Prepared by: Sai Lakshmi Nikitha Akarapu

ID: SXA210112 Date: 07/22/2022

BUAN 6341 Applied Machine Learning

ASSIGNMENT NO 3

Seoul Rental Bike prediction Part III

Executive Summary

- **Sensible feature selection and preparation improve prediction accuracy.**
- **Converted the data set to binary classification using a median of the output parameter.**
- **Experimented with the data set by implementing Artificial Neural Networks and optimizing with a number of layers and nodes, activation functions, and solvers.**
- **Optimizing parameters like learning rates and tolerance improves the Neural Networks' accuracy.**
- **Experimented using different K-folds for checking model performance on new data and experimented with different data sizes for checking model performance.**

Introduction

In this project, the objectives were to convert the data set to a binary classification problem by thresholding the output to a class label and implement Artificial Neural Networks with different activation functions, solvers, number of layers, and nodes. I have optimized hyperparameters like learning rate and convergence threshold for choosing the best model through experimentation for classifying the data set.

About the Data

The dataset consists of 14 features and 8760 records. The data is the rented bike count captured at each hour and weather conditions (Temperature, Humidity, Windspeed, Visibility, Dewpoint, Solar radiation, Snowfall, Rainfall), along with date information. To remove the serial correlation with time, the hours variable is converted to dummies to predict.

Project Outline

Part 1: Implement Artificial neural networks to classify your classification problems and experiment with a number of layer and number of nodes, and activation functions.

Part 2: Use the Cross-validation technique with optimized models of MLP classifier using the K-Fold technique to check the accuracies of the models to deploy in production.

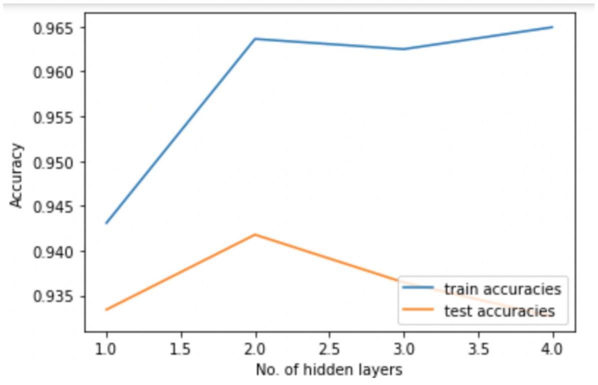
Part 3: Experiment with the different data set sizes to check the accuracies of the models.

Part 1: Artificial Neural Networks

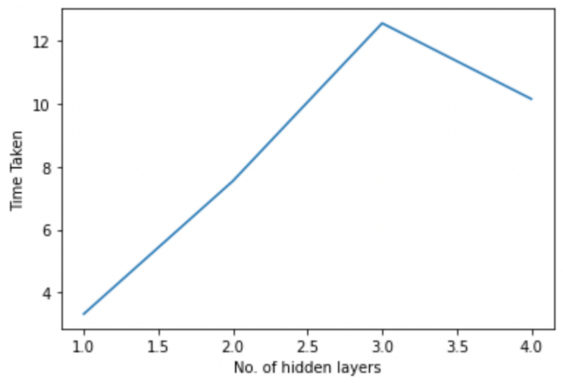
In assignment 02, we have already converted the data set to a binary classification problem using a median of the output parameter to a class label. ANN algorithm is implemented on this data set since it fits well to the data set with a very high number of features and observations. We will use the Multi-layer Perceptron classifier (neural_network.MLPClassifier) from a scikit-learn package for the experimentation.

1. Experimented with a different number of layers by keeping the solver for weight optimization as stochastic gradient descent ('sgd'):

The learning process of a neural network is performed with the hidden layers. For most problems, one hidden layer is sufficient, but for capturing complex input-output relationships, we go for the addition of more hidden layers. We experimented to know how many 'number of layers' are suitable for our dataset.



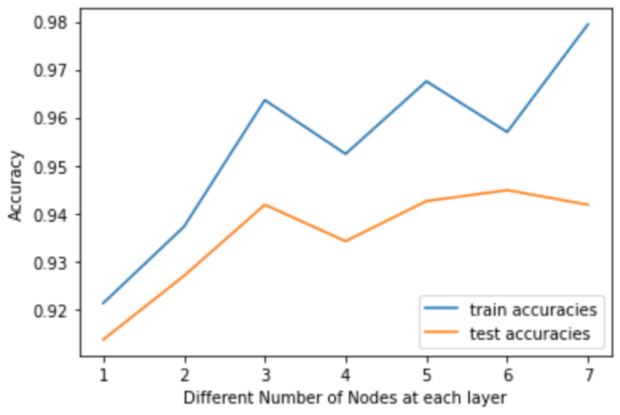
At the number of hidden layers of 2, the test accuracy is maximum compared with other points, and train and test accuracies are closer. The model on test data is moving toward bias with the increasing number of layers. From the experimentation, we can conclude that the model is efficient and ideal at the Number of Hidden Layers of 2.



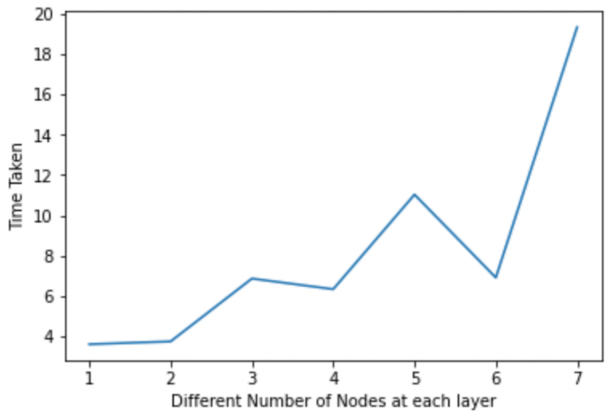
From the experimentation, we can observe that by increasing the number of layers, the time is taken for convergence also increased and peaked at the Number of hidden layers=3. The model will one layer has taken less time to converge but the accuracies are not up to the mark. The model with 2 layers takes ideal time to converge and accuracies are efficient.

2. Experimented with the model by keeping the number of layers as 2 and with a number of nodes, below are the observations:

From the above experiment, we have concluded to go with 2 layers for this binary classification problem. For the number of nodes experiment, we have built 7 cases which are (2,2), (5,5), (10,10), (15,15), (20,20), (25,25), (30,30) represent the number of nodes at each layer and named 1 to 7 respectively.



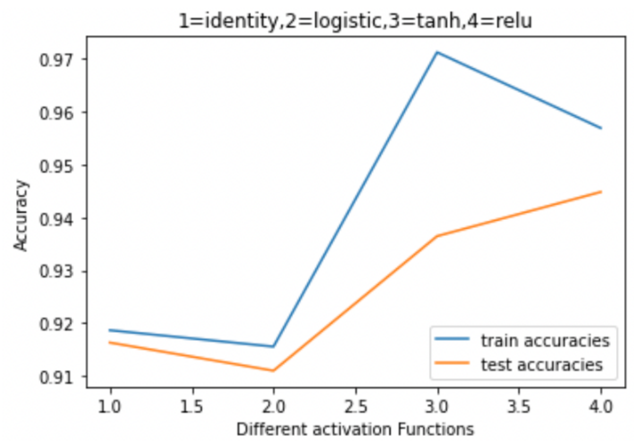
From the experimentation, we can observe that at point 6, the train and test accuracies are good and closer to each other. With the increase in the number of nodes, the model on the train dataset moves towards high variance (overfitting). We can conclude the model is efficient and ideal when the number of nodes at each layer is 25.



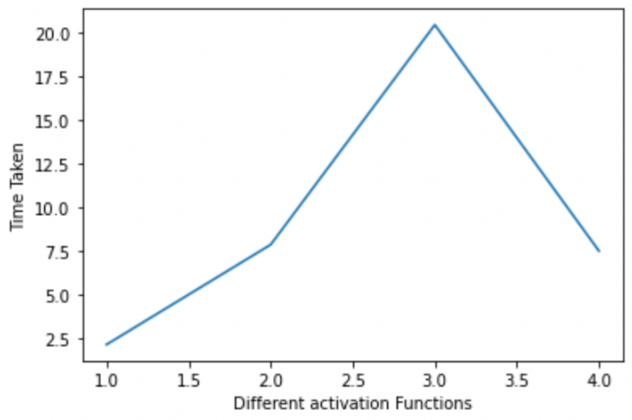
From the experimentation, we can observe that with an increase in the number of nodes, the time is taken for convergence also increases but not in a linear fashion. The model with 25 nodes at each layer is converging better.

3. Experimented with the model by keeping the number of layers as 2, the number of nodes as (25,25), and with activation functions:

The activation function in the hidden layer will determine the type of predictions the model can make and controls how well the model learns the training dataset. The most commonly used activation functions are ‘identity’, ‘logistic’, ‘tanh’, and ‘relu’.

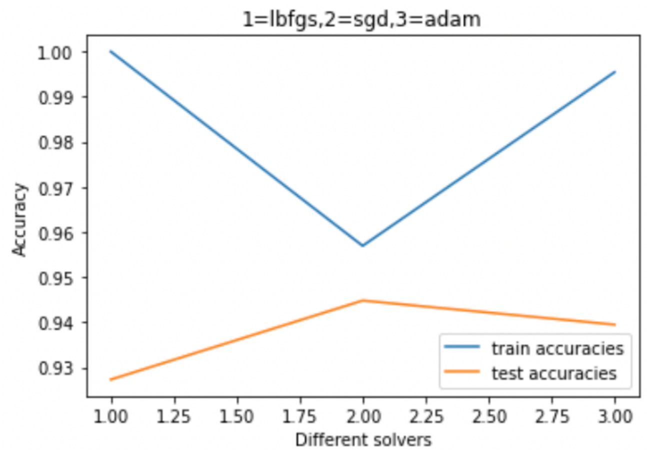


From the experimentation, we can observe that model with the hyperbolic tan function ('tanh') is performing well on the training dataset compared to the test dataset leading to overfitting. In point of variance and accuracies, the model with the rectified linear unit function ('relu') is efficient in classifying.

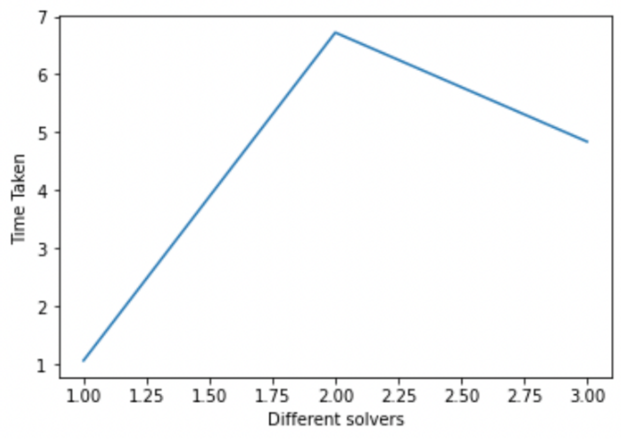


From the experimentation, we can observe that the time taken for convergence is greater for the model with the ‘tanh’ activation function. The model with the ‘relu’ function converges faster than ‘tanh’.

4. Experimented with the model by keeping the number of layers as 2, the number of nodes as (25,25), ‘relu’activation function and with solvers:

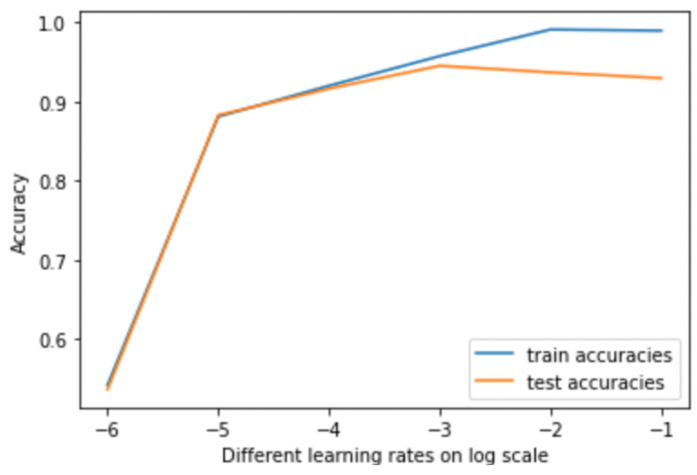


From the experimentation, we can observe that ‘lbfgs’ and ‘adam’ models on the training dataset led to overfitting (high variance). In point of variance and accuracies, the model with the stochastic gradient descent solver is efficient in classifying.

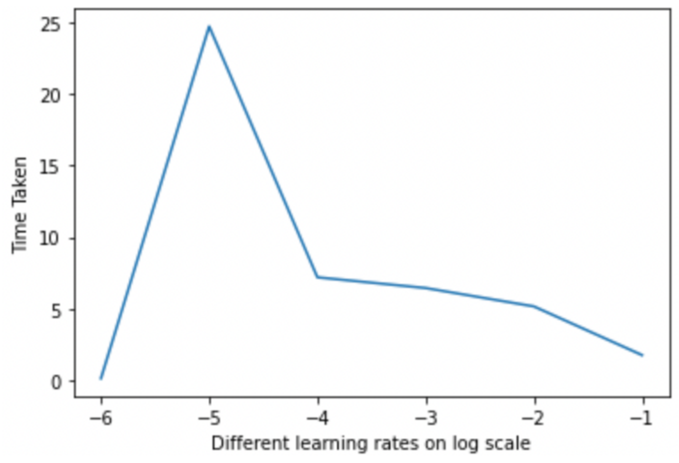


From the experimentation, we can observe that the time taken for convergence is greater for the model with the stochastic gradient descent solver but the accuracies and gap between the accuracies are ideal.

5. Experimented with the model by keeping the number of layers as 2, the number of nodes as (25,25), 'sgd' activation function, and with learning rates:

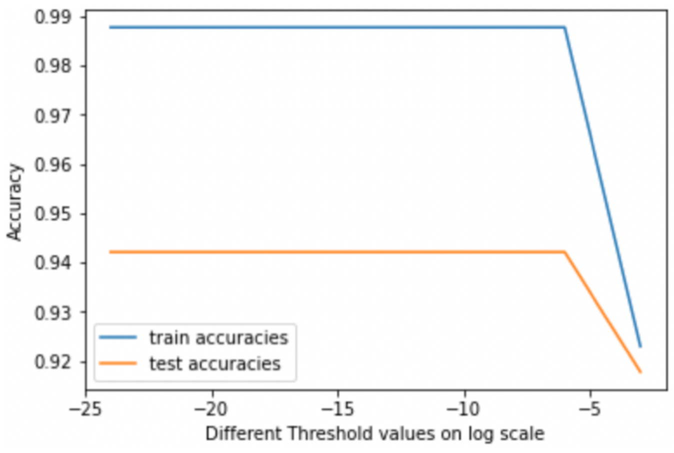


The experimentation shows that both train and test accuracies travel in the same direction through the learning rate range. The accuracies for train and test led to overfitting after a 0.001 learning rate. From this, we can conclude that the train and test accuracies are better converging and optimal at an alpha value of 0.001.

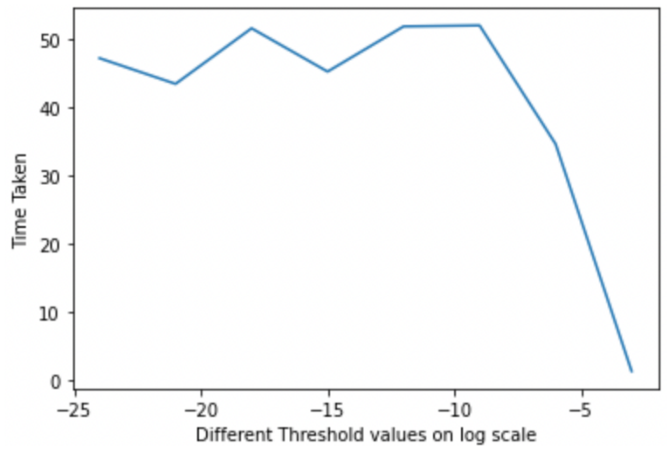


As the learning rate increases, we can observe that time taken for convergence is increased till alpha = 0.00001 and then decreased. We can say that model at alpha = 0.001 has an ideal time to converge.

6. Experimented with the model by keeping the number of layers as 2, the number of nodes as (25,25), 'sgd' activation function, learning rate at 0.001, and thresholds:



The experimentation shows that both train and test accuracies travel in the same direction through the threshold range. The accuracies for train and test didn't differ till 1e-6. Where the model on the 'train' dataset moves from overfitting to bias. We can consider the model at a threshold value of 1e-4 as a baseline.



As the threshold increases, we can observe that time taken for convergence decreases after the 1e-10 threshold value. The model at a threshold value of 1e-4 converges faster than at other values.

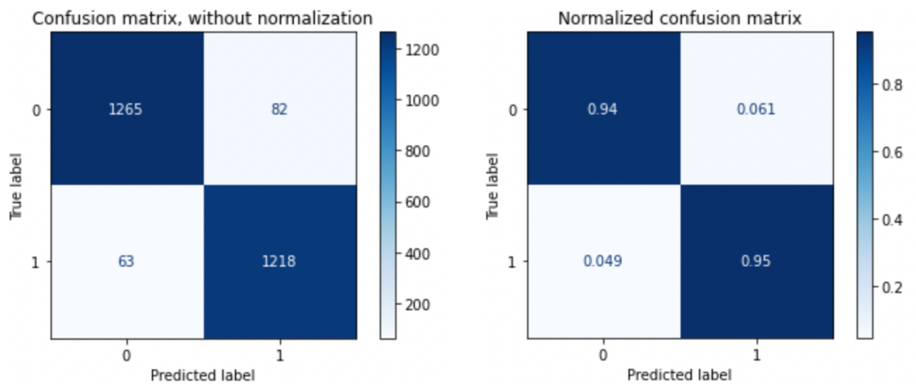
Conclusion:

1) From the MLP classifier experiment, we can conclude that the model has better accuracies for train and test data set at the **number of layers = 2, the number of nodes at each layer = 25, activation function = 'relu', solver = 'sgd', alpha = 0.001 and tol = 1e-4.**

2) The accuracies of train and test data sets of optimized MLP classifier.

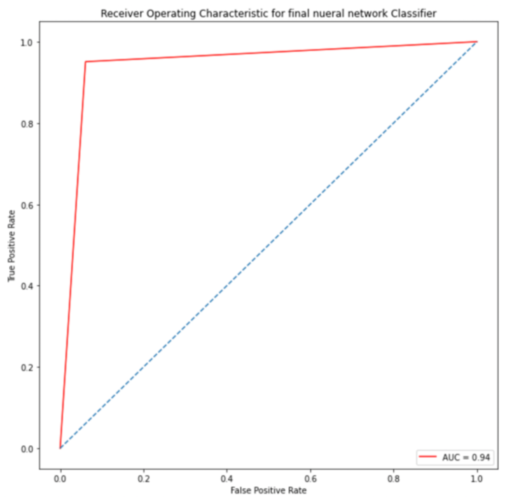
Model	Train Accuracy	Test Accuracy
MLPClassifier	95.69	94.48

3) Confusion matrix of optimized MLP classifier on test data set.



After optimizing the model with the hyperparameters of number of layers = 2, the number of nodes at each layer = 25, activation function = ‘relu’, solver = ‘sgd’, alpha = 0.001 and tol = 1e-4, the model has better convergence and classification. There are 82 Type I Errors and 63 Type II Errors, we look for minimizing the Type I errors as we are going to predict the count of rental bikes.

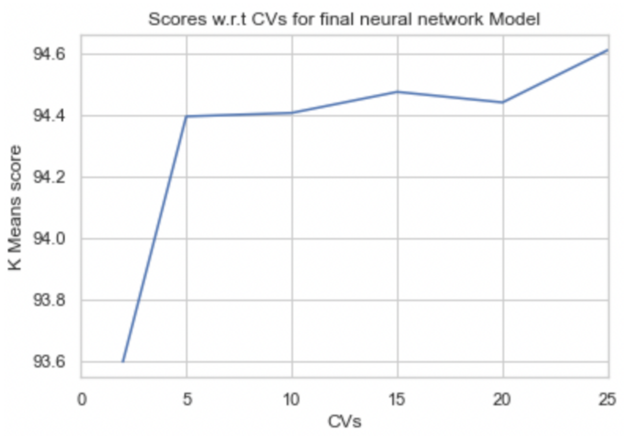
4) ROC and AUC curve of optimized MLP model



The optimized model with the MLP classifier has an AUC of 0.94. The model with the MLP classifier is good at classifying.

Part 2: Cross Validation

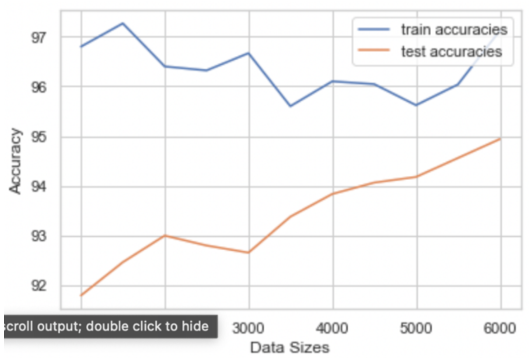
Now we will use K-Fold Cross Validation Techniques to check the model performance for the optimized neural networks MLP classifier.

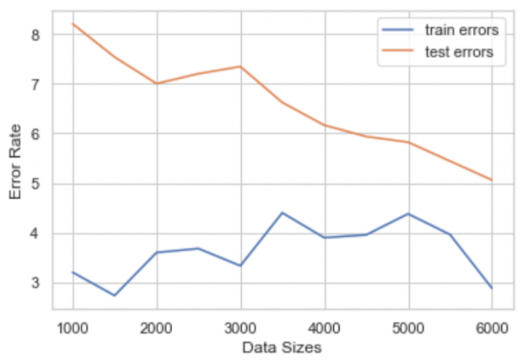


The mean value of accuracy of the model increases from 93.6% to 94.6% with an increasing number of folds. But without the cross-validation technique, the accuracies are around 94 for the MLP model. That means the optimized model is best for classifying the unseen data or new data.

Part 3: Data set Sizes

Now we will use the optimized neural networks classifier model to check the model performance for different train and test data sets sizes.





With the increase in data sizes, we can observe that the accuracies are increasing and reducing the bias. The model is performing better on the test data set with an increase in data sizes which rose from below 92 to 95% whereas declined from 8 to around 5%.

Results:

Through MLP classifier experimentation, we found that hyperparameters like number of layers, number of nodes, solvers, activation functions, learning rate, and threshold are effective for better accuracy. Therefore, machine learning involves balancing accuracy and computational limitations.

Models	Train Accuracy	Test Accuracy	AUC	Cross-Validation	Type I Error	Ranking
Optimized SGD	91.94	91.2	0.91	70-76	108	4
Optimized SVM	96.46	92.35	0.92	74-79	95	3
Optimized DT	88.47	87.74	0.88	87.5-88	207	2
Optimized MLP	95.69	94.48	0.94	93.6-94.6	82	1

1) From the normal split accuracy table, we can say that all classifiers are good at classifying the data whereas the SVM and MLP classifiers may lead to overfitting the training dataset or good with the present dataset.

2) When we observe the AUC measure of separability, the model with the MLP classifier is highly capable of distinguishing between class 0 and class 1, on the other hand, the AUCs for other classifiers are almost the same.

3) From the cross-validation perspective, we can conclude that the model with the Neural Network MLP classifier is better at classifying the existing data and new data when compared to the other classifiers as the scores are almost similar in the case of k-fold and normal split.

4) As we focus on predicting classes 0 and 1, we concentrate on Type I error to minimize them as possible. We can say that the MLP classifier has a good class separation capacity and its performance on new data is good.

5) Increasing the dataset sizes led the model towards low bias and low variance as both train and test accuracies are getting better with the data sizes.

Overall, we can conclude that the model with the Neural Networks classifier is good at predicting classes from the accuracy, AUC, Type I Errors, and Cross-Validation point of view as it is great with unseen or new data.