Sujeeth Shetty 04/12/2020

BUAN 6341 Applied Machine Learning

ASSIGNMENT 3

Implementation of Artificial Neural Network & k-NN

Introduction:

The objective of this project is to implement Artificial Neural Network & k-nearest neighbors to perform binary classification on two Datasets. This report details the various experiments conducted using the two datasets to understand the effect of the cross-validation and various hyperparameters. It also shows the effect of the use of various independent variables/features in selecting the best model.

About the Data:

The first data set is of SGEMM GPU kernel performance which consists of 14 features and 241600 records. This data set measures the running time of a matrix-matrix product A*B = C, where all matrices have size 2048 x 2048, using a parameterizable SGEMM GPU kernel with 261400 possible parameter combinations. Out of 14 features, the first 10 are ordinal and can only take up to 4 different powers of two values, and the 4 last variables are binary.

The second dataset is Bank Marketing UCI Dataset published by Banco de Portugal which consists of 20 features and 45307 records. This dataset is used to measures the success of Bank Telemarketing where the binary classification goal is to predict if the client will subscribe to a bank term deposit. There are few missing values, and all are coded with the “unknown” label. These missing values are treated as a possible class label. One of the reasons for selecting this dataset is because of the imbalance in class distribution. This would help in understanding the caveats when trying to leverage cross-validation, bias associated with classification output and risk associated with false-negative or false-positive predictions.

Project Outline:

Algorithm Implementation

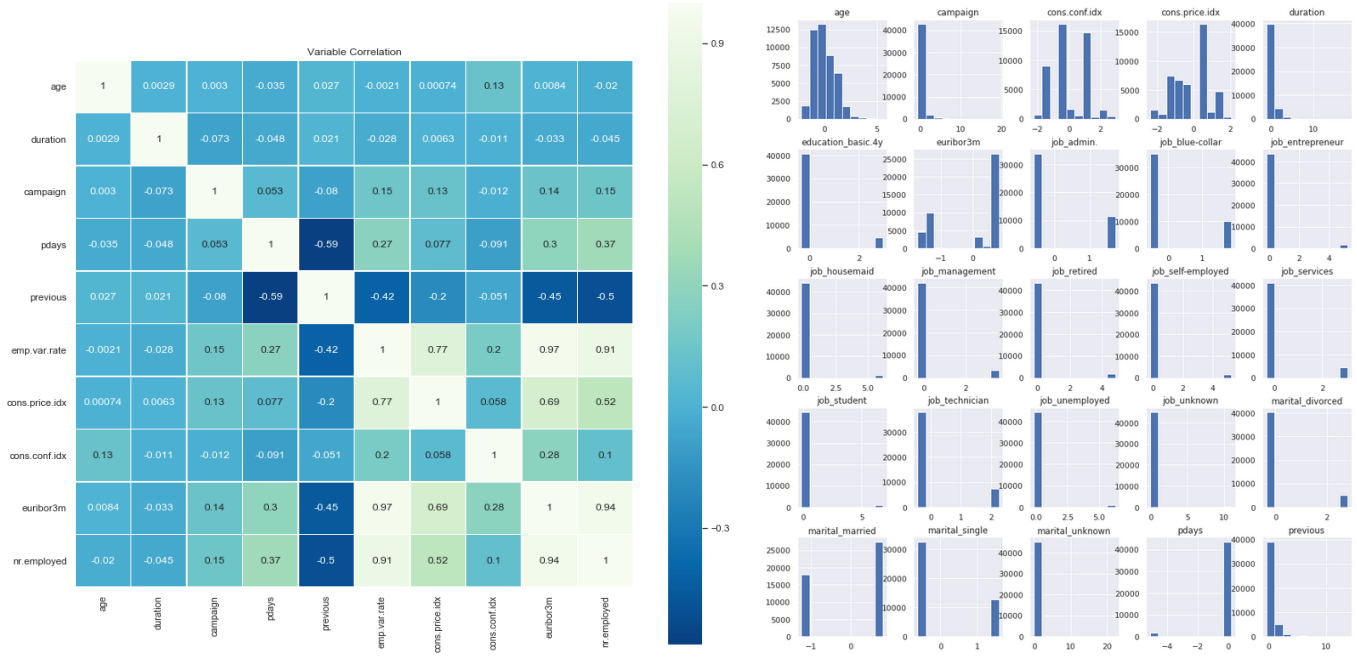
The Neural Network Sequential model is implemented using python package “Keras” and k-NN is implemented using “scikit-learn”. Keras is a powerful and easy-to-use free open-source Python library for developing and evaluating deep learning models. It wraps the efficient numerical computation libraries Theano and TensorFlow and allows us to define and train ANN models in just a few lines of code. On the other hand, Scikit-learn is a very popular package because of its ease of use. It provides a layer of abstraction on top of Python. Therefore, to make use of the k-NN algorithm, it’s enough to create an instance of KNeighborsClassifier. By default, the KNeighborsClassifier looks for the 5 nearest neighbors.

The learning algorithms are implemented for two different binary classification datasets. The algorithms are implemented with options to change the hyperparameters: kernel, penalty param, gamma, max tree depth, metric to split on variables.

Data Preparation and Exploratory Data Analysis

For **SGEMM GPU Kernel Performance Dataset**, the the runtime was converted into a classification problem by classifying all values above the mean runtime as 1 and value below as 0 . By this classification, we are trying to predict when the GPU runtime is high. Scaling was performed on the features to speed up the networks learning and faster convergence.

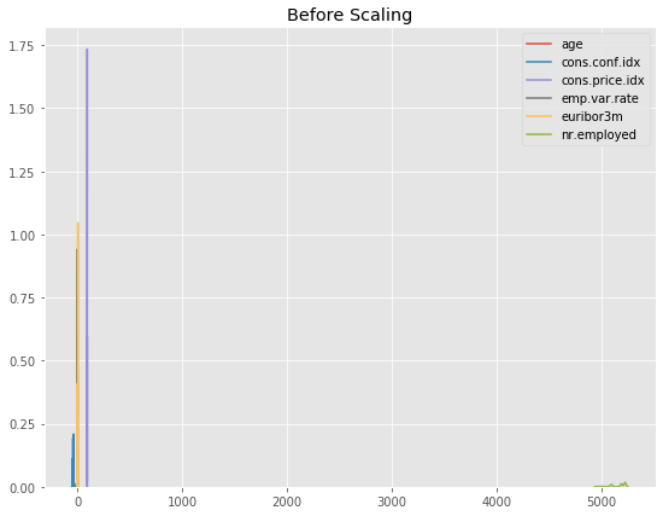
The objective of **Bank Marketing UCI Dataset** is to correctly predict if client will subscribe a bank term deposited. The correlation plot indicated high correlation between employment variation rate, euribor 3-month rate and number of employees(nr.employeed). To deal with multicollinearity, the variables nr.employeed & emp.var.rate are removed from the dataset.

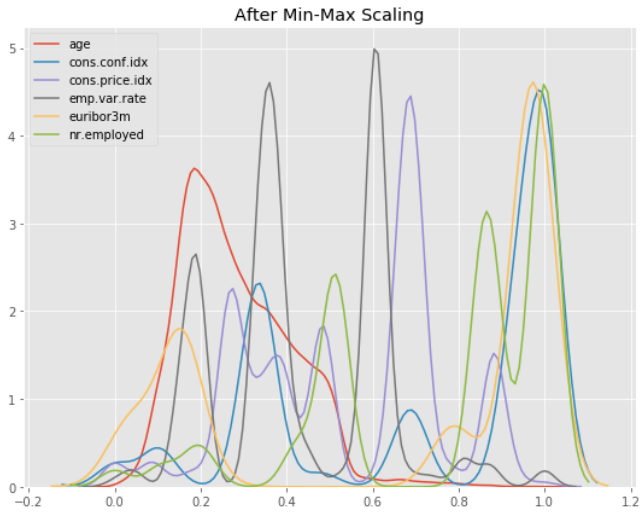


The target variable ‘y’ has imbalanced in class distribution. The dataset has only 5091 of positive instance and rest 40216 are of negative class.

```
y
0    40216
1     5091
dtype: int64
```

The algorithm performs much better when all the numeric features have a similar scale. So, all these features standardized using Min-Max scaling. To improve the model performance, one hot encoding was performed on all categorical variable and label encoding was performed on month & day_of_week.





Experimentation and Results

Two tasks were undertaken in which we ran the classification problem on Sequential Neural Network & k-NN. At least two experiments were conducted on each learning algorithms where we explored the effect of adding new layers, increasing/decreasing epochs, and discuss the effectiveness of feature selection in predictive accuracy/precision. These tasks & experiments were repeated both for SGEMM GPU Kernel Performance Dataset and Bank Marketing Dataset. Finally, the results of the experiments were discussed in the results section and further improvement opportunities were discussed.

Tasks & Experimentation:

1 Artificial Neural Network

1.1 SGEMM GPU Kernel Performance Dataset

Initially, the data was split into Train & Test with ratio of 80:20. The experimentation was performed between two layers sequential network & 3 layers sequential network. The first model has 14 inputs, 1 hidden layer with 100 neurons and output layer with two output. Rectified linear activation functions are used in each hidden layer and a sigmoid activation function is used in the output layer, for binary classification. In the second model an additional hidden layer with 50 neurons were added.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 100)               1500
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 202
=================================================================
Total params: 1,702
Trainable params: 1,702
Non-trainable params: 0
_________________________________________________________________
None
```

Figure 1 Model with 2 layers

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 100)               1500
_________________________________________________________________
dense_4 (Dense)              (None, 50)                5050
_________________________________________________________________
dense_5 (Dense)              (None, 2)                 102
=================================================================
Total params: 6,652
Trainable params: 6,652
Non-trainable params: 0
_________________________________________________________________
None
```

Figure 2 Model with 3 layers

The two models trained for 300 & 200 epochs respectively. The results of the two models are depicted below. By

looking at the loss vs epochs & accuracy vs epochs plot, we can conclude that Model 2(3 layers) performed better than the rest. The second model showed an accuracy of 99.57% on test dataset & low validation loss of 1.56%.

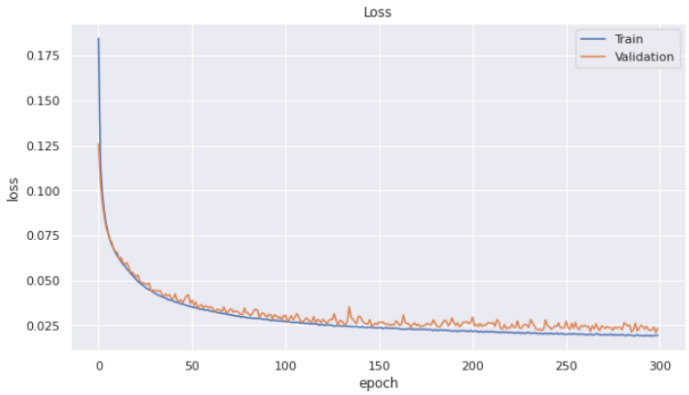


Figure 3 Model1 - Loss during training

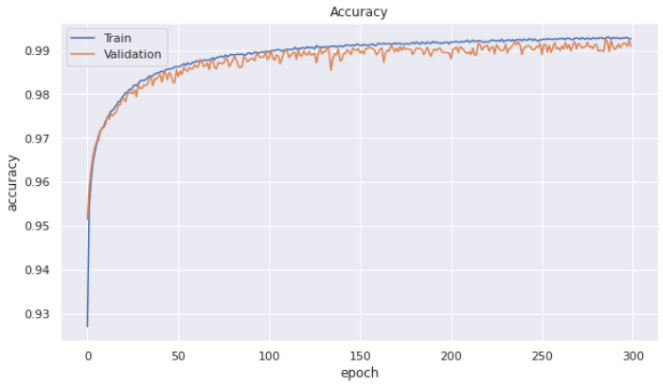


Figure 4 Model1 - Accuracy during training

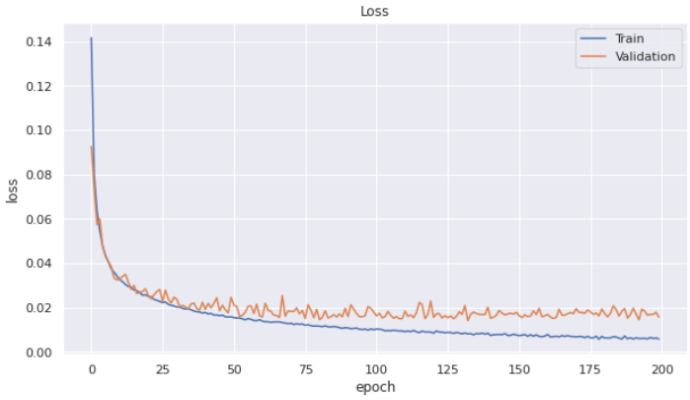


Figure 5 Mode2 - Loss during training

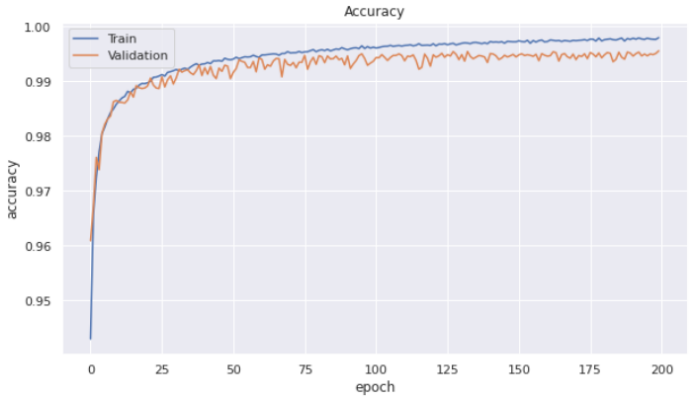


Figure 6 Model2 - Accuracy during training

Model	Model1(2 Layers)	Model2(3 layers)
Train Accuracy	99.26	99.80
Validation Accuracy	99.09	99.56
Test Accuracy	99.08	99.57
Train Loss	0.0194	0.0058
Validation Loss	0.0235	0.0156

1.2 Bank Marketing UCI Dataset

Two models, one with 2 layers and another with 3 layers were trained with Bank Marketing Dataset. The data was split into Train & Test with a ratio of 80:20. Out of 80% train data, 20% is used for validation while training the model. The number of epochs used to train the two models is 300 & 200 respectively. Both models gave out a similar performance since the class distribution is highly skewed both the model suffered in classifying positive class(1). The first model(1 layer) was slightly better in classifying positive class with precision 57.78% compared to 57.51%.

```
[[7638  407]
 [ 460  557]]
```

```
[[7672  373]
 [ 512  505]]
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 100)               1500
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 202
=================================================================
Total params: 1,702
Trainable params: 1,702
Non-trainable params: 0
_________________________________________________________________
None
```

Figure 7 Model1 – 2 layers

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 100)               1500
_________________________________________________________________
dense_4 (Dense)              (None, 50)                5050
_________________________________________________________________
dense_5 (Dense)              (None, 2)                 102
=================================================================
Total params: 6,652
Trainable params: 6,652
Non-trainable params: 0
_________________________________________________________________
None
```

Figure 8 Model2 – 3 layers

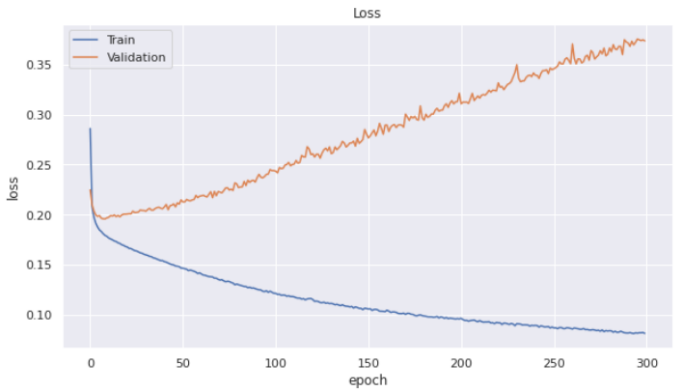


Figure 9 Model1 – Loss during training

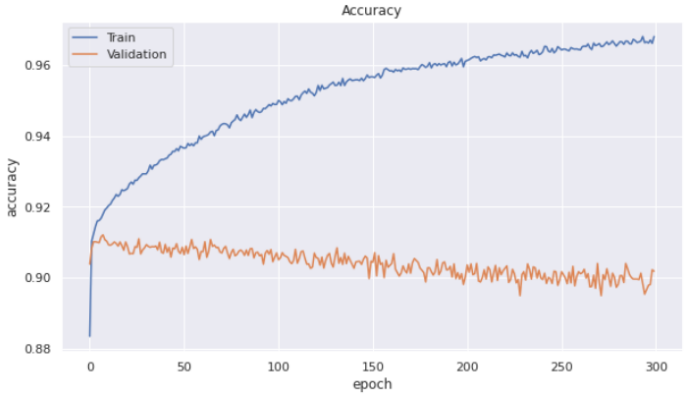


Figure 10 Model1 – Accuracy during training

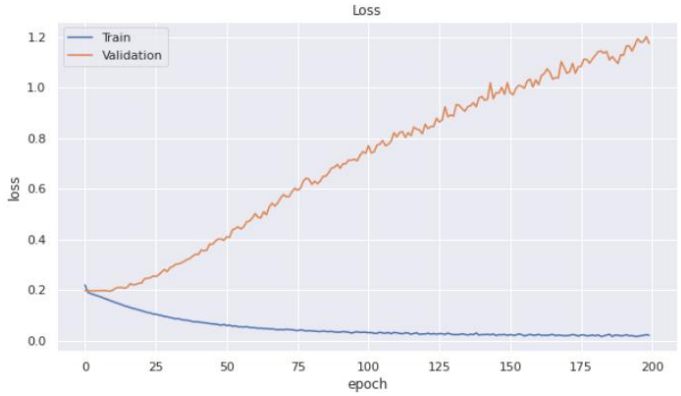


Figure 11 Model2 – Loss during training

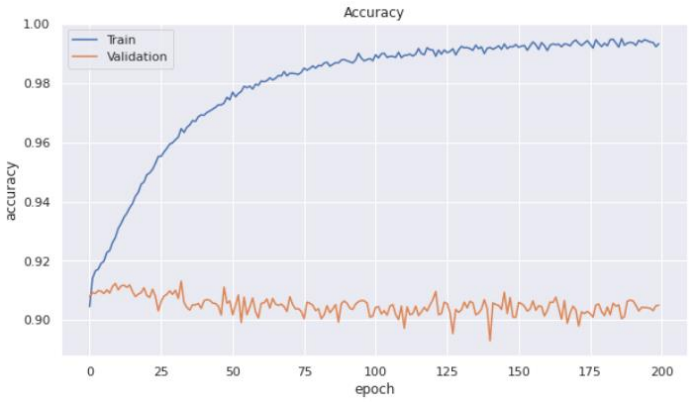


Figure 12 Model2 – Accuracy during training

Model	Model1(2 Layers)	Model2(3 layers)
Train Accuracy	96.82	99.34
Validation Accuracy	90.18	90.50
Test Accuracy	90.23	90.43
Train Loss	0.0817	0.0208
Validation Loss	0.9018	1.17

To deal with severely skewed class distribution(100/12), stratified 5-fold cross-validation was performed on the dataset and trained the 3-layered model(Model2) with 200 epochs. The output was slightly better than the above models with a test accuracy of 91.69% and a loss of 0.082.

2 K Nearest Neighbors

2.1 SGEMM GPU Kernel Performance Dataset

The first step of the experiment was to identify the best K value. Here the data was split into train & test with a ratio of 80:20. A k value ranging from (1,25) was passed to model and the Minkowski distance metric was used with a value of p as 2 i.e. k-NN classifier will use Euclidean Distance Metric formula. The Accuracy vs k value was plotted and from the below plot we can conclude that k=3,5 had the best accuracy.

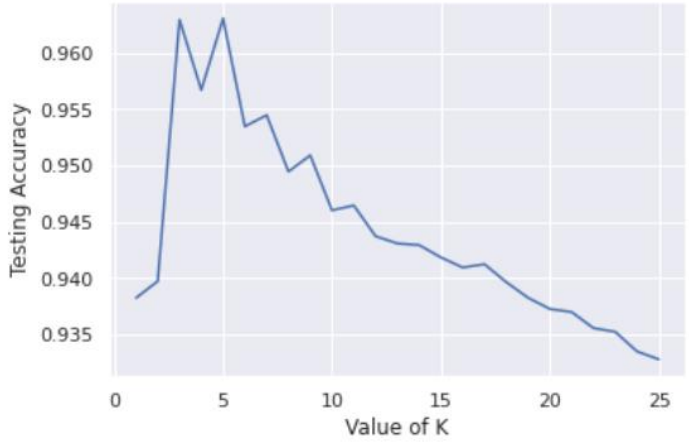


Figure 13 Finding best K

```
[[34922   755]
 [ 1031 11612]]
              precision    recall  f1-score   support

           0       0.97      0.98      0.98     35677
           1       0.94      0.92      0.93     12643

    accuracy                           0.96     48320
   macro avg       0.96      0.95      0.95     48320
weighted avg       0.96      0.96      0.96     48320
```

Figure 14 Confusion Matrix - k=5

To perform the prediction, the k-NN model was trained with k=5 and Euclidian distance. The test accuracy was overall 96% and performing k-fold cross validation gave the similar result.

```
[[34922   755]
 [ 1031 11612]]
              precision    recall  f1-score   support

           0       0.97      0.98      0.98     35677
           1       0.94      0.92      0.93     12643

    accuracy                           0.96     48320
   macro avg       0.96      0.95      0.95     48320
weighted avg       0.96      0.96      0.96     48320
```

Figure 15 Confusion Matrix - kfold -knn

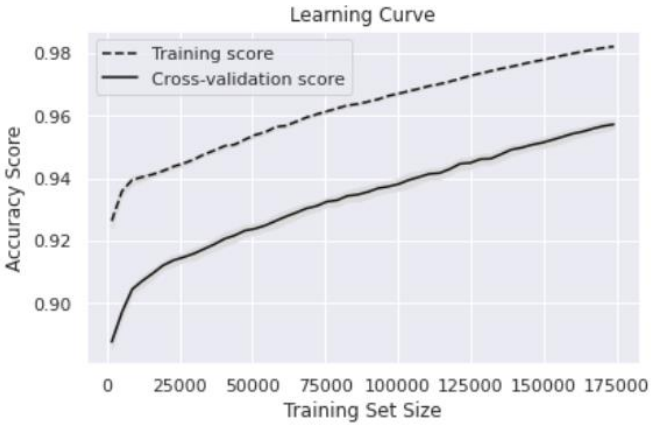


Figure 16 Learning Curve - Kfold – knn

2.2 Bank Marketing UCI Dataset

The goal of this experiment is to appropriately identify the class of interest 1(whether customers will opt for bank term plan), so to measure the performance we will be Precision rather than Accuracy.

The first step was to identify the number of neighbor's value, here we found the best value to be k=9.

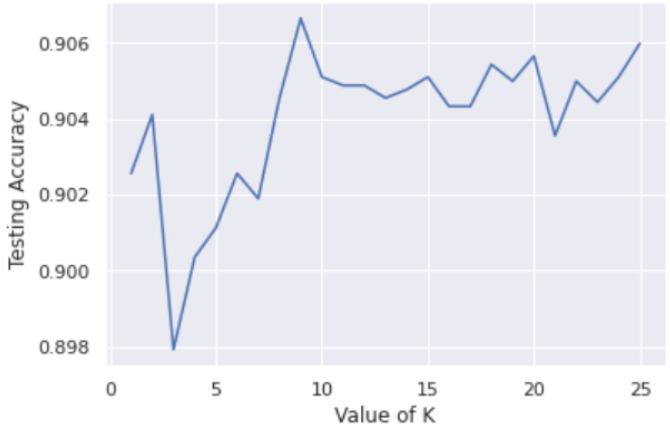


Figure 17 Find best K

By default, the k-NN considers uniform weights where all points in each neighborhood are weighted equally. The k-NN was first run with uniform weights and the second run had distance weights(closer neighbors will have greater influence). As you can see below, the distance weights model performed better with 70% precision(test data) in classifying class of interest.

```
[[7814  231]
 [ 615  402]]
              precision    recall  f1-score   support

           0       0.93      0.97      0.95      8045
           1       0.64      0.40      0.49      1017

    accuracy                           0.91      9062
   macro avg       0.78      0.68      0.72      9062
weighted avg       0.89      0.91      0.90      9062
```

Figure 18 Confusion Matrix - uniform weights

```
[[7841  204]
 [ 538  479]]
              precision    recall  f1-score   support

           0       0.94      0.97      0.95      8045
           1       0.70      0.47      0.56      1017

    accuracy                           0.92      9062
   macro avg       0.82      0.72      0.76      9062
weighted avg       0.91      0.92      0.91      9062
```

Figure 19 Confusion Matrix - nearest distance weights

In the final experiment, a stratified k-fold was performed on the data before running k-NN. As you can see below, the test performance wasn't better compared to the distance weights model.

```
[[7739  306]
 [ 590  427]]
              precision    recall  f1-score   support

           0       0.93      0.96      0.95      8045
           1       0.58      0.42      0.49      1017

    accuracy                           0.90      9062
   macro avg       0.76      0.69      0.72      9062
weighted avg       0.89      0.90      0.89      9062
```

Figure 20 Confusion Matrix - stratified kfold

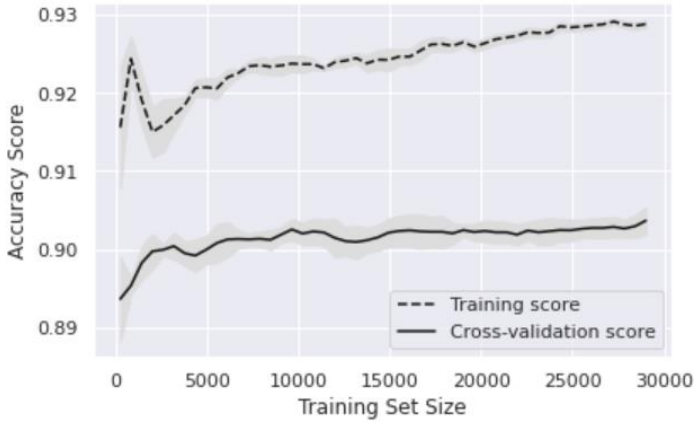


Figure 21 Learning Curve - kfold

Results

Through experimentation, we found that input from multiple models can provide new and unique insight that can help Data Scientists to understand various properties of the data. We also found that hyperparameters tuning like k neighbors, weights, metrics is effective for better accuracy of the classification using k-NN. In terms of ANN, adding a new layer with multiple neurons gave a better result. Looking at the performance of the modes, for the **SGEMM GPU Kernel Dataset**, Sequential Neural Network gave a better accuracy of 99.57%. On the other hand, for **Bank Marketing UCI Dataset,** k-NN was a better performing model with a test accuracy of 92% and a precision of 70%. This concludes that single learning cannot be used to learn datasets of various domains.

Comparing the current performance with SVM, Decision Tree, Boosting & Logistic Regression; Sequential Neural Network was the best learning algorithm for **SGEMM GPU Kernel Dataset** and k-NN for **Bank Marketing UCI Dataset.**

Also, we saw the importance of feature selection and engineering to improve the performance of the ML algorithm. The best predictors for **the SGEMM GPU Kernel** dataset are MWG, NWG, MDIMC, NDIMC, VWM, VWN. The per-matrix 2D tiling at the workgroup level, local workgroup size & local memory shape might reduce the GPU runtime. In the case of **Bank Marketing**, the duration of call had a significant impact on classifying subscription of term deposit by the client. One important thing note here is that this attribute highly affects the output target (e.g., if duration=0 then y="no"). Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes and in the future, it will be discarded while building a realistic predictive model.

There is a potential to improve the performance of the ANN model by dealing with an imbalanced **Bank Marketing Dataset**. Here, using Accuracy to measure the goodness of the model which classifies all testing samples into “0” will have an excellent accuracy of 92%, but this model won’t provide any valuable information. To get a better insight, alternative evaluation metrics like Precision & Recall were used. One of the techniques used to deal with this is Stratified K-fold which ensures each class is(approximately) equally represented across each test fold. But this further deteriorated the test performance. Additional techniques like Oversampling & Under sampling of data can be tried in the future to increase the precision of the ANN model.

Citation

[Moro et al., 2014] S. Moro, P. Cortez and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, In press, http://dx.doi.org/10.1016/j.dss.2014.03.001

Rafael Ballester-Ripoll, Enrique G. Paredes, Renato Pajarola. Sobol Tensor Trains for Global Sensitivity Analysis. In arXiv Computer Science / Numerical Analysis e-prints, 2017 (https://128.84.21.199/abs/1712.00233)

Cedric Nugteren and Valeriu Codreanu. CLTune: A Generic Auto-Tuner for OpenCL Kernels. In: MCSoC: 9th International Symposium on Embedded Multicore/Many-core Systems-on-Chip. IEEE, 2015 (http://ieeexplore.ieee.org/document/7328205/)