CS-UY 4563: Machine Learning:

Final Project Written Report

Bank Marketing Prediction

11:00 AM Section

Dec 7, 2022

Professor Linda N. Sellie

Shao Jin, Kevin Zhou

1. Introduction

In this project, we focused on using machine learning models to predict whether the client will subscribe to a term deposit product in the bank. The client dataset was taken from OpenML and included 16 input variables. Some of them are related to clients: age, job, marital status, education level, default credit, average yearly balance, housing loan, and personal loan. Some of them are related to the contact information: contact communication type, last contact duration, number of contacts performed during this campaign and for this client, number of contacts performed before this campaign and for this client, and outcome of the previous marketing campaign.

There are 54211 rows of data in total. We randomly selected 10000 samples from the dataset and split them into 7000 training samples and 3000 testing samples.

We use supervised analysis (logistic regression, support vector machines, and neural networks) in classifying clients and making predictions. All three models were run multiple times using different regularization, feature transformation, and normalization, and we evaluated each model's result.

2. Data Preprocessing

2.1. Cleaning Dataset

In our dataset, there might be some mistyped data or invalid data. In order to align with the other data in the dataset, we wrote an algorithm to replace them with the number ‘0’. But the result shows that there actually is not any invalid data in our dataset, because the number of samples is unchanged after doing the algorithm.

2.2. Categorical Encoding

Some of the features in the dataset were not numerical, which makes it difficult for us to train. We encoded features by labeling them with numbers starting from ‘1’. For example, marital status had 3 categories: "married", "divorced", and "single". We labeled “married” as 1, “divorced” as 2, and “single” as 3.

2.3. Data Scaling

In our dataset, there are also some features that have a large range of data values. The three input features - age, bank balance, and last contact duration - have drastically varied values in the dataset. We normalize them to let them have mean values of 0 and scaled all data points by calculating their z-scores, thus scaling the data to be normally distributed.

2.4. Data Splitting

When we deal with the dataset, we discovered that all samples of class 1 (the clients that did not subscribe to the product) come first, and then all samples of class 2. In order to prevent the situation that all samples we choose are of the same class, we wrote a function to randomly choose 10000 samples from our dataset. Then we split them into 7000 training sets and 3000 testing sets for future use.

2.5. Data Transformation

We also did a data transformation and created X_train_2 and X_set_2, in which we took the square of all data in the original dataset. We used them in the data analysis part.

After doing all of the transformations, as shown above, a random row of data would look like:
[1.037371 1 1 2 2 -0.309556 1 2 3 7.0 5 -0.687574 1.0 -1.0 0.0 1]
[1.076139 1 1 4 4 0.095825 1 4 9 49.0 25 0.472758 1.0 1.0 0.0 1]
The first row is in X_train or X_test, and the second row is in X_train_2 or X_test_2.

3. Supervised Analysis

3.1. Logistic Regression

The first model we use in our project is logistic regression. We tried 4 different regularization settings: no regularization, L1 regularization, L2 regularization, and elastic-net regularization. For elastic-net regularization, we choose L1_ratio = 0.5, which means a model with 50% L1 regularization and 50% L2 regularization. Then, for each regularization setting, we train both X_train and X_train_2. Also, we train all the logistic regression models with 11 different C-values: [0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000, 100000]. In total, we trained 67 logistic regression models.

For each logistic regression model, we got their corresponding training accuracy and testing accuracy. For each regularization setting and data transformation setting combination, we plot the training scores and testing scores versus C-values to see the relationship between model performance and C-value. Also, we selected the model with the highest testing accuracy in each regularization setting, and print out their classification report, which include precision, F1 score, accuracy, etc. The followings are the data:

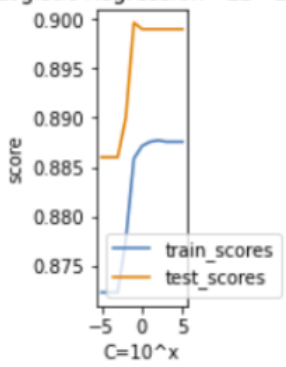
Logistic Regression - No regularization - 1

	precision	recall	f1-score	support
1	0.91	0.98	0.94	2658
2	0.63	0.28	0.39	342
accuracy			0.90	3000
macro avg	0.77	0.63	0.67	3000
weighted avg	0.88	0.90	0.88	3000

Logistic Regression - No regularization - 2

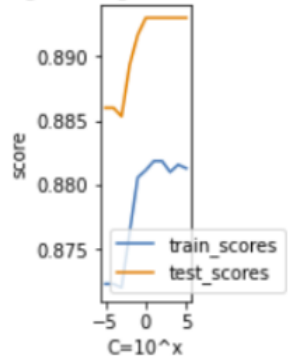
	precision	recall	f1-score	support
1	0.90	0.99	0.94	2658
2	0.56	0.13	0.21	342
accuracy			0.89	3000
macro avg	0.73	0.56	0.58	3000
weighted avg	0.86	0.89	0.86	3000

Logistic Regression - L1 - 1



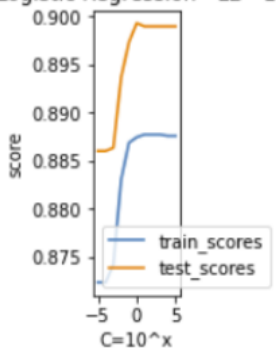
	precision	recall	f1-score	support
1	0.91	0.98	0.95	2658
2	0.65	0.26	0.37	342
accuracy			0.90	3000
macro avg	0.78	0.62	0.66	3000
weighted avg	0.88	0.90	0.88	3000

Logistic Regression - L1 - 2



	precision	recall	f1-score	support
1	0.90	0.99	0.94	2658
2	0.61	0.17	0.26	342
accuracy			0.89	3000
macro avg	0.76	0.58	0.60	3000
weighted avg	0.87	0.89	0.86	3000

Logistic Regression - L2 - 1

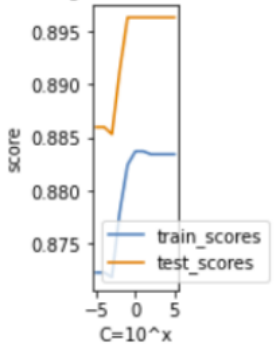


	precision	recall	f1-score	support
1	0.91	0.98	0.95	2658
2	0.63	0.28	0.39	342
accuracy			0.90	3000
macro avg	0.77	0.63	0.67	3000
weighted avg	0.88	0.90	0.88	3000

Logistic Regression - L2 - 2

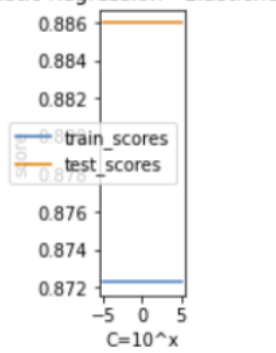
	precision	recall	f1-score	support
1	0.90	0.98	0.94	2658
2	0.58	0.17	0.26	342
accuracy			0.89	3000
macro avg	0.74	0.58	0.60	3000
weighted avg	0.86	0.89	0.86	3000

Logistic Regression - Elasticnet - 1



	precision	recall	f1-score	support
1	0.91	0.98	0.94	2658
2	0.62	0.23	0.34	342
accuracy			0.90	3000
macro avg	0.77	0.61	0.64	3000
weighted avg	0.88	0.90	0.87	3000

Logistic Regression - Elasticnet - 2



	precision	recall	f1-score	support
1	0.89	1.00	0.94	2658
2	0.00	0.00	0.00	342
accuracy			0.89	3000
macro avg	0.44	0.50	0.47	3000
weighted avg	0.78	0.89	0.83	3000

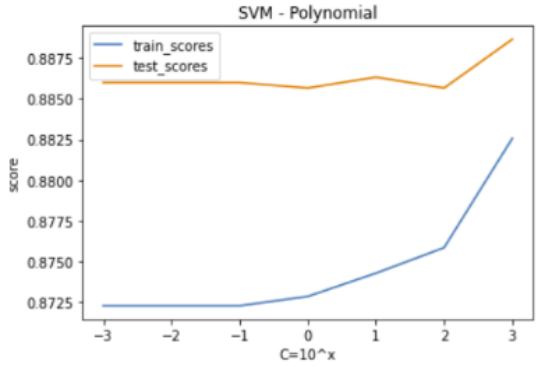
3.2. SVM

We also trained our dataset with the Support Vector Machine (SVM). In this part, we tried 4 different kernel settings: linear kernel, polynomial kernel, RBF kernel, and sigmoid kernel. Then, similar to the logistic regression, we train these 4 kernel settings with 7 different C-values: [0.001, 0.01, 0.1, 1, 10, 100, 1000]. We reduced the number of C-values because the SVM algorithm is more time-consuming and complicated than the logistic regression, and running too many iterations might crash the computer. In total, we trained 28 logistic regression models.

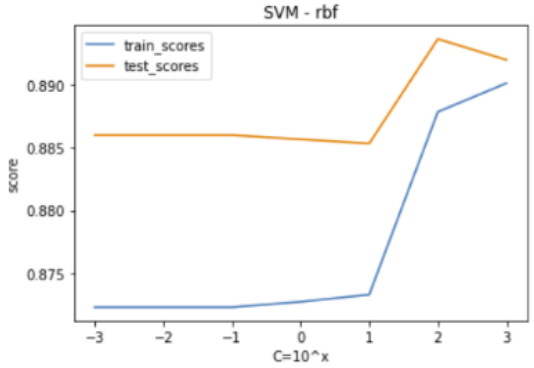
Then, for each SVM model we trained, we got their corresponding training accuracy and testing accuracy. For each kernel setting, we plot the training scores and testing scores versus C-values to see the relationship between model performance and C-value. Also, we selected the model with the highest testing accuracy in each kernel setting, and print out their classification report. The followings are the data:

	precision	recall	f1-score	support
1	0.91	0.98	0.94	2658
2	0.59	0.26	0.36	342
accuracy			0.90	3000
macro avg	0.75	0.62	0.65	3000
weighted avg	0.87	0.90	0.88	3000

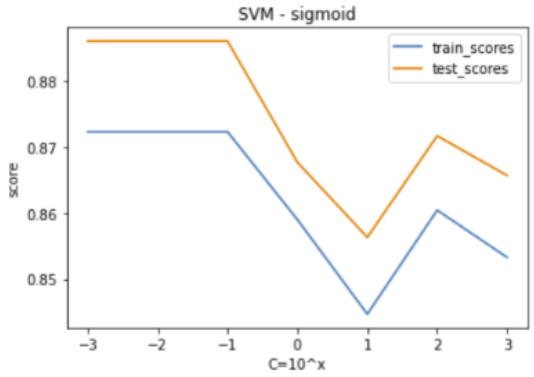
	precision	recall	f1-score	support
1	0.89	0.99	0.94	2658
2	0.57	0.09	0.16	342
accuracy			0.89	3000
macro avg	0.73	0.54	0.55	3000
weighted avg	0.86	0.89	0.85	3000



	precision	recall	f1-score	support
1	0.90	0.99	0.94	2658
2	0.63	0.16	0.26	342
accuracy			0.89	3000
macro avg	0.77	0.57	0.60	3000
weighted avg	0.87	0.89	0.86	3000



	precision	recall	f1-score	support
1	0.89	1.00	0.94	2658
2	0.00	0.00	0.00	342
accuracy			0.89	3000
macro avg	0.44	0.50	0.47	3000
weighted avg	0.78	0.89	0.83	3000



3.3.Neural Network

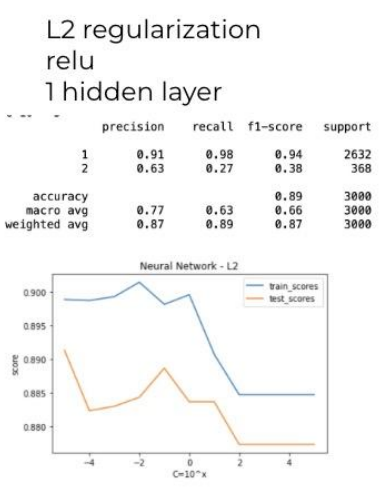
The last model we used in this project is Neural Network. We used different types of activation functions including Rectified Linear Unit, Logistic, and tanh. For our data set, we had 16 features, so we have 16 neurons in the input layer. It was a classification problem, so we had 1 neuron in the output layer. Then we combined these activation functions with different neural network hidden layers: one hidden layer of neurons (10), two hidden layers of neurons(10, 5), and no hidden layer. At last, we combined the three neural networks with different Ridge Regression configurations where the hyperparameter C-values = [0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000, 100000].

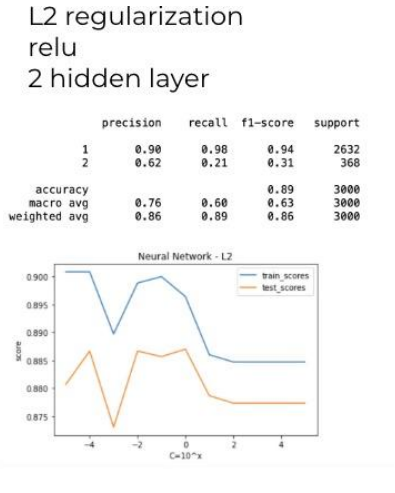
The following is the no-regularization part.

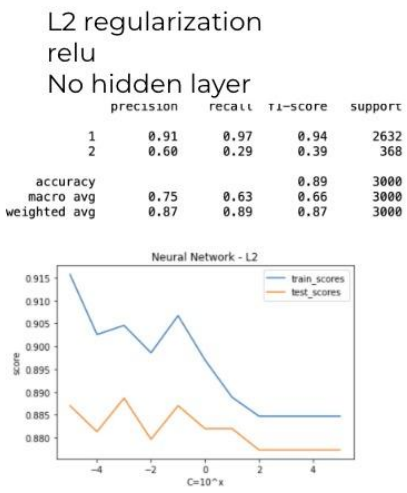
	Tanh	Logistic	Relu
No hidden layer	Train:0.949 Test:0.886	Train:0.936 Test:0.885	Train:0.914 Test:0.882
One hidden layer	Train:0.902 Test:0.886	Train:0.898 Test:0.888	Train:0.889 Test:0.889
Two hidden layer	Train:0.9 Test:0.879	Train:0.896 Test:0.886	Train:0.902 Test:0.886

The followings are respective graphs. The x-axis is the C-values. The y-axis is the accuracy score.

Relu:



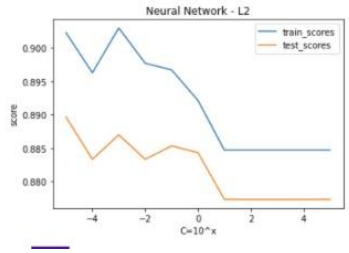




Logistic:

L2 regularization
logistic
1 hidden layer

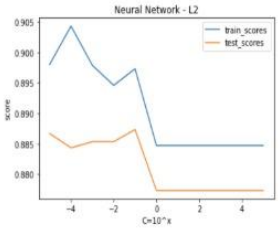
	precision	recall	f1-score	support
1	0.91	0.97	0.94	2632
2	0.59	0.32	0.41	368
accuracy			0.89	3000
macro avg	0.75	0.64	0.68	3000
weighted avg	0.87	0.89	0.87	3000



L2 regularization
logistic
2 hidden layer

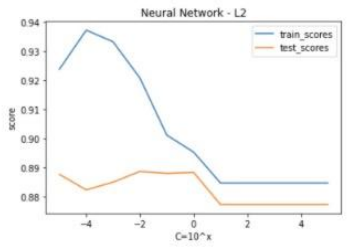
C=10^ -1

	precision	recall	f1-score	support
1	0.91	0.97	0.94	2632
2	0.58	0.30	0.39	368
accuracy			0.89	3000
macro avg	0.74	0.63	0.66	3000
weighted avg	0.87	0.89	0.87	3000



L2 regularization
logistic
No hidden layer

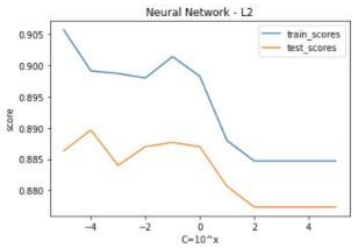
	precision	recall	f1-score	support
1	0.92	0.96	0.94	2632
2	0.57	0.36	0.45	368
accuracy			0.89	3000
macro avg	0.74	0.66	0.69	3000
weighted avg	0.87	0.89	0.88	3000



Tanh:

L2 regularization
tanh
1 hidden layer

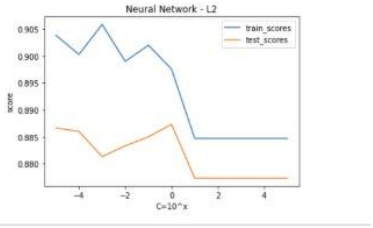
	precision	recall	f1-score	support
1	0.91	0.97	0.94	2632
2	0.59	0.34	0.43	368
accuracy			0.89	3000
macro avg	0.75	0.65	0.69	3000
weighted avg	0.87	0.89	0.88	3000



L2 regularization
tanh
2 hidden layer

C=10^ 0

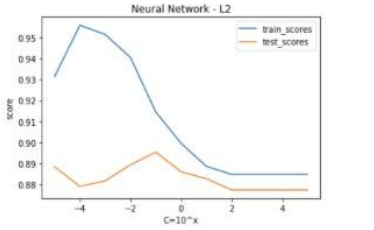
	precision	recall	f1-score	support
1	0.91	0.97	0.94	2632
2	0.59	0.27	0.37	368
accuracy			0.89	3000
macro avg	0.75	0.62	0.65	3000
weighted avg	0.87	0.89	0.87	3000



L2 regularization
tanh
No hidden layer

C=10^ -1

	precision	recall	f1-score	support
1	0.91	0.97	0.94	2632
2	0.65	0.33	0.43	368
accuracy			0.90	3000
macro avg	0.78	0.65	0.69	3000
weighted avg	0.88	0.90	0.88	3000



4. Results

In this part, we get the train accuracy and test accuracy of all the models we have trained.

4.1. Logistic Regression

C=10^	-5	-4	-3	-2	-1	0	1	2	3	4	5
L2, X^1, train score	0.8839	0.8839	0.8866	0.8953	0.8996	0.9009	0.9007	0.9007	0.9006	0.9007	0.9009
L2, X^1, test score	0.8803	0.8800	0.8827	0.8903	0.8953	0.8963	0.8963	0.8960	0.8960	0.8960	0.8960
L2, X^2, train score	0.8841	0.8866	0.8919	0.8934	0.8937	0.8926	0.8927	0.8940	0.8927	0.8957	0.8946
L2, X^2, test score	0.8797	0.8820	0.8873	0.8873	0.8880	0.8883	0.8877	0.8883	0.8867	0.8900	0.8893
L1, X^1, train score	0.8839	0.8839	0.8839	0.8906	0.8997	0.9007	0.9007	0.9007	0.9007	0.9007	0.9007
L1, X^1, test score	0.8803	0.8803	0.8803	0.8850	0.8960	0.8967	0.8963	0.8963	0.8963	0.8963	0.8963
L1, X^2, train score	0.8839	0.8839	0.8860	0.8930	0.8974	0.8974	0.8976	0.8976	0.8976	0.8976	0.8976
L1, X^2, test score	0.8803	0.8803	0.8810	0.8890	0.8907	0.8917	0.8917	0.8917	0.8917	0.8917	0.8917
Elasticnet, X^1, train score	0.8839	0.8839	0.8843	0.8900	0.8977	0.8976	0.8976	0.8974	0.8974	0.8974	0.8974
Elasticnet, X^1, test score	0.8803	0.8803	0.8803	0.8870	0.8910	0.8927	0.8927	0.8930	0.8930	0.8930	0.8930
Elasticnet, X^2, train score	0.8839	0.8839	0.8839	0.8839	0.8839	0.8839	0.8839	0.8839	0.8839	0.8839	0.8839
Elasticnet, X^2, test score	0.8803	0.8803	0.8800	0.8800	0.8800	0.8800	0.8800	0.8800	0.8800	0.8800	0.8800

4.2. SVM

C=10^	-3	-2	-1	0	1	2	3
Linear, train score	0.8810	0.8810	0.8814	0.8814	0.8874	0.8890	0.8846
Linear, test score	0.8903	0.8903	0.8910	0.8910	0.8953	0.8977	0.8950
Polynomial, train score	0.8810	0.8810	0.8810	0.8811	0.8827	0.8836	0.8893
Polynomial, test score	0.8903	0.8903	0.8903	0.8907	0.8900	0.8907	0.8937
RBF, train score	0.8810	0.8810	0.8810	0.8810	0.8814	0.8941	0.9000
RBF, test score	0.8903	0.8903	0.8903	0.8903	0.8893	0.8957	0.8943
Sigmoid, train score	0.8810	0.8810	0.8810	0.8746	0.8554	0.8569	0.8511
Sigmoid, test score	0.8903	0.8903	0.8903	0.8813	0.8593	0.8610	0.8517

4.3.1. Neural Network(No regularization)

	Tanh	Logistic	Relu
No hidden layer	Train:0.949 Test:0.886	Train:0.936 Test:0.885	Train:0.914 Test:0.882
One hidden layer	Train:0.902 Test:0.886	Train:0.898 Test:0.888	Train:0.889 Test:0.889
Two hidden layer	Train:0.9 Test:0.879	Train:0.896 Test:0.886	Train:0.902 Test:0.886

4.3.2. Neural Network(L2 regularization)

C→ Layer ↓	0.00001	0.0001	0.001	0.01	0.1	1	10	100	1000	10000	100000
No hidden layer (Relu)	Train: 0.9039 Test: 0.8807	Train: 0.9041 Test: 0.8833	Train: 0.9069 Test: 0.8897	Train: 0.8994 Test: 0.8753	Train: 0.9043 Test: 0.8863	Train: 0.9 Test: 0.8887	Train: 0.8886 Test: 0.8820	Train: 0.8847 Test: 0.8773	Train: 0.8847 Test: 0.8773	Train: 0.8847 Test: 0.8773	Train: 0.8847 Test: 0.8773
One hidden layer (Relu)	Train: 0.8979 Test: 0.885	Train: 0.8939 Test: 0.8797	Train: 0.8956 Test: 0.8817	Train: 0.8961 Test: 0.8853	Train: 0.8976 Test: 0.882	Train: 0.8967 Test: 0.8853	Train: 0.8907 Test: 0.8867	Train: 0.8847 Test: 0.8773	Train: 0.8847 Test: 0.8773	Train: 0.8847 Test: 0.8773	Train: 0.8847 Test: 0.8773
Two hidden layer (Relu)	Train: 0.8967 Test: 0.8827	Train: 0.8921 Test: 0.8843	Train: 0.897 Test: 0.8857	Train: 0.9043 Test: 0.8873	Train: 0.8997 Test: 0.8903	Train: 0.8971 Test: 0.884	Train: 0.8883 Test: 0.8817	Train: 0.8847 Test: 0.8773	Train: 0.8847 Test: 0.8773	Train: 0.8847 Test: 0.8773	Train: 0.8847 Test: 0.8773
No hidden layer (Logistic)	Train: 0.9194 Test: 0.880	Train: 0.9483 Test: 0.8800	Train: 0.9119 Test: 0.89	Train: 0.9169 Test: 0.897	Train: 0.9009 Test: 0.8857	Train: 0.8963 Test: 0.89	Train: 0.8847 Test: 0.8773	Train: 0.8847 Test: 0.8773	Train: 0.8847 Test: 0.8773	Train: 0.8847 Test: 0.8773	Train: 0.8847 Test: 0.8773
One hidden layer (Logistic)	Train: 0.9003 Test: 0.886	Train: 0.8996 Test: 0.887	Train: 0.9001 Test: 0.887	Train: 0.8989 Test: 0.8847	Train: 0.8971 Test: 0.888	Train: 0.8921 Test: 0.8837	Train: 0.8847 Test: 0.8773	Train: 0.8847 Test: 0.8773	Train: 0.8847 Test: 0.8773	Train: 0.8847 Test: 0.8773	Train: 0.8847 Test: 0.8773
Two hidden layer (Logistic)	Train: 0.8973 Test: 0.883	Train: 0.8991 Test: 0.8847	Train: 0.89 Test: 0.888	Train: 0.8877 Test: 0.884	Train: 0.897 Test: 0.8877	Train: 0.8847 Test: 0.8773	Train: 0.8847 Test: 0.8773	Train: 0.8847 Test: 0.8773	Train: 0.8847 Test: 0.8773	Train: 0.8847 Test: 0.8773	Train: 0.8847 Test: 0.8773
No hidden layer (Tanh)	Train: 0.951 Test: 0.8847	Train: 0.9516 Test: 0.8757	Train: 0.9456 Test: 0.8878	Train: 0.9374 Test: 0.885	Train: 0.9174 Test: 0.8903	Train: 0.8976 Test: 0.886	Train: 0.8891 Test: 0.882	Train: 0.8847 Test: 0.8773	Train: 0.8847 Test: 0.8773	Train: 0.8847 Test: 0.8773	Train: 0.8847 Test: 0.8773
One hidden layer (Tanh)	Train: 0.8969 Test: 0.8877	Train: 0.8987 Test: 0.8837	Train: 0.9001 Test: 0.8810	Train: 0.899 Test: 0.8867	Train: 0.901 Test: 0.8887	Train: 0.8999 Test: 0.887	Train: 0.8873 Test: 0.881	Train: 0.8847 Test: 0.8773	Train: 0.8847 Test: 0.8773	Train: 0.8847 Test: 0.8773	Train: 0.8847 Test: 0.8773
Two hidden layer (Tanh)	Train: 0.9024 Test: 0.8807	Train: 0.9011 Test: 0.89	Train: 0.9034 Test: 0.886	Train: 0.9039 Test: 0.885	Train: 0.9059 Test: 0.889	Train: 0.9006 Test: 0.8897	Train: 0.8847 Test: 0.8773	Train: 0.8847 Test: 0.8773	Train: 0.8847 Test: 0.8773	Train: 0.8847 Test: 0.8773	Train: 0.8847 Test: 0.8773

5. Conclusion

Logistic Regression:

According to the accuracy table for logistic regression as shown in the results part, the best model for the logistic regression is the model where we train the dataset with no data transformation, L2 regularization, and C = 1. We ran the whole program for multiple times, and the best model is always with this setting. This is exactly the default setting of sklearn logistic regression function.

Also, according to the table and the plots for logistic regression, we can see that smaller C-value leads to lower testing accuracy for all the models. This is because smaller C-values cause overfitting.

V12	duration	1.051400069906811
V7	housing	0.8977622400989785
V16	poutcome	0.7680505773823237
V8	loan	0.4828391047416844
V9	contact	0.47682649539725935
V5	credit	0.4095914643939325
V4	education	0.2237617327764284
V3	marital	0.22180476426109405
V13	campaign	-0.1089097149160986
V1	age	0.08516346963207476
V6	balance	0.02167827232824146
V10	day	-0.009262085110660526
V2	job	-0.006951327081755144
V14	pdays	-0.0020429956792170365
V11	month	-0.0008688615578767077
V15	previous	0.0007194549175827377

The above table shows W values for all features. We sort it according to the absolute value of W. Thus, the order they are shown in the table represents their weights in the model. We can see that the "duration" feature is the first feature. This means that the duration of last contact with the client is the most important feature to be considered in this model. This result makes sense because: if the staff in the bank can have a long and detailed talk with the client, there is a much higher probability for him to successfully sell the product to this client, compared to the clients who hang up the phone very quickly.

The model has an accuracy of 90.09% for the training set, and an accuracy of 89.63% for the testing set. These accuracy values are relatively high, which means that the model is accurate.

SVM:

According to the accuracy table for the SVM as shown in the results part, the best model for the SVM is the model where we train the dataset with linear kernel setting. We ran the whole program for multiple times as well, and the best model is always with linear kernel. But the best C-value setting is different every time, even though the overall accuracy for both the training set and the testing set is approximately the same.

V5	credit	-0.48094938255862535
V8	loan	-0.3013407941188018
V12	duration	0.17244918583304264
V16	poutcome	0.1582036694208702
V1	age	0.1353747179898286
V7	housing	0.0948856592627792
V3	marital	0.046066845866337636
V4	education	0.01634695424913625
V9	contact	0.01048587347836185
V13	campaign	-0.007097586903728892
V2	job	-0.0031695068529891076
V6	balance	0.000999037108611094
V15	previous	-0.0008469677135408886
V10	day	-0.00040139392592551004
V11	month	0.00026497434033721513
V14	pdays	-2.6597524789551797e-06

The above table shows W values for all features. Similarly, we sort it according to the absolute value of W. In this model, the “credit” feature is the first feature. This means that the client’s default credit score is the most important feature to be considered in this model. The clients with higher credit score is more likely to subscribe the products.

The model has an accuracy of 88.90% for the training set, and an accuracy of 89.77% for the testing set. Compared to the best model of logistic regression, it has an lower training set accuracy but a higher testing set accuracy. Still, these accuracy values are relatively high, which means that the model is accurate as well.

Neural Network:

The best neural network model for this data set is the neural network with tanh activation function and no hidden layer, no regularization. Running this model on the testing set, we achieved an accuracy of 90%. The following is its classification report.

C=10^ -1

	precision	recall	f1-score	support
1	0.91	0.97	0.94	2632
2	0.65	0.33	0.43	368
accuracy			0.90	3000
macro avg	0.78	0.65	0.69	3000
weighted avg	0.88	0.90	0.88	3000

From our result table, we can see there was no much difference between no hidden layer, 1 hidden layer and 2 hidden layers, which indicated that our dataset was linearly seperatebale and didn't need specific transformation.

In addition, we noticed that there was no much difference between no regularization and L2 regularization, which implied that our training data didn't have overfitting problems.

What's more, from the graph, we noticed that the accuracy was a little lower when c=100, and 1000 than when c=0.1 and 1 among different activation functions. This showed that the dataset have small underfitting problems when c was large.

Across different hyperparameter C, different activation functions, different layers changing, the performance of the neural network were very similar, which is around 0.88-0.9 test accuracy. In addition, we used random sampling and shuffling when the data was split, which indicated that there are some features in dataset is unaccountable. Also there may be natural variability and uncertainty in our dataset which couldn't be trained by the neural network.

Improvements:

When we trained the dataset and tried different models with different settings, we discovered that the dataset is actually not balanced. Among all samples, only less than 12% of the clients subscribed to the product sold by the bank. Thus, even a model that predict all samples as class 1 (not subscribe) can have a 88% accuracy. In all the models we have trained, we found out that the accuracy for class 1 is much higher than the accuracy for class 2. For example, in the best model of neural network, the testing accuracy for class 1 is 91%, while the testing accuracy for class 2 is 65%.

An improvement we can probably do is that we can train all the 54,211 samples to get better models. Although the dataset is still unbalanced, training all the samples can give us a more balanced model although the overall accuracy might remain unchanged.

Works Cited:

Bank Marketing Dataset:
S. Moro, R. Laureano and P. Cortez. Using Data Mining for Bank Direct Marketing: An Application of the CRISP-DM Methodology. In P. Novais et al. (Eds.), Proceedings of the European Simulation and Modelling Conference - ESM'2011, pp. 117-121, Guimarães, Portugal, October, 2011. EUROSIS.
https://www.openml.org/search?type=data&status=active&sort=runs&id=1461

Sklearn Logistic Regression:
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Logistic Regression in Python:
https://realpython.com/logistic-regression-python/

Sklearn SVM:
https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

Neuron Networks in Python:
https://www.pluralsight.com/guides/machine-learning-neural-networks-scikit-learn