Report
Data Collected on different architectures

	Configurations	Accuracy		Precision		Recall		F1-Score	
		Train Set	**Test Set**	**Train Set**	**Test Set**	**Train Set**	**Test Set**	**Train Set**	**Test Set**
ANN-1	10x23, Sigmoid 23x21, Sigmoid 21x19, ReLU 19x17, ReLU 17x15, ReLU 15x13, Sigmoid 13x11, ReLU 11x9, Sigmoid 9x7, Sigmoid 7x5, ReLU 5x3, Sigmoid 3x1, Sigmoid Epochs = 150 Batch_Size=5 Scaling=yes	.87	.86	.80	.71	.41	.49	.54	.58
ANN-2	10x11, Sigmoid 11x11, Sigmoid 11x9, Sigmoid 9x7, Sigmoid 7x5, Sigmoid 5x3, Sigmoid 3x1, Sigmoid Epochs = 100 Batch_Size=2 Scaling=yes	.86	.85	.81	.71	.39	.47	.53	.57
ANN-3	10x60, ReLU 60x51, ReLU 51x42, ReLU 42x33, Sigmoid 33x24, ReLU 24x16, Sigmoid 16x10, Sigmoid 10x5, ReLU 5x3, Sigmoid 3x1, Sigmoid	.95	.79	.87	.51	.91	.58	.90	.54

	Epochs = 500 Batch_Size=10 Scaling=yes								
ANN-4	10x11, ReLU 11x11, ReLU 11x9, ReLU 9x7, ReLU 7x5, ReLU 5x3, ReLU 3x1, Sigmoid Epochs = 475 Batch_Size=8 Scaling=yes	.87	.82	.74	.59	.54	.56	.62	.57
ANN-5	10x120, Sigmoid 120x90, ReLU 90x54, Sigmoid 54x31, Sigmoid 31x25, ReLU 25x13, ReLU 13x8, Sigmoid 8x4, ReLU 4x2, Sigmoid 2x1, Sigmoid Epochs = 90 Batch_Size=20 Scaling=yes	.85	.84	.75	.66	.44	.52	.56	.58
ANN-6	10x302, Sigmoid 302x266, ReLU 266x241, Sigmoid 241x208, Sigmoid 208x180, ReLU 180x153, ReLU 153x122, Sigmoid 122x97, ReLU 97x51, Sigmoid 51x39, Sigmoid 39x22, ReLU 22x11, ReLU 11x9, Sigmoid 9x7, Sigmoid 7x6, ReLU 6x4, Sigmoid 4x3, Sigmoid	.80	.79	0	0	0	0	0	0

	3x1, Sigmoid Epochs = 140 Batch_Size=30 Scaling=yes								
ANN-7	10x7, Sigmoid 7x6, ReLU 6x5, ReLU 5x3, Sigmoid 3x1, ReLU Epochs=300 Batch_Size=20 Scaling=yes	.85	.83	.82	.70	.34	.40	.48	.51

For this assignment, I have created an artificial neural network sequential model using Keras. I have kept 10 inputs which are CreditScore, Geography, Gender, Age, Tenure, Balance, NumOfProducts, HasCrCard, IsActiveMember, and EstimatedSalary. I have removed the CustomerId and Surname because I think they don't have any correlation to whether the customer exited or not. I have tested different ANN models and I have found that ANN-1 is the best model used which has an average accuracy of 85%. This average accuracy was the highest out of all the ANN models I have tried.

With the 10 inputs, it will go into layer 1 containing 23 neurons using sigmoid and the kernel initializer is he_uniform. Layer 2 contains 21 neurons using sigmoid and the kernel initializer is glorot_uniform. Layer 3 contains 19 neurons using ReLU and the kernel initializer is glorot_uniform. Layer 4 contains 17 neurons using ReLU and the kernel initializer is he_uniform. Layer 5 contains 15 neurons using ReLU and the kernel initializer is he_uniform. Layer 6 contains 13 neurons using sigmoid and the kernel initializer is he_uniform. Layer 7 contains 11 neurons using ReLU and the kernel initializer is glorot_uniform. Layer 8 contains 9 neurons using sigmoid and the kernel initializer is glorot_uniform. Layer 9 contains 7 neurons using sigmoid and the kernel initializer is he_uniform. Layer 10 contains 5 neurons using ReLU and the kernel initializer is he_uniform. Layer 11 contains 3 neurons using sigmoid and the kernel initializer is glorot_uniform. The final layer which is layer 12 contains 1 neuron using sigmoid and the kernel initializer is glorot_uniform. The rest of the parameters that the layers in the sequential model that I did not mention above are on the default specifications.

For my .compile() function, I have set the optimizer to adam, the loss is set to binary_crossentropy, and the metrics will have accuracy, precision, and recall. This compile function will then output the training and validation information for loss, accuracy, precision, and recall. I have used a .fit() function to train my model. I have trained my model on the dataset.csv file by splitting the data into 80% training set and 20% testing set. I have passed my x_train scaler as x and my y_train as y. I set the batch_size to 5 and have an epoch at 75. I chose

these parameters for my .fit() function because I noticed that when you have an extremely high epoch, it does not improve the accuracy from what my model has shown me, but if the epoch is too low, the accuracy suffers. I have also chosen the batch size to be 5. All the other parameters on my .fit() function that I did not mention above is on the default specifications the .fit() function comes with. My .evaluate() function will evaluate and output the accuracy, precision, recall, and f1-score of the training set. I will then use the model that I trained and test it on the 20% test set. I will predict whether the customers will exit or not exit on the test set and compare my predicted answers with the actual answers. I will then use the classification report from the imported files and pass the predicted answers and the actual answers. This will then calculate the accuracy, recall, precision, and the f1-score of our test set.

The highest accuracy I have gotten was with ANN-1 with the highest accuracy score being 87%. This is a pretty high accuracy score in my opinion. When I run the ANN-1 multiple times, I will usually have an accuracy in the range of 83% to 87%. I am very happy with this accuracy range so I have decided to run my ANN-1 on the judge.csv file provided. I have removed the CustomerId and the Surname because as I said before, I think those two have no correlation to whether the customers exited or not. I have changed the geography, and gender from strings into integers so it is easier to pass the data from the file into my ANN-1 model. Using the same 10 inputs I used to train my model, I pass the judge.csv dataset into my model to have it predict whether or not the customers will exit. Once my trained model has predicted whether or not the customer will exit on the judge.csv file, I have taken my predicted results and stored it in one dataframe. I have stored the CustomerId on another dataframe. I have then merged the two dataframes together and stored the merged dataframe onto a new csv file called judge-pred.csv which contains the CustomerId and the predicted answers to whether or not the customers exited.

I have built my code using Google Collaboratory which utilizes two files. One file named “train_code.ipynb” will train our artificial neural network model using dataset.csv and will output the train set and test set accuracy, precision, recall, and f1-score using the .evaluate function. It will then save the newly trained ANN model into a json file and the weights of the model into a h5 file. The other file named “test_code.ipynb” will take the judge.csv file and predict if the customers exit or not using the trained ANN model from the “training_code.ipynb”. The “test_code.ipynb” will load both the json and the h5 file, compile it and run our loaded ANN model on the judge.csv file. This will predict whether the customers exited or not and create a new file called judge-pred.csv file containing only CustomerId and the predicted exited.