9738	0.815	0.4674	0.7668	0.7592	0.168	0.58
		CatBoostClassifier	0.76	0.5294	0.7774	0.9295	0.8073	0.9671	0.8222	0.4833	0.7823	0.771	0.1687	0.5794
		LightGBMClassifier	0.77	0.549	0.7682	0.9228	0.79	0.9611	**0.8223**	0.4795	0.7799	0.7143	0.1733	0.5777
		StackingClassifier	0.75	0.5283	**0.7799**	**0.9388**	**0.8482**	**0.9821**	0.8195	**0.5163**	**0.783**	**0.8093**	0.2177	0.64

Table 5: Performances of classifiers with raw data and other sampling techniques in four credit scoring datasets

In Table 5, performance scores are highlighted in green when they are the highest among the combinations of balancing techniques and models.
It is evident that most of the highest scores are found in the class weight and neural network-based approaches. Notably, the CatBoost and Stacking models stand out as significantly superior compared to the other models.

Table 6 clearly highlights the superiority of the class weight technique and neural networks.

	Dataset German			Dataset HMEQ			Dataset Taiwan			Dataset HomeCreditDefaultRisk		
	accuracy	f1	auc_roc	accuracy	f1	auc_roc	accuracy	f1	auc_roc	accuracy	f1	auc_roc
Raw data, no balancing	0	0	0	0	0	0	0	0	0	**0**	0	0
SMOTE (oversampling)	0.01	0.0598	**0.0148**	**0.0084**	0.0299	-0.0009	-0.0015	0.0398	-0.0076	-0.0062	-0.0749	-0.0244
ClusterCentroid (undersampling)	-0.165	-0.0061	-0.0438	-0.0378	-0.0412	-0.0227	-0.2684	-0.034	-0.0907	-0.8273	-0.0536	-0.1769
SMOTEENN (hybrid: OS and US)	-0.075	0.0311	-0.0237	0.005	0.0175	-0.0049	-0.0314	0.064	-0.0053	-0.2205	0.0228	-0.0473
SMOTETomek (hybrid: OS and US)	0.01	0.0785	0.0083	0.005	0.0207	-0.003	-0.0014	0.0485	-0.0069	-0.0063	-0.069	-0.0185
Balanced Random Forest (ensemble)	-0.125	0.0107	-0.0273	-0.0361	-0.0354	-0.0111	-0.1074	0.0437	-0.0045	-0.2733	0.0345	0
Class Weights	**0.015**	**0.085**	0.0136	0.005	0.0255	-0.0019	-0.0067	**0.07**	-0.0008	-0.0028	**0.0835**	**0.017**
cWGAN (NN)	**0.015**	-0.0046	-0.0207	**0.0084**	**0.0331**	**0.0061**	**0.0076**	0.0527	**0.01**	-0.1168	0.0189	-0.0842

Table 6: The difference of performances when using balancing techniques with using raw data. The bolded scores are the highest positive differences.

Section two: Conclusion

The application of data balancing techniques offers significant benefits, especially in credit scoring, where imbalanced data is common. Among the methods, the class weight balancing technique has proven to be exceptionally effective due to its simplicity, ease of application, and ability to enhance model performance. Additionally, neural network models demonstrate substantial potential due to their flexibility in learning and ability to process data more efficiently, particularly in generating strong and robust features. This gives neural networks an advantage over traditional methods like SMOTE, which can struggle with highly complex datasets.

Boosting models like XGBoost and CatBoost also show better performance compared to the other models. The final results indicate that the top combination is a neural network combined with stacking techniques, leveraging the strengths of multiple models. The second position is held by CatBoost with the class weight balancing technique. These combinations highlight the importance of data balancing techniques.

Besides neural network-based approaches like cWGAN, other methods such as semi-supervised learning or graph neural networks (GNNs) are also being explored to balance data more effectively. These approaches show great potential in handling complex, structured data. However, implementing and optimizing these techniques remains a challenge that requires extensive research.