Prepared by: Jyotsna Nagaprasad Ambekar Date: 07/24/2022

BUAN 6341 Applied Machine Learning

ASSIGNMENT NO 3
Seoul Bike Data

Executive Summary

- **Classification problem with data classified based on median.**
- **Features without high correlation are used to build model using Neural Network algorithm**.
- **Implementation of Stratified K-fold Cross Validation show that the model hasn't overfitted on the data with 0.16% standard deviation of accuracies.**
- **Prediction accuracy can be further improved by implementing XGBoost or Random Forest Algorithm that is apt when data has multiple discrete features.**

Context

Currently Rental bikes are introduced in many urban cities for the enhancement of mobility comfort. The goal is to help commuting in the cosmopolitan cities and reduce the pollution amounts caused by cars, individual motorcycles etc. and create green cities altogether. It also is a healthiest way for travelling to work or school. It is important to make rental bike available and accessible to the public at the right time lessening the waiting time. Eventually, providing the city with stable supply of rental bikes becomes a major concern as it can be used as alternative to the commuters in cities. The crucial part is the prediction of bike count required at each hour for the stable supply of rental bikes.

Introduction

In this project, the objectives were to implement Neural Network algorithm to predict the Rented Bikes count on a given day with given attributes. This report details various experiments conducted using the dataset.

About the Data

The dataset consists of 14 features and 8760 records with no missing values. The data contains information regarding weather and how much solar radiation is observed on the day, seasons, holiday, functional hours etc. The target variable is the rented bike count per hour.

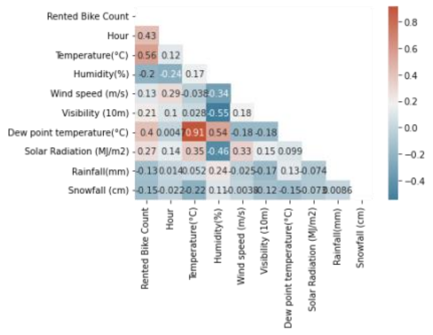
Attribute information

Date : year-month-day
Rented Bike count - Count of bikes rented at each hour
Hour - Hour of the day
Temperature-Temperature in Celsius
Humidity - %
Windspeed - m/s
Visibility - 10m
Dew point temperature - Celsius
Solar radiation - MJ/m2
Rainfall - mm
Snowfall - cm
Seasons - Winter, Spring, Summer, Autumn
Holiday - Holiday/No holiday
Functional Day - NoFunc(Non Functional Hours), Fun(Functional hours)

The independent variables used for training the model are Hour, Temperature(°C), Humidity(%), Wind speed(m/s), Visibility (10m), Solar radiation (MJ/m2), Rainfall(mm), Snowfall, Seasons & Holidays one hot encoded columns. The season column encoded as 0 is Autumn, 1 is spring, 2 is summer, 3 is winter. The Holiday column encoded as 4 is Holiday and 5 is No-Holiday

Data Preprocessing

It is the technique in which data is processed so that it is appropriate to feed the model. Preprocessing steps applied were One hot encoding to encode categorical variables like Seasons & Holiday columns, Splitting the dataset into training and test sets, Feature scaling for standardizing data. Firstly, data exploration was done by understanding the types of data present and description of data. The Date column which was initially string was converted to DateTime format.



The temperature and dew point temperature have correlation value of 0.91. Due to this high correlation, multi collinearity arises if both the features are used to develop the model. Hence dew point temperature is dropped from the independent variables.

Experimentation

Task – 1:

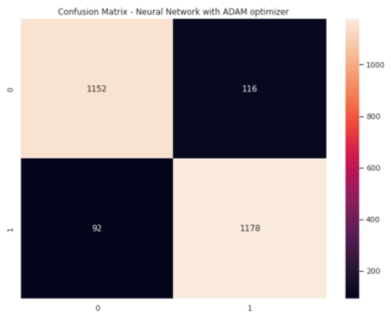
Artificial Neural network algorithm is intended to replicate the behavior of neurons in human brain. The nodes in the Neural network take the inputs from the data and compute weights to suit the input values. Similar to neurons connected in human brain, the nodes between different layers are interconnected to form dense network. These nodes are capable of learning, generalizing training data through a non-linear model and derive results. The 3 main layers of ANN are Input Layer through which the data is fed into the NN, Hidden Layer in which processing occurs, Output Layer which tells us the output of the algorithm. Computation of weights and understanding the trend of the data happens in Hidden layer. The Hidden layers have activation functions apart from adding non-linearity in the network, also convert input signal of weighted sum of the inputs into acceptable output. It helps the network to understand the complex trends in data.

ANN model developed is a multi-layer feed forward Neural Network with 1 input layer with 14 nodes, 4 hidden layers with 15, 12, 10, 5 nodes in each and one output layer with one node. The output layer has one node because the classification is done on binary target variable. For output layer, activation function is sigmoid. Sigmoid function is not used in any hidden layer or input layer because it is computationally expensive, causes vanishing gradient problem and not zero-centered. All other layers have 'ReLU (Rectified Linear Unit) f(x) = max(0, x)' as activation function. This is widely used in NNs as it is easy to compute, doesn't saturate and doesn't cause vanishing gradient problem. The model was compiled and fitted with different optimizers and graphs of accuracies for training and test sets were plotted along with time taken for different epochs.

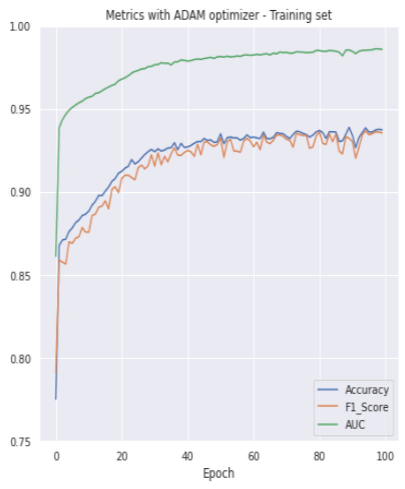
The optimizers include method to train the model during which hyper parameter tuning is done and weights are calculated to reduce loss to make prediction as close as to the ground truth. Since this classification is a binary one, loss function is chosen to be 'binary_crossentropy'. Optimizer shape the model such that it is most accurate with respect to the ground truth and loss function tells optimizer whether it's moving in the right direction. Out of many optimizers, ADAM, Stochastic Gradient Descent (SGD), RMS Prop were used.

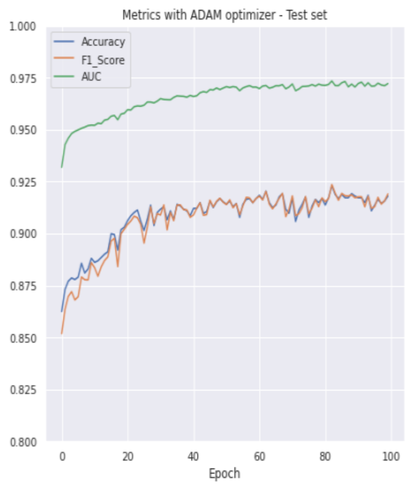
Case – 1: ADAM optimizer

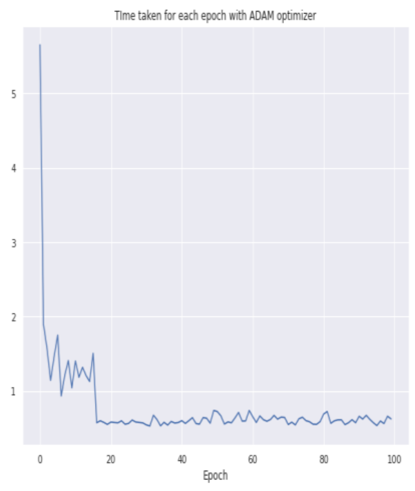
It uses past gradient to calculate current gradient. It uses adaptive estimation of first – order and second – order moments i.e., concept of momentum by adding fractions of previous gradients to current ones. ADAM Class - tf.keras.optimizers.Adam (learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-07, amsgrad=False, name="Adam", **kwargs). It has learning rate default set 0.001. The model is compiled with the same learning rate for each epoch. The accuracy obtained through ADAM optimizer is 91.81%. Confusion matrix –



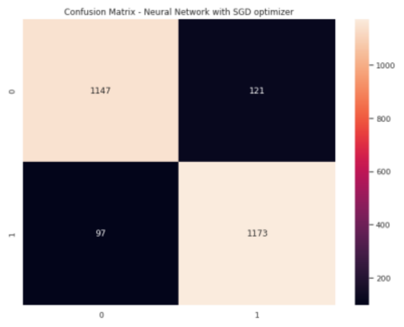
Different metrics were plotted for training and test sets with ADAM optimizer. Accuracies, F1 scores, AUC are plotted for each epoch. Since the dataset is balanced, the F1 scores and accuracies almost have similar values. For an imbalanced dataset it'll be crucial to look at accuracies and F1 scores such that we don't have a misleading model. Area under the curve (AUC) is the measure of the ability of a classifier to distinguish between classes and this is used as a measure of ROC (Receiver Operator Characteristic) curve. ROC curve is probability curve which plots True Positive rate against False Positive Rate at various thresholds and separates signal and noise. Higher the AUC better is the performance of the model at distinguishing positive and negative classes. The model developed can distinguish between the classes because AUC values range between 87% to 98% approximately for training set and 93% to 97% for test set. Time taken for each epoch is also calculated and plotted. For ADAM optimizer, during the initial epochs it takes comparatively longer time to calculate appropriate weights during training. As the model converges, it takes relatively short time. The same behavior can be seen in the graph.





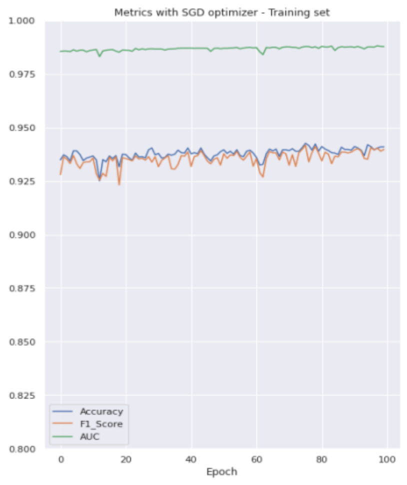


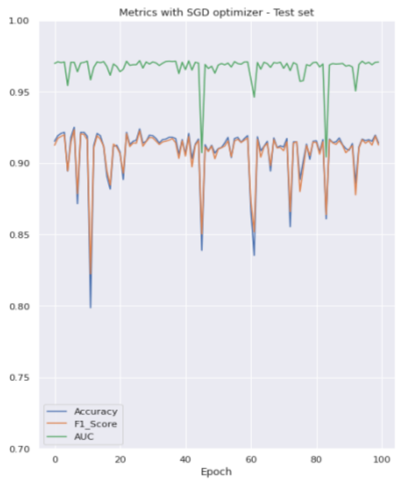
Case – 2: Stochastic Gradient Descent optimizer

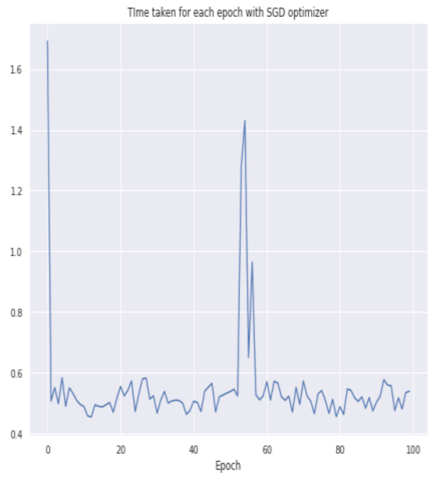


SGD Class: tf.keras.optimizers.SGD (learning_rate=0.01, momentum=0.0, nesterov=False, name='SGD', **kwargs) Each parameter is updated to the opposite direction of partial derivative. The accuracy obtained through SGD optimizer is 91.41%. Confusion matrix –

The Accuracies and F1 scores are almost same for both training and test sets. There is lot of variation of these scores for intermediate epochs in test set. AUC curve also varies a lot for the test set. Since SGD has higher learning rate compared to ADAM, the time taken is less.

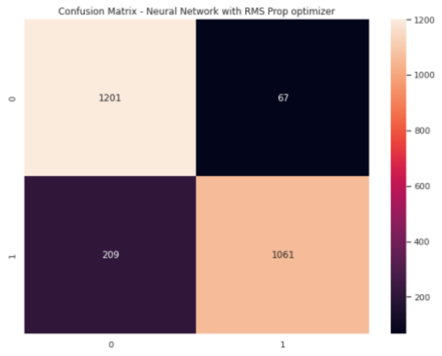




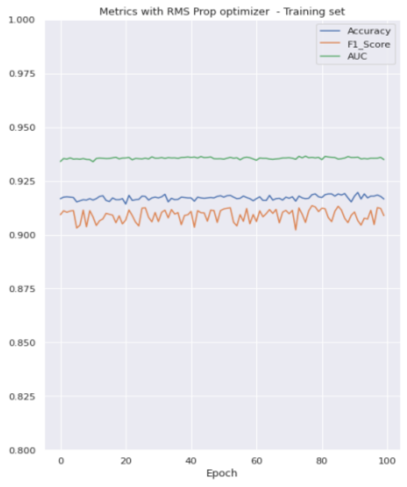


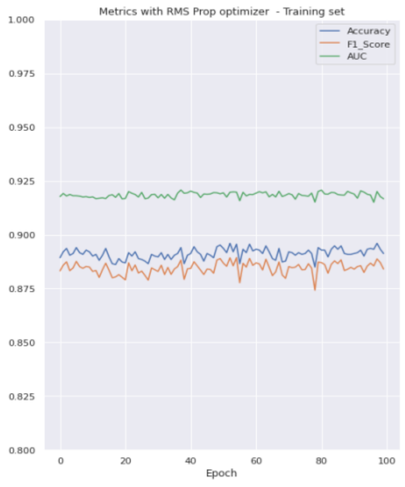
Case – 3: RMS Prop optimizer

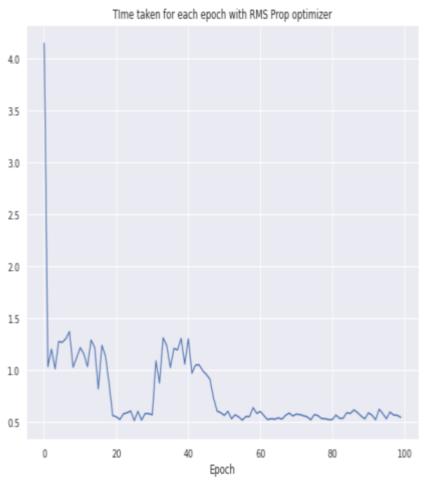
RMSProp Class: tf.keras.optimizers.RMSprop (learning_rate=0.001, rho=0.9, momentum=0.0, epsilon=1e-07, centered=False, name="RMSprop", **kwargs) The RMS Prop algorithm maintains moving average of square of gradients then divides gradient by root of this average. This algorithm uses the plain momentum but not Nesterov momentum. This algorithm combines the idea of only using the sign of the gradient with the idea of adapting the step size separately for each weight. The accuracy obtained through this optimizer is 89.12%. Confusion Matrix –



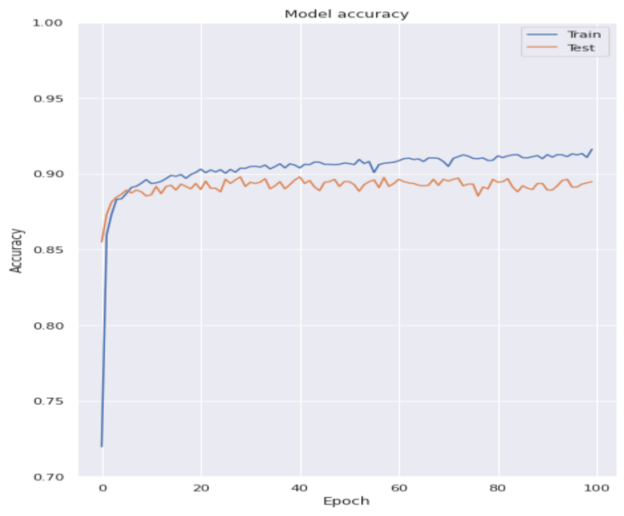
Metrics plot and time plot for RMSProp optimizer -

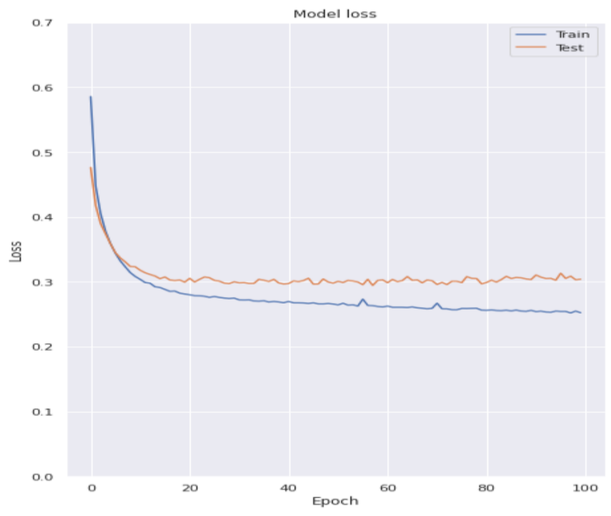






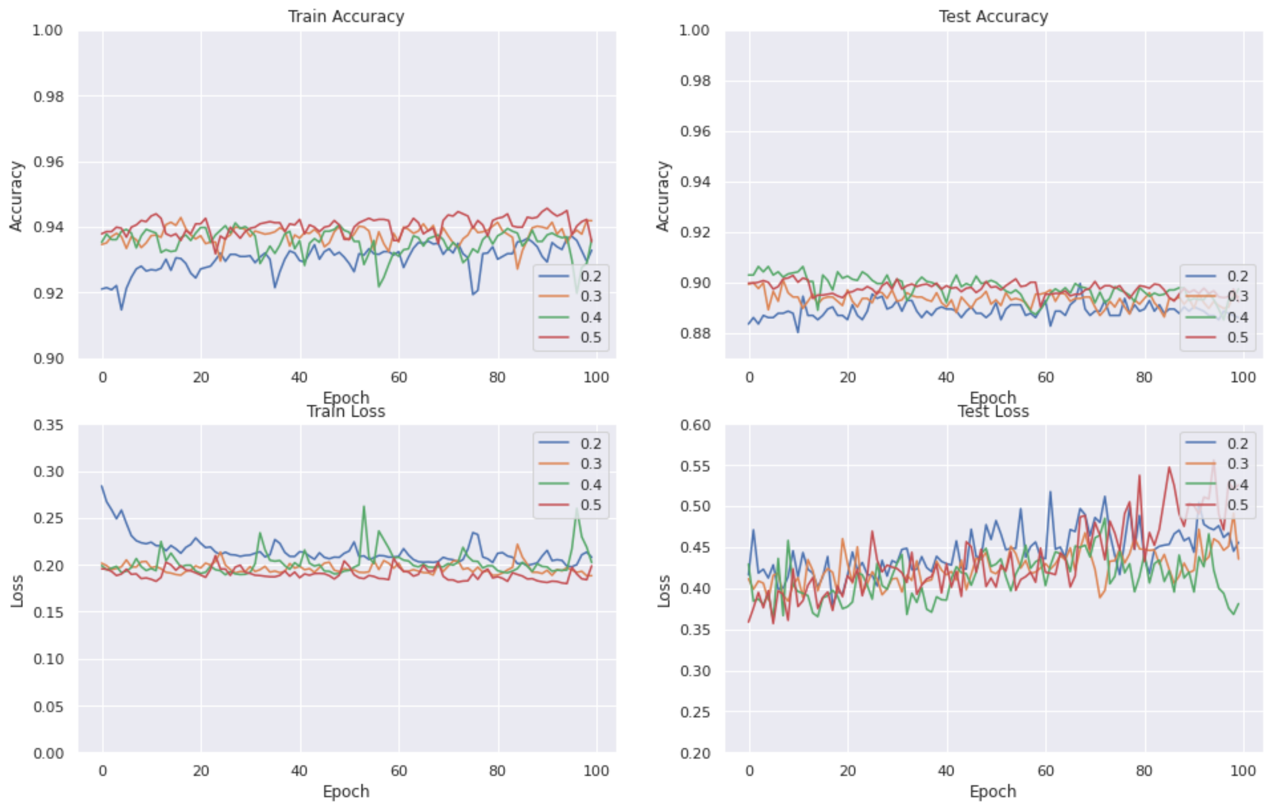
Since we found ADAM optimizer to be best for this dataset, the accuracy changes and loss changes are plotted for both training and test sets. With comparatively little gap between training and test sets curves, it is understood that there is no overfitting in the data. In the last few epochs there is very little change in accuracy. This indicates that model is converged. The same can be interpreted from the loss curve. It also indicates that there is nearly no overfitting as similar performance is seen on both training & test sets. Since there are no big fluctuations of loss through the epochs, we can make sure that model has learned well on the dataset and the batch size provided is optimum and the model is not stuck at local minima.





Stratified K-fold CV – In Stratified k-fold, the distribution of classes remains same in each split. This is recommended for imbalanced dataset. The dataset is split into 10 parts. The accuracy is found to be 91.49% with standard deviation of 0.16%. This shows that there is no overfitting in the model. The model accuracy obtained is not by chance and the same accuracy is obtained on all the 10 different models developed through Stratified k-fold CV.

For different test sizes, the variation in accuracies and losses is plotted against epochs. This shows that best split compared to others considering balance between accuracies and losses is 0.3 which was used to split.



Conclusion

The best model in Regression, SVM, Decision Tree, Neural Network is Neural Network (ADAM optimizer) with accuracy 91.81%. The next is Decision Tree with 91.68% accuracy. 3rd is SVM with 90.66%. Last one is Linear regression model with 55% accuracy. Few reasons why neural networks produce best models compared to other algorithms is due to their ability to understand the non-linearity in the dataset efficiently. They do not impose restrictions on the distribution of input variables. Hence we got best accuracy in model developed through Neural Network algorithm.

With different activation functions, the best accuracy obtained is 92.11%. However, Neural Network apart from having many advantages has a few limitations. They are black-boxes and one doesn't have control over the weights generated. Also, these are slow and require many epochs to converge. It becomes difficult when there is large number of layers. Also, activation function 'ReLU' is not zero-centered. It suffers from 'dying ReLU' problem. Since all the output for negative numbers is zero, it causes nodes to die completely and not learn anything. We can use Leaky ReLU or parametric ReLU instead.

If the data is imbalanced instead of accuracy, we can look at F1 scores, based on the important class (false positive or false negative) and try to reduce the costlier mistakes through neural networks. Better results can be brought by implementing Random Forest algorithm, XGBoost which is suitable for datasets with discrete variables