# **Introduction:**

The data set obtained is to classify the intention of a shopper whether the product is purchased or not based on the various online shopping parameters. Through this assignment, we would be able to classify the outcome performance of each combination using SVM, Decision Trees, Boosting, cross validation techniques, Artificial Neural Networks and KNN algorithm. By using these classification algorithms, we model the shoppers intention from various parameters.

# **DataSet:**

The Dataset is downloaded from the UCI machine learning repository. A brief description of the dataset is as follows:

1. It contains 12330 records on 18 attributes.
2. Dependent Variable is Revenue
3. Missing values in the dataset were removed as they constituted only 1% of the data.

# **Data Preparation and Exploratory Analysis:**

1. Imported the Dataset and analyzed the various parameters to figure out their distribution.
2. Performed label encoding for the outcome variable as 0 and 1.
3. Data Source <https://www.kaggle.com/roshansharma/online-shoppers-intention>
4. Used the train\_test\_split function to split the data into training (80%) set and test (20%) set.
5. The dataset consists of 18 features with categorical variables. I performed one-hot encoding through the dummies() function to convert the dataset to numerical values.

The goal of the model is to classify the shopper intention and the various classification algorithms gives us the information on how the function learns from the data. For measuring the performance of the algorithm, I have used ROC curves, classification matrix and area under curve parameters.

# **Support Vector Machines**

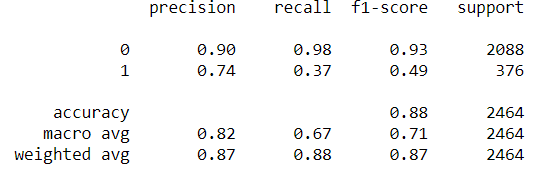
I have used the SVC package from sklearn. It provides with options to change the kernel functions, C and Gamma value. I have performed modelling using 3 kernel functions – linear, RBF (gaussian) and polynomial. The results and interpretation are given below.

I have used the SVC package from sklearn with options to change the kernel functions, C and Gamma value. I have performed modelling using 3 kernel functions – linear, RBF (gaussian) and sigmoid. The results and interpretation are given below.

**Linear Kernel:**

Implemented the linear kernel package using the training dataset. Obtained the optimized value for the kernel with value of C as 1. By using the test data, obtained an accuracy of 88.4% for the default case.

The classification report is as below:

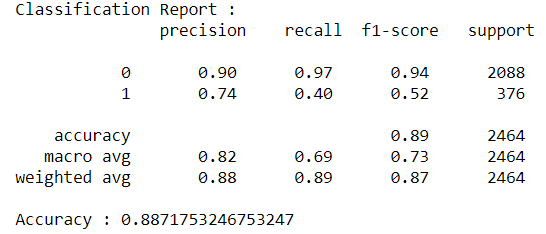


We can interpret the precision of the dataset to be 0.87 (weighted average). Following confusion matrix gives classification rates for this kernel.

Here we can interpret that the model performs good as we have a high precision and high recall.

**RBF kernel:**

Implemented the RBF kernel package using the training dataset. Used GridCv to obtain the optimized value for the kernel with value of C as 1 and gamma as 0.01. By using the test data, obtained an accuracy of 89% for the default case.

The classification report is as below:

We can interpret the precision of the dataset to be 0.88 (weighted average) with an accuracy of 89%. The accuracy of the model is slightly higher than Linear kernel model.

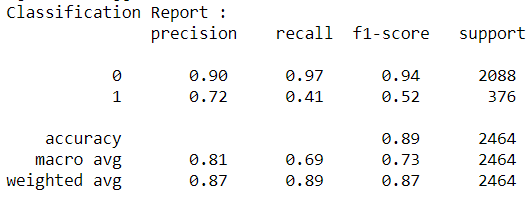
Following confusion matrix gives classification rates for this kernel.

Here we can interpret that the model performs good as we have a high precision and high recall. The precision and recall values are slightly higher than the linear model and hence proves to be a better classifier.

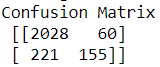
**Sigmoid kernel:**

Implemented the Sigmoid kernel package of SVC using the training dataset. Used GridCv to obtain the optimized value for the kernel with value of C as 1 and gamma as 0.01. By using the test data, obtained an accuracy of 89% for the default case.

The classification report is as below:

We can interpret the precision of the dataset to be 0.89 (weighted average) with an accuracy of 89%. The accuracy of the model is similar to RBF and higher than Linear kernel model.

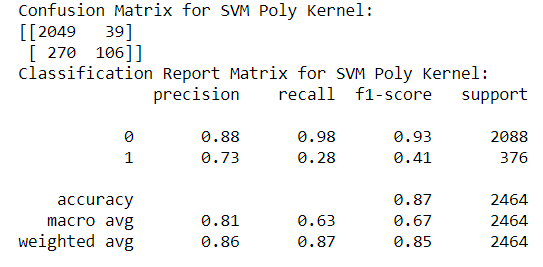
Following confusion matrix gives classification rates for this kernel.

Here we can interpret that the model performs good as we have a high precision and high recall. The precision and recall values are higher than the linear and RBF model and hence proves to be a better classifier.

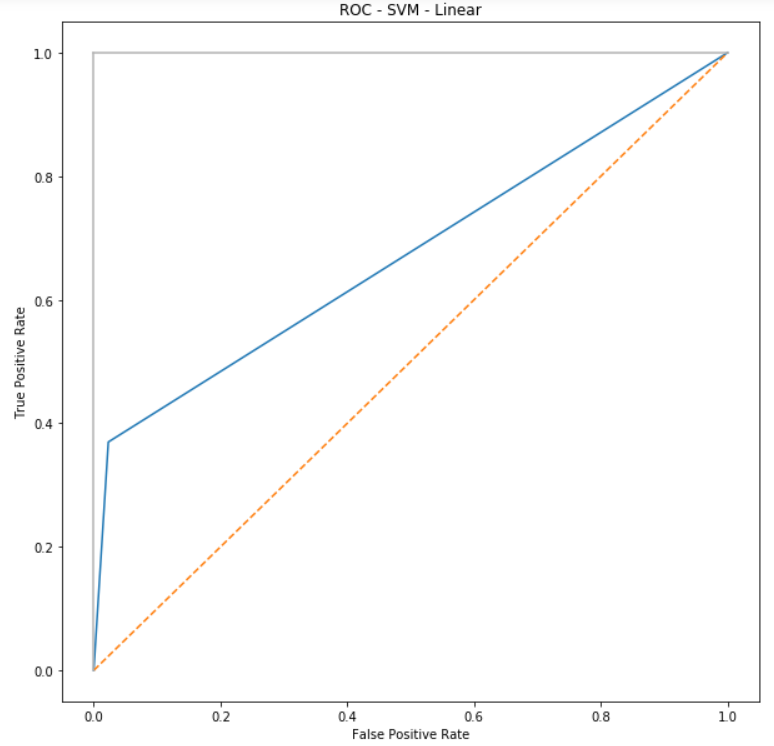
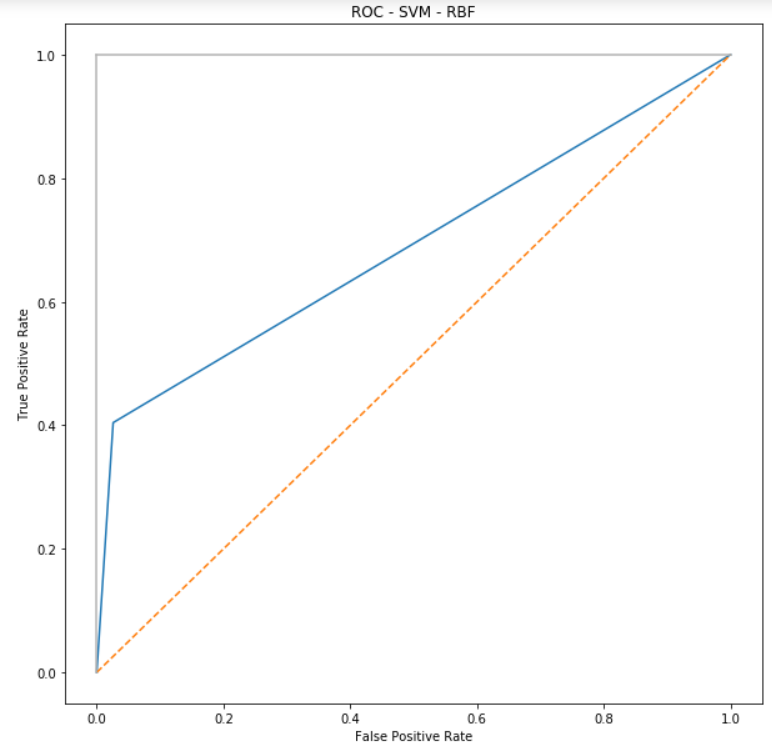
**Polynomial kernel:**

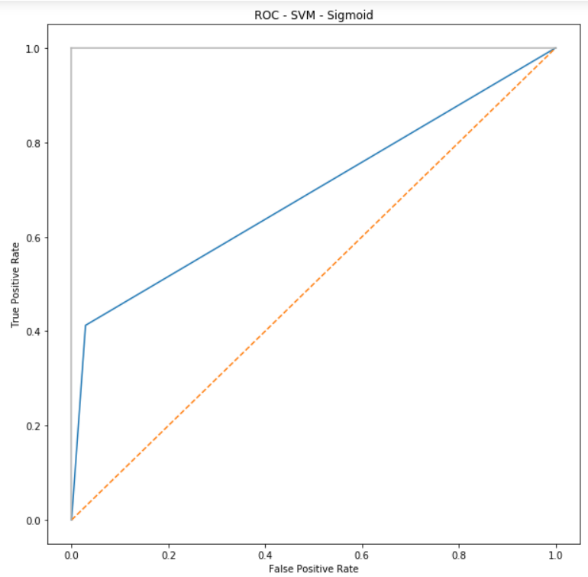
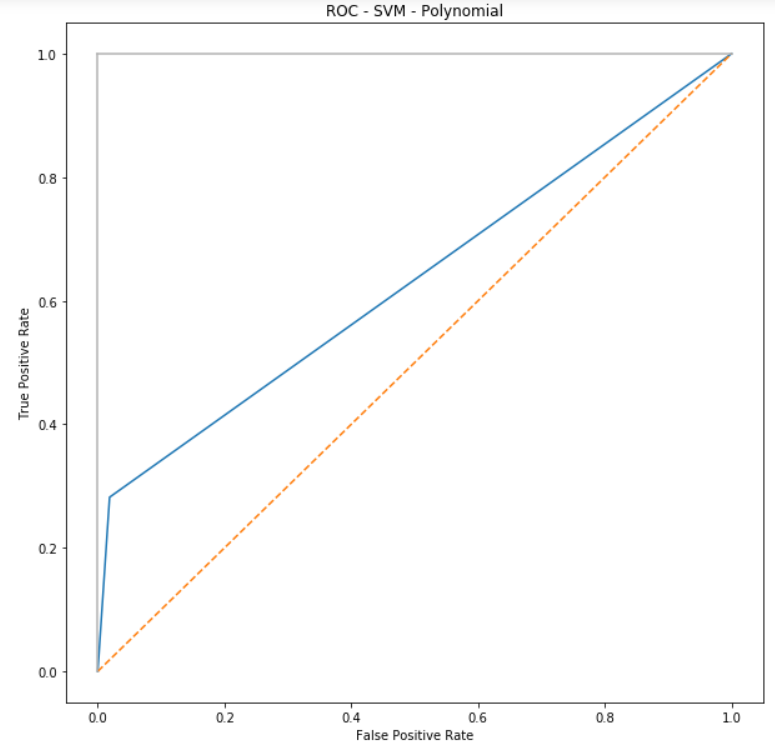
Implemented the Polynomial kernel package of SVC using the training dataset. By using the test data, obtained an accuracy of 89% for the default case.

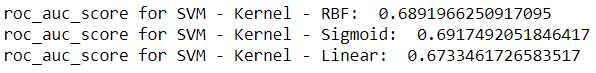
The classification report and confusion matrix is as below:

We can interpret the precision of the dataset to be 0.86 (weighted average) with an accuracy of 87%. The accuracy of the model is lesser than RBF, Linear and sigmoid kernel model.

From the above reports we can conclude that the RBF kernel model performs best among the SVM kernel models. Plotting the ROC Curves:

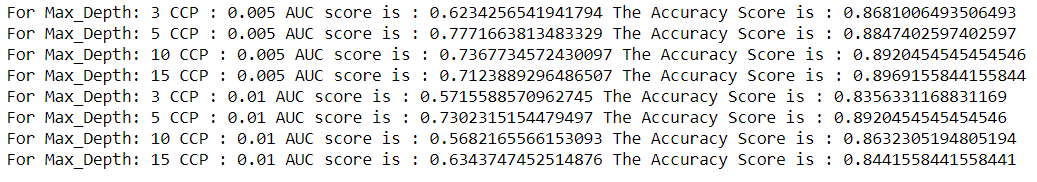


From the above ROC curves, we can interpret that the AUC is highest for  Sigmoid SVM model. The model performs best with high precision and recall.

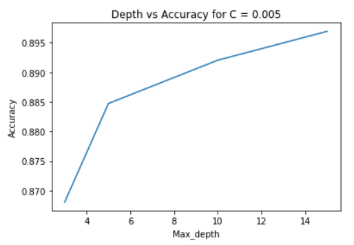
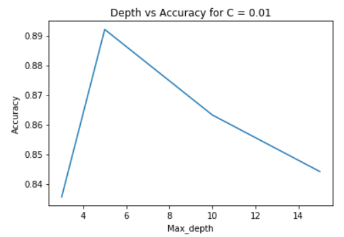
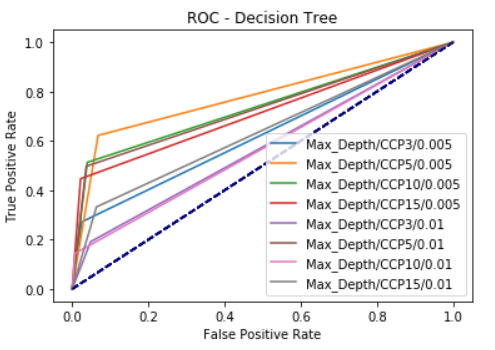
|  |  |  |
| --- | --- | --- |
| **Kernel** | **Accuracy Score%** | **ROC Score %** |
| Linear | 88 | 0.67 |
| Gaussian | 89 | 0.68 |
| Polynomial | 87 | 0.63 |
| Sigmoid | 89 | 0.69 |

# **Decision Tree Classifier**

We used the gini to calculate the best attribute for each node. As there is not much imbalance in the dataset, we use Gini. We determine the best model by running various combinations for maximum depth and pruning alpha values. Below is the summary statistics:

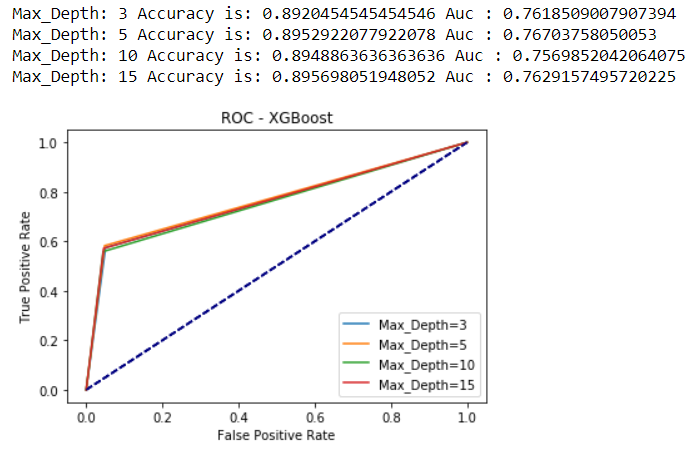


We can interpret that for depth value of 5 and alpha (CCP) as 0.005, we obtain a combination of highest accuracy and AUC score. Below is the ROC graph –

We can infer from the above graphs for validation of our findings. Accuracy – 88.4% AUC – 0.77

# **XG Boosting**

Implemented the xgboosting package from the sklearn library. We perform the same decision tree algorithm with n\_estimators as 1000 for various child weights and gamma value of the tree. We run gridsearchCV to obtain the best parameters for the model with child weight as 7 and gamma as 0. Running this model on various depths and obtaining accuracy scores and ROC curves to obtain the best fit.

Here we notice that Accuracy and AUC is maximum for depth 3 and 5. As we need a model which is not complex, we can say that model with depth - 3 performs best given the complexity and high accuracy and AUC. Accuracy – 89.9 AUC 0.76

# **Cross Validation**

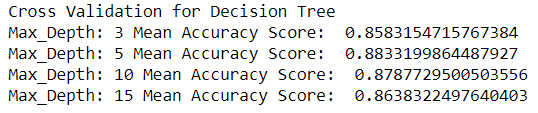
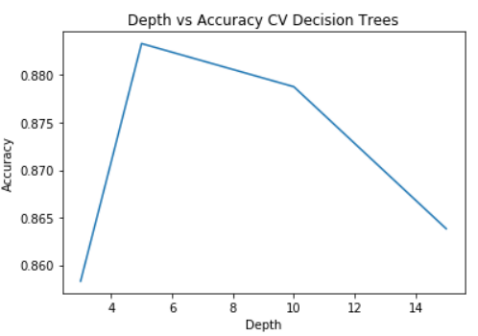
Performed cross-validation across all the models above with their best parameter values. The results are:

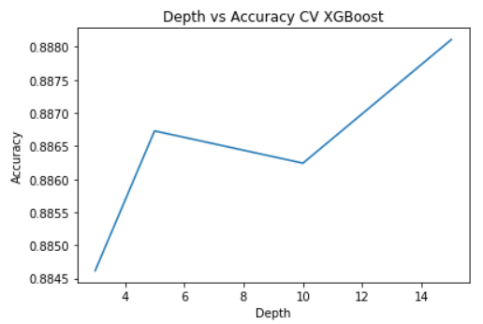
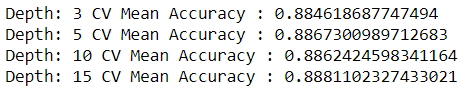
Accuracy of SVM Linear model is 88.35%

Accuracy of SVM rbf is 88.7%

Accuracy of SVM polynomial is 86.28%

Accuracy of SVM Sigmoid is 83.95%



Cross validation for XGBoost models are below:

From the above, we obtain varied results from the normal model and the Cross-validation model. Here after performing CV, the model accuracy is drastically improving for SVM. We can see that RBF SVM performs best with Cross Validation Techniques.

The Decision Tree model performs best for depth as 5. It is similar to the result we obtained in the normal decision tree model. The accuracy is slightly better in this model.

Also, the XG boost model delivers similar results across all the depth variation. Hence, we can consider the minimum depth model which reduces complexity and performs better

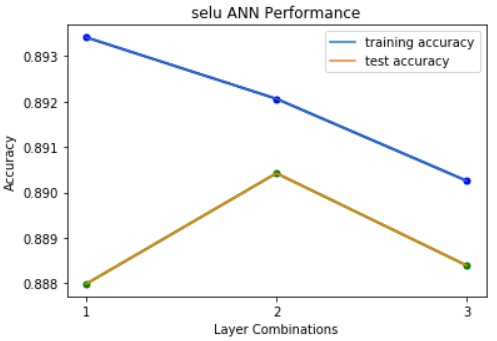
|  |  |
| --- | --- |
| **Kernel** | **Cross Validation Score** |
| linear | 0.8835 |
| rbf | 0.887 |
| polynomial | 0.8628 |
| Sigmoid | 0.8395 |

The two best models based on accuracy and roc\_score is the xgboosted model with a accuracy of 89% for a tree depth of 5. The next best model is the SVM rbf model with an accuracy of 88%. Comparing these 2 models, we can imply that the boosted model would be lot faster in the computation time and provides better results compared to all the models. This is because, most of the data in the model is categorical and hence, a better result for this model.

# **Neural Networks:**

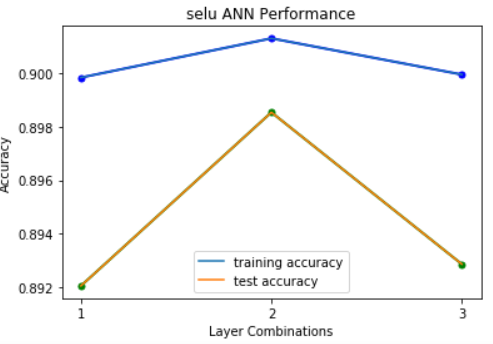
I have used sequential package from keras to implement the various parameters for neural networks. The dense package is used to add layers and number of nodes into the model. I have experimented with number of layers, number of nodes and activation functions and used the best parameters to run batch normalization and dropout networks.

I have used binary cross entropy as the loss model because our target values are in the set {0,1} and accuracy metrics to validate the model. I have used stochastic gradient descent for the model optimizer. Also, I have used early stopping to stop the model if the model stops improving.

**Selu:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Combination** | **Nodes** | **Accuracy** | **Precision** | **Recall** | **AUC** |
| 1 | 30 | 88.80% | 0.88 | 0.89 | 0.7005 |
| 2 | 40 | 89.04% | 0.88 | 0.89 | 0.7009 |
| 3 | 50 | 88.84% | 0.88 | 0.89 | 0.6975 |

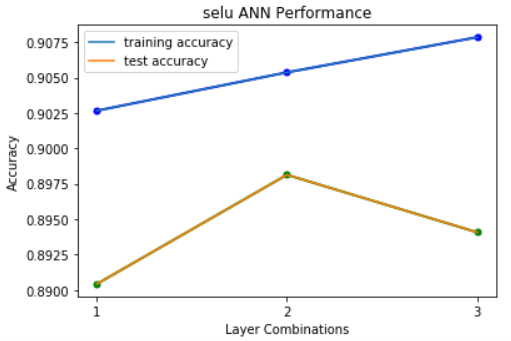
**1-Layer combination:**

From the above tabulations we can infer that with 30 nodes in hidden layer, we have 88.80% accuracy and increases to 89.04% when we increase the nodes to 40 and then slightly decreases as we increase the nodes to 50. We obtain a high precision and recall score - 0.88. This model performance can further be compared when we increase the number of layers.

**2-layer combination:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Combination** | **Nodes** | **Accuracy** | **Precision** | **Recall** | **AUC** |
| 1 | 30,30 | 89.20% | 0.88 | 0.88 | 0.7356 |
| 2 | 40,40 | 89.85% | 0.89 | 0.90 | 0.7504 |
| 3 | 50,50 | 89.29% | 0.88 | 0.89 | 0.7328 |

From the above tabulations we can infer that with 30 nodes in hidden layer, we have 89.20% accuracy and increases to 89.85% when we increase the nodes to 40 and then slightly decreases as we increase the nodes to 50. We obtain a high precision and recall score - 0.89. This model performance is better than the model with 1 hidden layer.

**3-layer combination:**

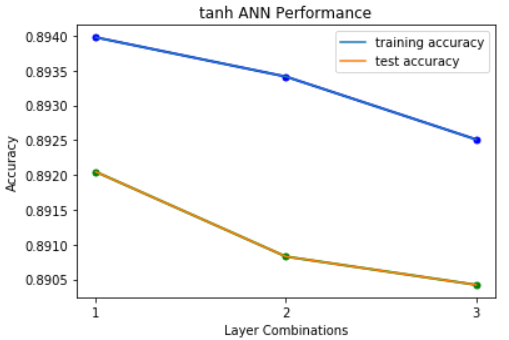
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Combination** | **Nodes** | **Accuracy** | **Precision** | **Recall** | **AUC** |
| 1 | 30,30,30 | 89.04% | 0.88 | 0.89 | 0.7412 |
| 2 | 40,40,40 | 89.81% | 0.89 | 0.90 | 0.7610 |
| 3 | 50,50,50 | 89.41% | 0.89 | 0.89 | 0.7597 |

From the above tabulations we can infer that with 30 nodes in hidden layer, we have 89.20% accuracy and increases to 89.85% when we increase the nodes to 40 and then slightly decreases as we increase the nodes to 50. We obtain a high precision and recall score - 0.89. This model performs similar to the model with 2 hidden layers.

In Selu, we can say that with combination 30,30 in the hidden layers, we obtain the highest accuracy of 89.85%. For this combination, we also have a very high precision and recall and AUC of 0.7504. With selu activation, this combination performs best.

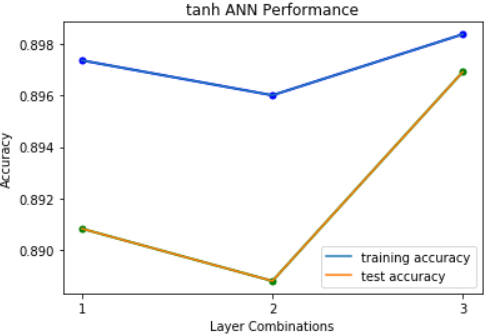
**Tanh:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Combination** | **Nodes** | **Accuracy** | **Precision** | **Recall** | **AUC** |
| 1 | 30 | 89.20% | 0.88 | 0.89 | 0.7116 |
| 2 | 40 | 89.08% | 0.88 | 0.89 | 0.7066 |
| 3 | 50 | 89.04% | 0.88 | 0.89 | 0.7074 |

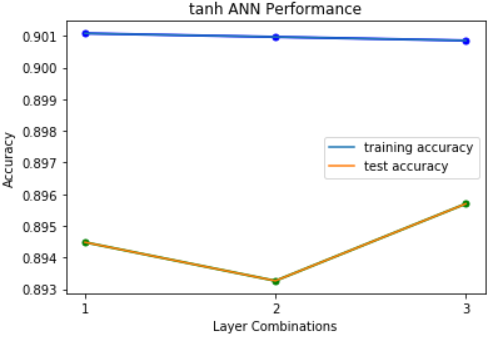
**1-Layer combination:**

From the above tabulations we can infer that with 30 nodes in hidden layer, we have 89.20% accuracy and reduces to 89.08% when we increase the nodes to 40 and further decreases to 89.04% as we increase the nodes to 50. We obtain a high precision and recall score - 0.89. This model performance can further be compared when we increase the number of layers.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Combination** | **Nodes** | **Accuracy** | **Precision** | **Recall** | **AUC** |
| 1 | 30,30 | 89.08% | 0.88 | 0.89 | 0.7338 |
| 2 | 40,40 | 88.88% | 0.88 | 0.89 | 0.7163 |
| 3 | 50,50 | 89.69% | 0.89 | 0.90 | 0.7418 |

**2-layer combination:**

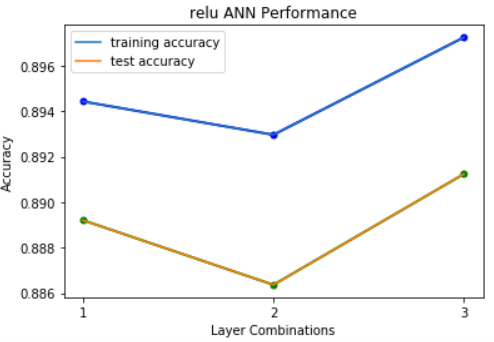
From the above tabulations we can infer that with 30,30 nodes in hidden layer, we have 89.08% accuracy and decreases to 88.88% when we increase the nodes to 40,40 and then slightly increases as we increase the nodes to 50,50. We obtain a high precision and recall score - 0.89. This model performs slightly worse than the model with 1 hidden layer.

**3-layer combination:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Combination** | **Nodes** | **Accuracy** | **Precision** | **Recall** | **AUC** |
| 1 | 30,30,30 | 89.45% | 0.89 | 0.89 | 0.7621 |
| 2 | 40,40,40 | 89.33% | 0.89 | 0.89 | 0.7484 |
| 3 | 50,50,50 | 89.57% | 0.89 | 0.90 | 0.7541 |

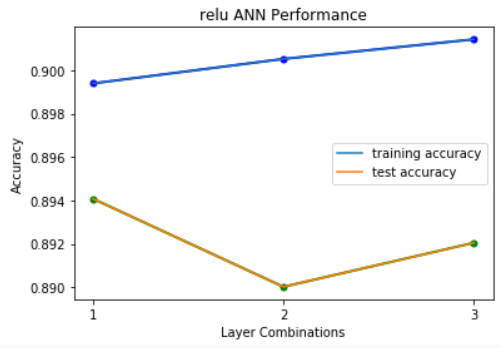
From the above tabulations we can infer that with 30,30,30 nodes in hidden layer, we have 89.45% accuracy but reduces to 89.33% when we increase the nodes to 40,40,40 and then increases as we increase the nodes to 50,50,50. We obtain a very high precision and recall score - 0.89. This model performance is better than the model with 1 hidden layer and 2 hidden layers.

In Tanh, we can say that with combination 40,40 in the hidden layers, we obtain the highest accuracy of 89.69%. For this combination, we also have a very high precision and recall and AUC of 0.7418. This combination performance is slightly lesser than selu activation.

**Relu:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Combination** | **Nodes** | **Accuracy** | **Precision** | **Recall** | **AUC** |
| 1 | 30 | 88.92% | 0.88 | 0.89 | 0.7012 |
| 2 | 40 | 88.64% | 0.87 | 0.89 | 0.6854 |
| 3 | 50 | 89.12% | 0.88 | 0.89 | 0.7057 |

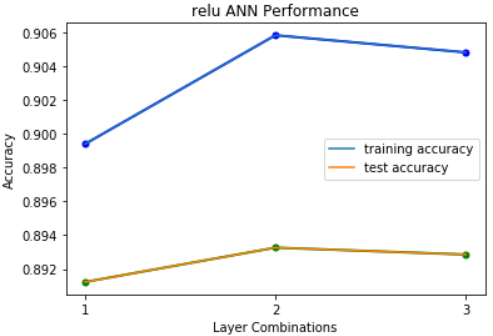
**1-Layer combination:**

From the above tabulations we can infer that with 30 nodes in hidden layer, we have 88.92% accuracy and decreases to 88.64% when we increase the nodes to 40 and then increases to 89.12% as we increase the nodes to 50. We obtain a very high precision and recall score - 0.88. This model performance can further be compared when we increase the number of layers.

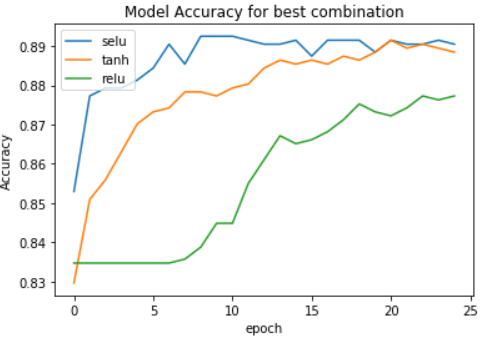
**2-layer combination:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Combination** | **Nodes** | **Accuracy** | **Precision** | **Recall** | **AUC** |
| 1 | 30,30 | 89.41% | 0.89 | 0.89 | 0.7423 |
| 2 | 40,40 | 89.00% | 0.88 | 0.89 | 0.7192 |
| 3 | 50,50 | 89.2% | 0.88 | 0.89 | 0.7356 |

From the above tabulations we can infer that with 30,30 nodes in hidden layer, we have 89.41% accuracy and decreases to 89% when we increase the nodes to 40,40 and then increases to 89.20% as we increase the nodes to 50,50. We obtain a very high precision and recall score - 0.88. This model performance is better than the model with 1 hidden layer.

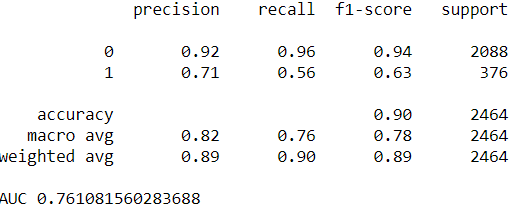
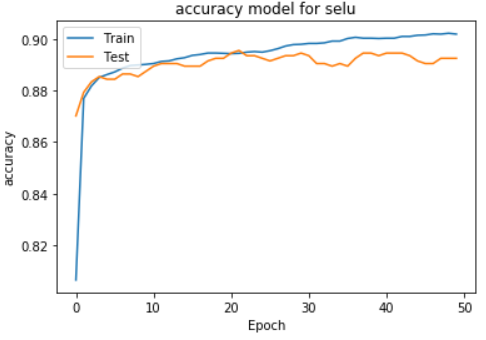
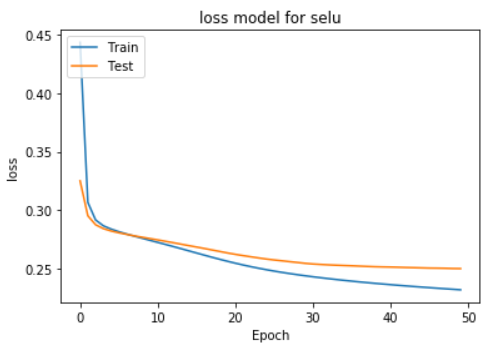
**3-layer combination:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Combination** | **Nodes** | **Accuracy** | **Precision** | **Recall** | **AUC** |
| 1 | 30,30,30 | 89.12% | 0.89 | 0.89 | 0.7700 |
| 2 | 40,40,40 | 89.33% | 0.89 | 0.89 | 0.7516 |
| 3 | 50,50,50 | 89.29% | 0.89 | 0.89 | 0.7568 |

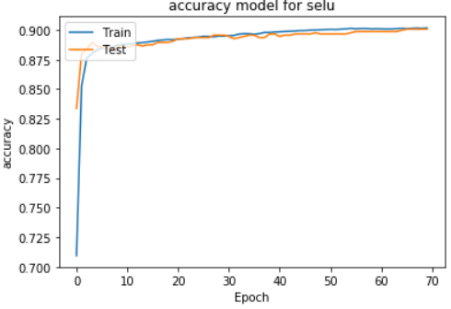
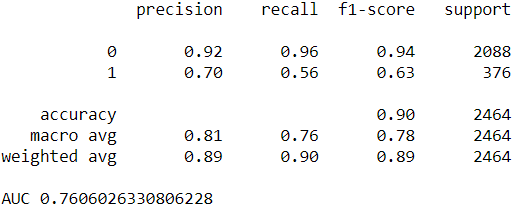
From the above tabulations we can infer that with 30,30,30 nodes in hidden layer, we have 89.12% accuracy and increases to 89.33% when we increase the nodes to 40,40,40 and then decreases as we increase the nodes to 50,50,50. We obtain a very high precision and recall score - 0.89. This model performance is better than the model with 1 hidden layer and 2 hidden layers.

For this dataset, with Selu activation, for combination 40,40,40 in the hidden layers, we obtain the highest accuracy of 89.81%. We have a very high precision - 0.89 and recall – 0.90 and AUC of 0.7610. Hence, this model is considered the best model as it has the highest accuracy.

For the model combination with 30,30,30 with selu activation, we obtain the classification report as follows:

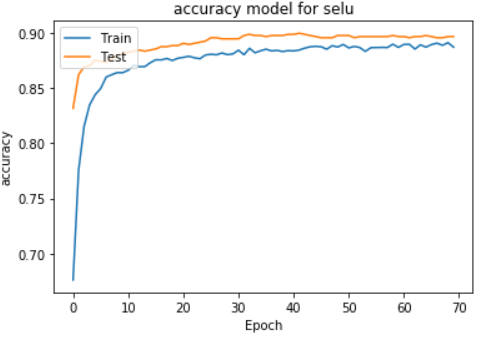


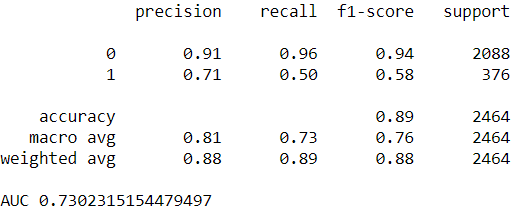
**Batch Normalization:**

We implement batch normalization to our model and observe the findings. By fitting the model, the classification report obtained is as follows:

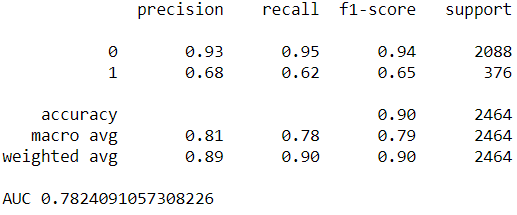
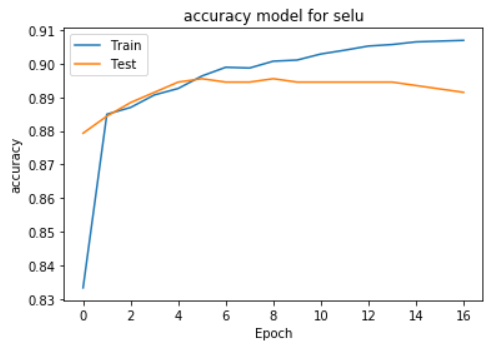
Obtained an accuracy of 89.73% with high precision and recall values and AUC of 0.7606

**Dropout:**

We implement dropout to our model and observe the findings for inference. By fitting the model, the classification report obtained is as follows:

 Obtained an accuracy of 89.20% with high precision and recall values and AUC of 0.7302

**Adam Optimizer:**

I have experimented with the adam optimizer for model optimization. By fitting the model, the classification report obtained is as follows:

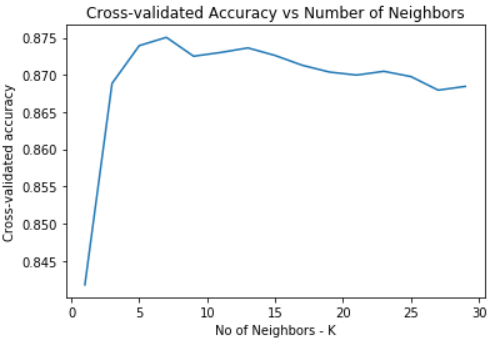
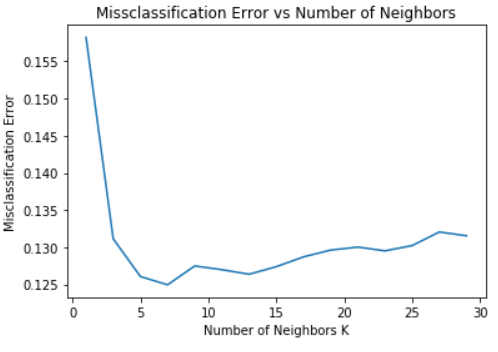
Due to early stopping, we have the model has stopped at 17 epochs and obtained an accuracy of 89.73% with high precision and recall values and AUC of 0.7824

Based on all the models, gradient descent optimizer, batch normalization, dropout and adam optimizer, the best model in neural network is with nodes 40,40,40 with selu activation.

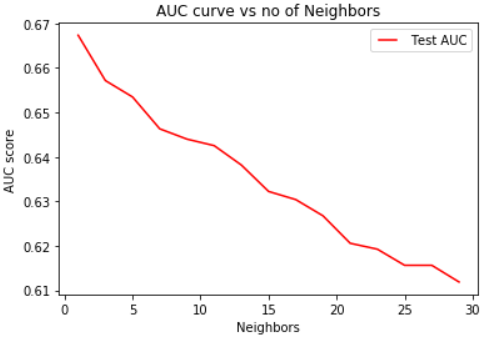
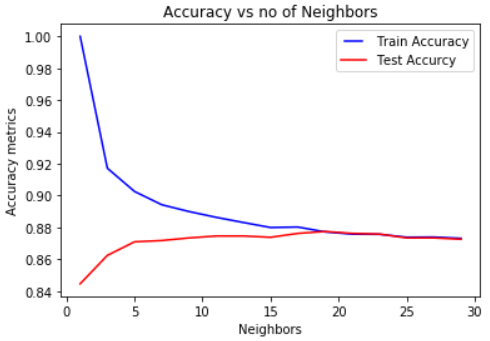
# **KNN model**

I have used the KNeighborsCLassifier package from sklearn. I have performed modelling by varying the number of neighbors to find the optimal value. The various distances that have been experimented with are minkowski, Euclidean, manhattan and Chebyshev. I have used gridsearchCV to find the best set of parameters for the KNN algorithm by varying the neighbors, distance metrics and weight options parameters. The weight options include 1. Uniform – where all neighbors are given equal weightage and 2. Distance – where closer neighbors are heavily weighed than further neighbors.

Experimenting with the number of neighbors, the results are:



From the above graph, we can say that with a cross validation score of 87.5%, the optimal solution for number of neighbors is 7. The misclassification error is least at 12.5% for the optimal solution. Also, for n = 15, we have cross validation score of 87.25% and misclassification error of 12.75%.

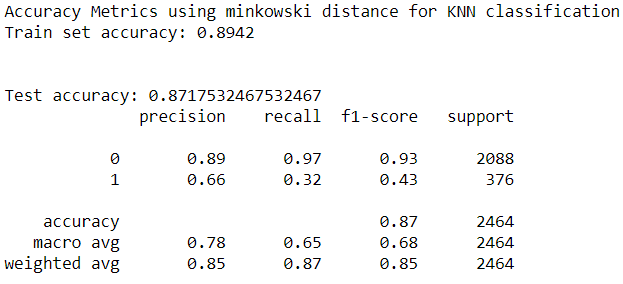
The AUC curve and accuracy curve obtained are as follows:

We observe that:

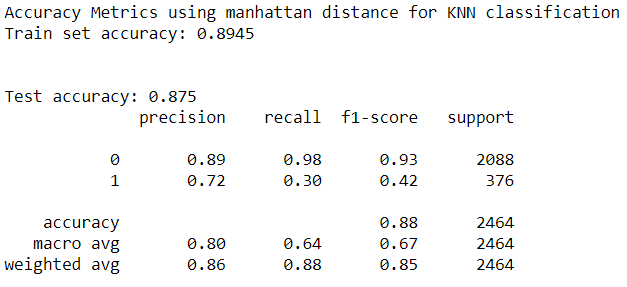
For n = 7 –> AUC = 0.64, Train Accuracy = 90%, Test Accuracy = 87%

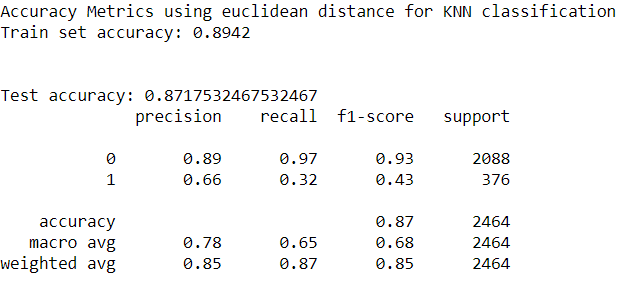
For n = 15 -> AUC = 0.63, Train Accuracy = 89%, Test Accuracy = 88%

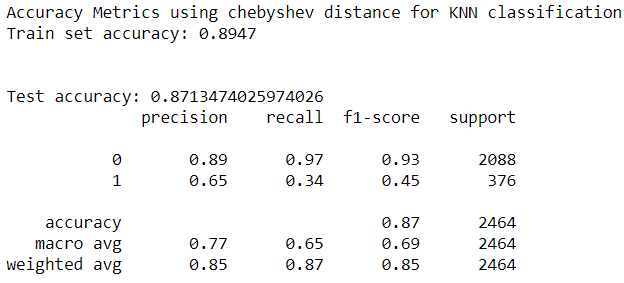
We see that for both the metrics the findings are very similar.

By trying to vary the distance metric we obtain the classification report as follows:

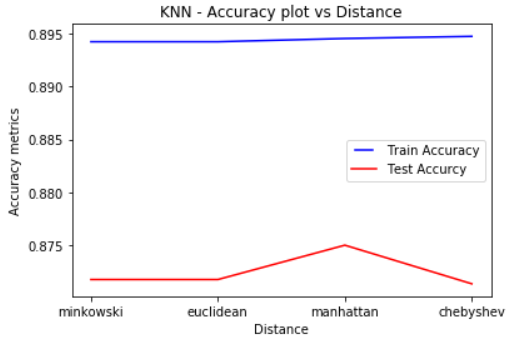
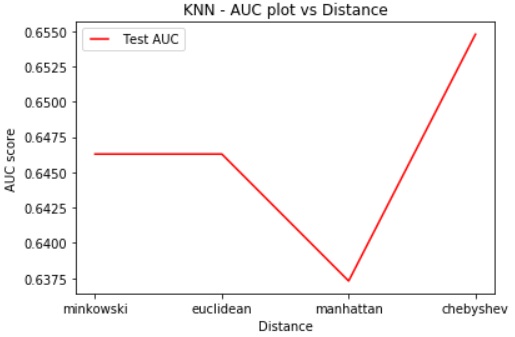
We can interpret the precision of the dataset to be 0.85 (weighted average) for minkowski with an accuracy of 87.15% for the test data.

We can interpret the precision of the dataset to be 0.86 (weighted average) for manhattan with an accuracy of 87.5% for the test data. The model performs better than minkowski distance metric.

We can interpret the precision of the dataset to be 0.85 (weighted average) for Euclidean with an accuracy of 87.15% for the test data. The model performs similar to the minkowski distance metric.



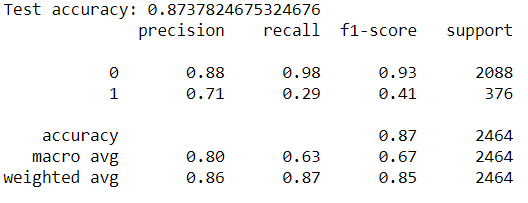
We can interpret the precision of the dataset to be 0.85 (weighted average) with an accuracy of 87.13% for the test data. The model performs better than minkowski and euclidean distance metric.

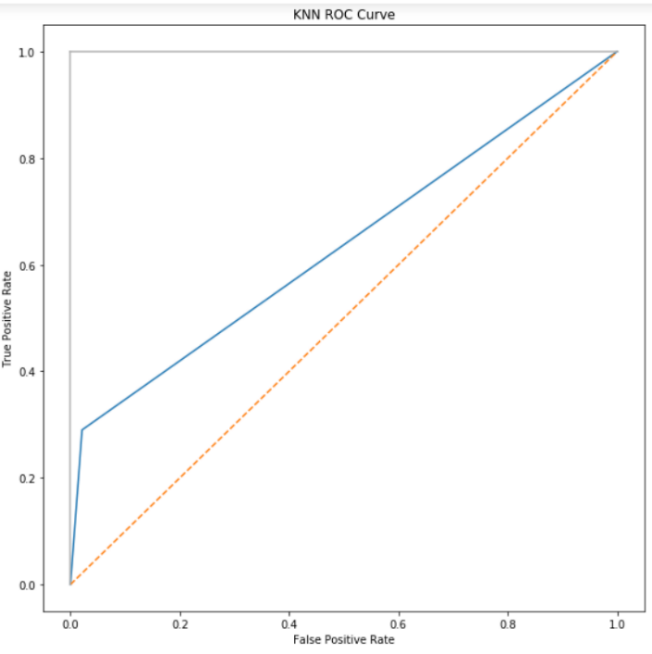


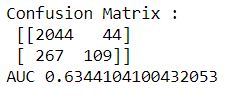
From the above graph, we can see that using Chebyshev distance metric we have the highest AUC value whereas the accuracy is highest using the manhattan distance metric. As there is a significant difference in AUC, we look to have high precision and hence conclude that the Chebyshev distance metric performs better.

The third parameter we experiment on is weight options. We use uniform and distance weights and perform grid search CV to conclude and validate our findings on the best set of neighbors, distance metrics. We obtain the best possible combination with number of neighbors as 15, distance metric as manhattan with weight option as distance.

We then create a model using the best parameters obtained and obtain the classification report.



The AUC for the best subset curve is 0.6344. From the classification report, this model has an accuracy of 87.38% with a precision of 0.86. This model performs best with high precision, high recall.



# **Comparison between models:**

For Neural Networks, the best model obtained is 40,40,40 with selu activation and accuracy of 89.81%. For KNN, the best model obtained is with 5 neighbors and manhattan distance metric with an accuracy of 87.38%. The best model for this data would be the Neural Network Model.

The neural network model performs the best across all the models.

Comparing with all the models for dataset 2:

|  |  |  |
| --- | --- | --- |
| Model | Accuracy (in %) | Rank (Best Model |
| SVM | 88 | 4 |
| Decision Tree | 88.4 | 3 |
| XGBoost | 89 | 2 |
| Neural Networks | 89.81 | 1 |
| KNN | 87.38 | 5 |