Assignment 3

Data Description

Dataset 1: Appliances Energy (https://archive.ics.uci.edu/ml/datasets/Appliances+energy+prediction)

We have cross-sectional dataset with 19735 observations and 29 variables. The variables are Appliances and lights energy consumption in the house, 9 temperatures and 9 relative humidity in 9 different locations in the house, 6 weather variables such as outside temperature, outside humidity, Pressure, Windspeed, Visibility and Dewpoint. We also have date column, and two random variables that are being ignored in this model. Additional information about the dataset can be found out in the above-mentioned link.

Dependent variable is Appliances which is categorized as binary energy high variable(1 if energy is greater than median) and Date variable is dropped from the dataset for convenience

None of the variables consists of missing attributes so no missing value imputation is required

Data is partitioned randomly into train and test using 70:30 split percentage(13814 and 5921 observations respectively)

All the features are scaled to the range [0,1] by subtracting each feature from its mean and dividing by standard deviation in order to maintain evenness in the data

Dataset 2: German Credit Data (Attached)

We have cross-sectional dataset with 1000 observations and 21 variables(7 numerical and 14 categorical). Target variable is Default which is renamed as Creditability (binary, 1=Good, 0=bad).

None of the attributes have missing values so no missing value imputation is required

The numerical attributes are rescaled(min-max normalization) and for categorical attributes OneHotEncoding is used to create the dummy variables. The dataset now has 1000 observations and 62 attributes.

Data is partitioned randomly into train and test using 70:30 split percentage(700 train and 300 test observations respectively)

Class imbalance exists in the data. Synthetic Minority Oversampling Technique(SMOTE) is used to balance the data after which there are 972 observations in the train data. All the algorithms performed better after the class imbalance is taken care of.

This dataset is chosen given the number of categorical variables it has for analyzing whether an applicant is creditable or not. The dataset provides a credit simulation close to the realistic credit simulation in the modern credit analyses with additional variables like the criminal records of applicants, their health information etc. This is similar to what is done by a bank before a credit is approved.

Modern credit analyses employ many additional variables like the criminal records of applicants, their health information, net balance between monthly income and expenses. A dataset with these variables could be acquired or complementary variables added to the dataset. This will make the credit simulations much realistic, similar to what is done by the banks before a credit is approved.

ROC Curves are used to evaluate the performance whereas algorithm learning is observed using Learning Curves

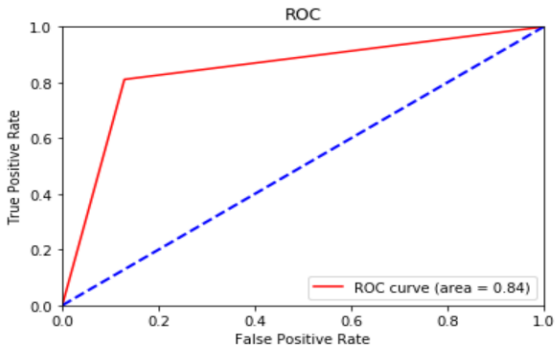
Cross-Validation is implemented to calculate the prediction accuracies of each of the algorithms

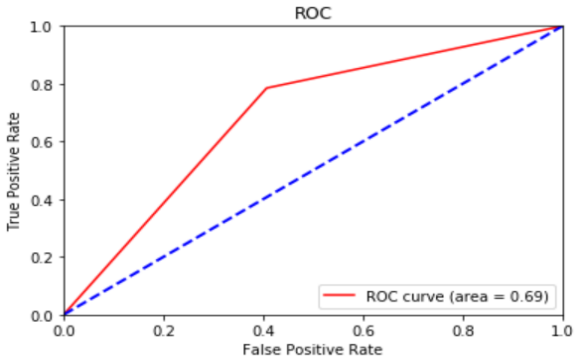
Randomized search and Grid Search are used for parameter tuning for each of the algorithms

Implementation using various Algorithms

Artificial Neural Network – Feed Forward MLP

ROC Curves





Appliances

AUC_ROC: 0.84

German Credit

AUC_ROC: 0.69

Appliances: Accuracy = 84% , Train Error = 12.4%, Test Error = 15.7%

Confusion Matrix:
```
[[2857  326]
 [ 436 2302]]
```
(2857+2302=5159) observations are classified correctly whereas (436+326 = 762) observations are misclassified

German: Accuracy = 85%, Train Error = 0%, Test Error: 27%

Confusion Matrix:
```
[[ 64  22]
 [ 90 124]]
```
188 observations are classified correctly whereas 112 observations are misclassified

Parameter Tuning

Appliances

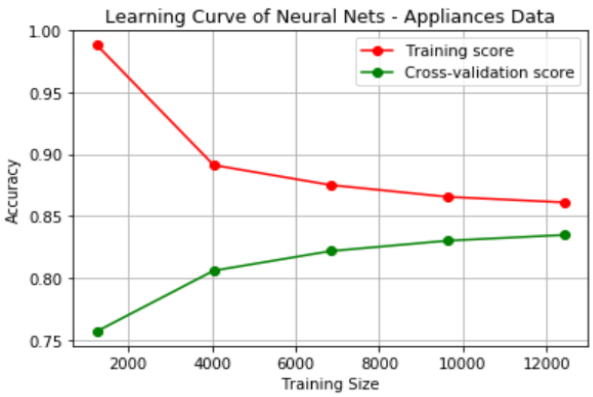
```
<bound method BaseEstimator.get_params of MLPClassifier(activation='relu', alpha=5, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(75, 75), learning_rate='constant',
              learning_rate_init=0.001, max_iter=200, momentum=0.9,
              n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
              random_state=None, shuffle=True, solver='lbfgs', tol=0.0001,
              validation_fraction=0.1, verbose=False, warm_start=False)>
```

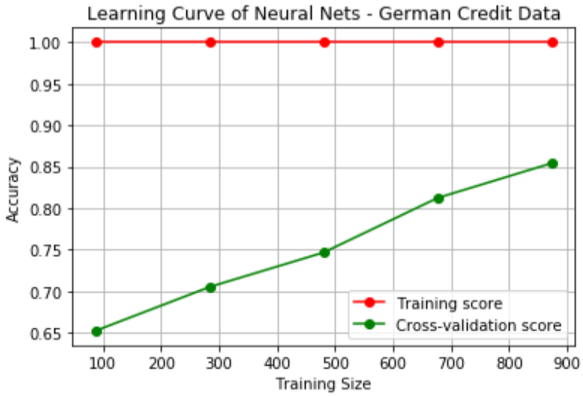
German Credit

```
Best parameters:
 {'activation': 'relu', 'alpha': 0.01, 'hidden_layer_sizes': (100, 100), 'learning_rate': 'constant', 'solver': 'adam'}
Results on the test set:
              precision    recall  f1-score   support

           0       0.58      0.59      0.59        86
           1       0.83      0.83      0.83       214

    accuracy                           0.76       300
   macro avg       0.71      0.71      0.71       300
weighted avg       0.76      0.76      0.76       300
```

The best parameters for both the Appliances data and German Credit data are obtained when relu is used as activation.

Learning Curves





Appliances: When the observations are less, high bias is evident in the graph. This might be due to under fitting and as the number of observations increases the bias problem no longer exists but the gap between the train and test curves suggests high variation in the train and test data. This might be due to overfitting in the data. Train accuracy decreases as the number of observations increases whereas the test accuracy increases as the number of observations increases and they come close with very large number of observations with better fit of the model due to low bias and low variance. The test accuracy settled at 84%.

German Credit: The training curve has the constant value of 1 which might be due to overfitting in the data resulting in zero error rate which is not as expected. The test curve shows that the model performs very badly when there are a smaller number of observations and the accuracy started increasing as the number of observations started increasing and settles at an accuracy of 85%. There is still a huge gap between the train and test curves which suggests that there is high variation in the train and test data. This problem can be solved when there are more number of observations are there in the dataset leading to better interpretations.

ANN – Tensor Flow Keras

Appliances

After 250 iterations, the accuracy is 78.15%

Train Loss: 0.4558, Train Accuracy: 0.7867

Test Loss: 0.4767 Test Accuracy: 0.7782

```
Confusion Matrix:
 [[2602  581]
 [ 732 2006]]
```

German Credit

After 250 iterations, the accuracy is 76.53%

Train Loss: 0.4602, Train Accuracy: 0.7716

Test Loss: 0.5577 Test Accuracy: 0.7133

```
Confusion Matrix:
  [[ 59  27]
 [ 59 155]]
```

<u>Parameter Tuning</u>

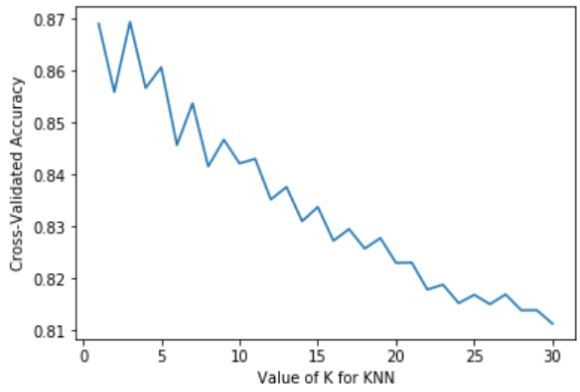
```
Best: 0.813786 using {'optimizer': 'Adam', 'momentum': 0.5, 'learn_rate': 0.1, 'epochs': 250, 'batch_size': 64, 'activatio
n': 'sigmoid'}
```

Keras models can be used in the sci-kit learn by wrapping them with the KerasClassifier. To use these wrappers a function is defined that creates and returns Keras sequential model, then the function is passed to the build_fn argument when constructing the KerasClassifier class. The constructor for the **KerasClassifier** class can take default arguments that are passed on to the calls to the model fit such as number of epochs, batch size, activations, optimizers, learning rate, momentum etc.

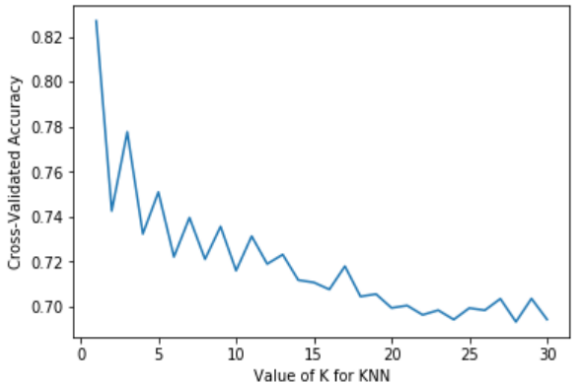
Hyperparameter Optimization which is big part of deep learning, is used to tune the parameters and the best accuracy obtained for German Credit dataset is 81.38% which is significantly higher when compared to the previously obtained accuracy 76.53%.

KNN

Elbow Graphs



Appliances – Elbow at k = 3



German Credit – Elbow at k = 3

Best Parameters after Parameter Tuning

Appliances

```
0.8715795569711886
s': 'distance', 'n_neighbors': 10}
rsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=10, p=2,
                     weights='distance')
```

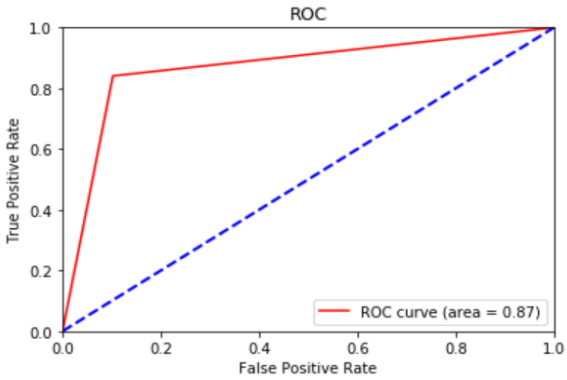
German Credit

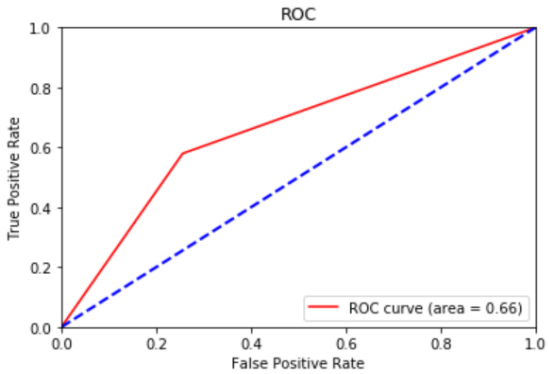
```
0.7685185185185185
{'weights': 'distance', 'n_neighbors': 10}
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=10, p=2,
                     weights='distance')
```

From the Elbow graph we have k=3 as the best choice of k and from the parameter tuning k=10 is the best choice of k. When modeled with both k=3 and 10 there is not much difference in the accuracies. So k=3 is chosen in order to reduce the complexity.

Of the two types of weights, uniform and distance the parameter tuning chose distance as the best measure and instead of the usual euclidean distance measure, minkowski distance is chosen for the best results. Appliances data performed very well using KNN algorithm when compared to the Germa Credit data. One of the reasons for this difference in the accuracies might be the small number of observations in the German Credit data when compared to the Appliances data.

ROC Curves





Appliances

ROC_AUC:0.87

German Credit

ROC_AUC:0.66

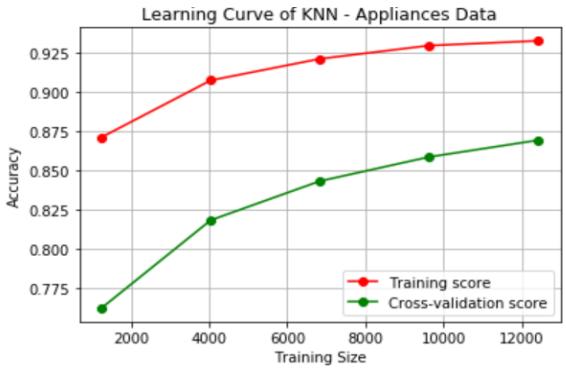
Appliances: Accuracy = 87% , Train Error = 6.5%, Test Error = 12.86%

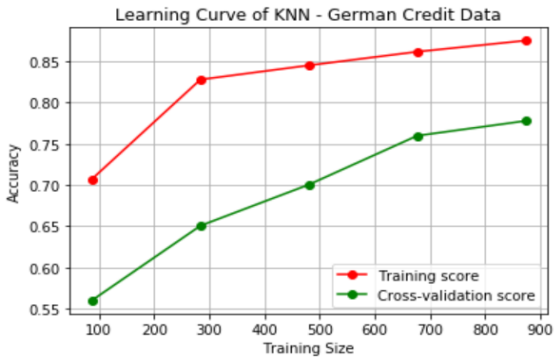
Confusion Matrix:
```
[[2857  326]
 [ 436 2302]]
```
(2857+2302=5159) observations are classified correctly whereas (436+326 = 762) observations are misclassified

German: Accuracy = 78%, Train Error = 12.24%, Test Error: 37.33%

Confusion Matrix:
```
[[ 64  22]
 [ 90 124]]
```
188 observations are classified correctly whereas 112 observations are misclassified

Learning Curves





Appliances: Both the training and test scores have significantly improved as the number of observations increased and the gap between the test and train scores in the graph implies the high variance in the data which might be due to overfitting. There is no bias visible but the variation in the data. As the number of observations increased both the train and test data performed better on the dataset and the test accuracy score of 87.15% is the best accuracy score among all the algorithms used so far

German: Training and test accuracy scores improved with the number of observations but both the values are low when compared to the algorithms used earlier. This might be due to the less number of observations in the dataset. Again, the gap between the train and test scores indicate high variation in the train and test data which might be due to overfitting.

Conclusion:

Appliances

The performance of the three algorithms achieved an average accuracy of 83%. KNN algorithm achieved maximum accuracy of 87.15% which is slightly higher than the accuracy obtained with Random Forest with 87% after modeling. Of all the algorithms, KNN turned in the top result using the training data. It achieved an average accuracy of 87.15%, followed by Random Forest with 87%, Feed forward MLP with 85% and Decision Tree with GINI index with 83%. KNN algorithm has the area under ROC curve 0.87 which is the highest of all the other algorithms followed by Random Forest with arear under ROC curve 0.86.

From the model-building activities, the KNN and Random Forest ensemble algorithms yielded the top-notch training and validation results. These two are the recommended algorithms to use from the accuracy perspective.

German Credit

The performance of the three algorithms achieved an average accuracy of 81%. Feed forward MLP achieved the maximum accuracy of 85% after modeling. Of all the algorithms used, MLP and keras turned in the top result using the training data. They achieved an average accuracy of 85% and 81.38% respectively, followed by Random forest and Gradient Boosting with 81%, followed by AdaBoost with 79%. Logistic Regression, Random Forest, Gradient Boosting and AdaBoost algorithms have the area under ROC curve 0.71 which is the highest of all the other algorithms. The low score on area is result of the size of the data.

From the model-building activities, for the data with a greater number of observations, KNN yielded the best results whereas Random Forest and Gradient Boosting ensemble algorithms yielded the consistent training and validation results along with small training and test error rates which implies very less misclassification. These two are the recommended algorithms to use from the accuracy perspective.

Appliances data showed better results compared to German Credit in terms of prediction accuracies because for the Appliances data 10 best features are selected based on the correlations with the target and the German Credit Data is modelled using all the 62 features. Class imbalance has been taken care of, but the number of features caused a problem in the prediction accuracies. Dimension Reduction algorithms such as PCA can be used to select the best features.