Introduction

The given data set pertains to the historical demand for bike-sharing in Seoul. The data contains the count of rented bikes on every hour of a given day from December 2017 to November 2018. This data also provides different predictors that could explain the variance in demand of bikes for rent such as temperature, humidity, wind speed, Visibility, Dew point, Solar Radiation, Rainfall, snowfall, season, and whether the day is a holiday or functioning day. The objective is to predict how the bike renting going to be give the features or conditions such as climatic and date and season etc., so that proper planning can be done by the business teams.

Date pre-processing:

Basic primary preprocessing of data is done such as scaling and hot encoding to improve the data structure and eliminating correlated features to improve accuracy and performance.

Binary classification threshold:

We can treat this problem as binary classification problem by converting the output variable ie., count of bikes into high or low. As the data is right skewed, we can take the median of the count of bikes rented(504.5) as the threshold and consider all the observations with count of bikes more than median as high and observations with count of bikes less than median as low. High is encoded as 1 and low is encoded as 0.

Artificial Neural Networks (ANN)

We can use ANN algorithm to this data set since ANN fits well to data set with very high number of features and observations. ANN can also be used for mostly all complex supervised or unsupervised learnings. Hence ANN will be a great fit for this problem. Different experimentations are carried with the ANN algorithm by altering various parameters. The algorithm is modeled using python with Scikit-learn library.

Experimenting with number of layers:

Usually simple to moderately complex problems can be fitted with just 1 or two layers of neural networks. Some cases of thousands or millions of features may need more layers. In the below experiment, we tried to experiment to see how many number of layers are appropriate for our problem.

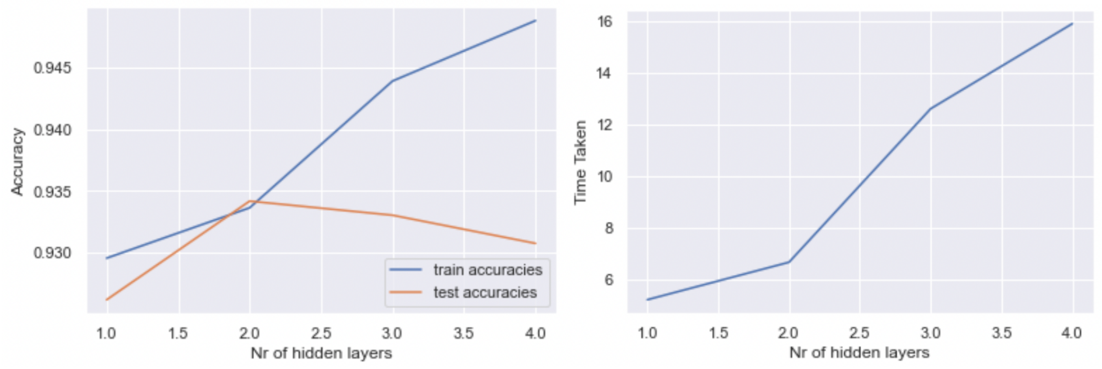


Fig 1.1 – accuracy and time taken as function of layers.

From the above graph we can see that with 2 hidden layers, the train and test accuracies are high and also close to each other which is the ideal case for model. At 1 layer, train and test accuracies are low having higher gap between them. And with 3 and 4 layers, the train accuracy is raising with downfall in test accuracy, which is a case of high variance and is not recommended.

As expected, time taken is increasing as we increase number of layers. We can observe that with 1 layer, time taken is lowest but accuracy is not the best. With 2 layers, the time taken and accuracies are optimum. We can say that 2 layers fits well to this data set and 2 layers are used in further experiments.

Experimenting with number of nodes:

Since we are clear that 2 layers are apt for this problem, now we experiment on number of nodes in each layer. 7 cases are constructed on number of nodes in each layer which are (2,2),(5,5),(10,10),(15,15),(20,20),(25,25),(30,30) for each case from 1 to 7 respectively.

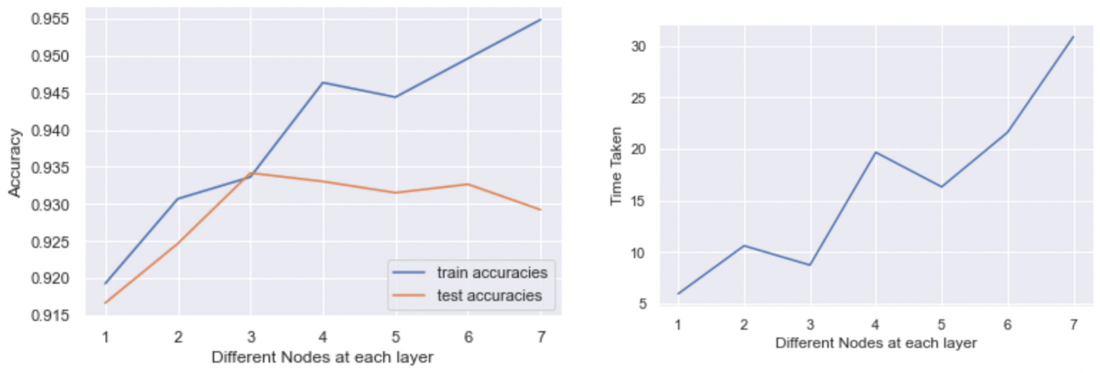


Fig 1.2 – accuracy and time taken as function of cases of number of nodes.

From the above figure, we can observe that at third case which is (10,10) nodes the train and test accuracy is optimum with lease gap between them. For cases from 4 to 7, though train accuracy is high, test accuracy is going down which is indicating the increase in variance. Time taken for case 3 is also second lowest of all cases after case-1. But case-1 has lease accuracies hence can't be selected. Hence for this problem we can conclude that 2 layers with 10 nodes in each layer is best for building neural network.

<u>Experimenting with Activation functions:</u>

At each node an activation function is used to determine the y values so that they are fired when crossed a threshold. Primarily there 4 types of activation functions such as identity, logistic, tanh, relu. Below experimentation is carried to find the best function that fit to our problem.

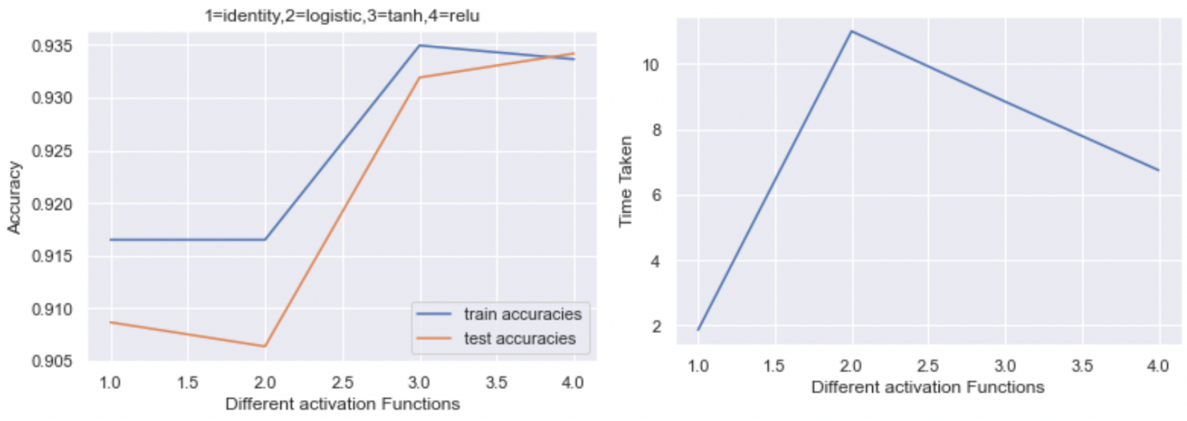


Fig 1.3 – Accuracy & time taken as function of Activation functions.

Above figure depicts that if relu function is used, the accuracies are highest and also the gap between train and test accuracy is also minimum which is best choice. For logistic function though train accuracy is higher, test accuracy is lower compared to relu causing higher gap between train and test which will lead to variance and should be avoided.

Also, time taken Is least for identity function and highest for logistic function. And comparatively low for relu function. Considering the accuracy and time taken, it is worth to spend more time on relu compared to identity function.

Experimenting with different solvers:

Different solvers can be used to converge the neural networks problem. Most important and widely used of them are SGD(Stochastic Gradient Descent), adam, lbfgs. Below experiment is done on different solvers to check accuracy and clock speeds.

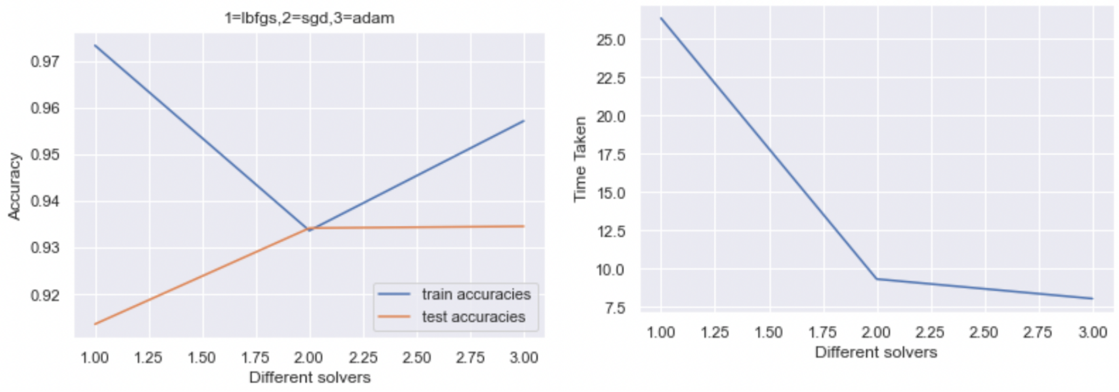


Fig 1.4 – Accuracy and time taken as function of Solvers.

From the above figures 1.4 we can observe that sgd classifier is performing best with optimum train and test accuracies with low gap and comparatively good clock speed compared to ibfgs. Adam's clock speed is highest compared to other two solvers with good train accuracy but gap between train and test accuracy is higher than sgd which leads to high variance and not to be selected.

Experimenting with learning_rate_init:

By giving learning rate manually as hyper parameter, convergence can be optimized for the algorithm, different learning rates are experimented with by taking 10^-6, 10^-5, 10^-4, 10^-3, 10^-2, 10^-1.

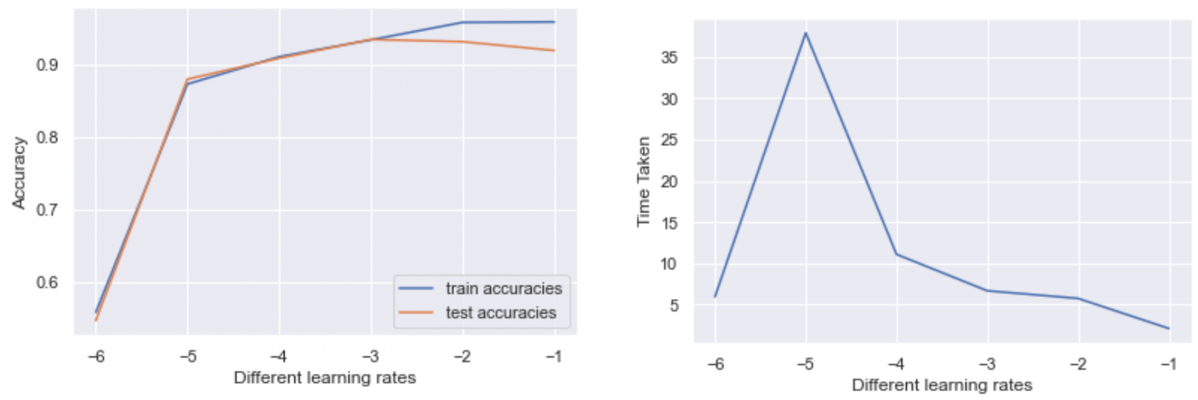


Fig 1.5 – Accuracy and time taken as function of learning rates.

From the above figure 1.5 is evident that 10^-3 is optimum learning rate with optimum accuracy and clock speed. As train accuracy improved for 10^-2 and 10^-1, the gap between the train and test accuracies grew as well which is not ideal.

Experimenting with threshold:

Experimentation is conducted on threshold which specifies the value for the partial derivatives of the error function as stopping criteria. Different values considered are 10^-3, 10^-6, 10^-9, 10^-12, 10^-15, 10^-3, 10^-18, 10^-21, 10^-24.

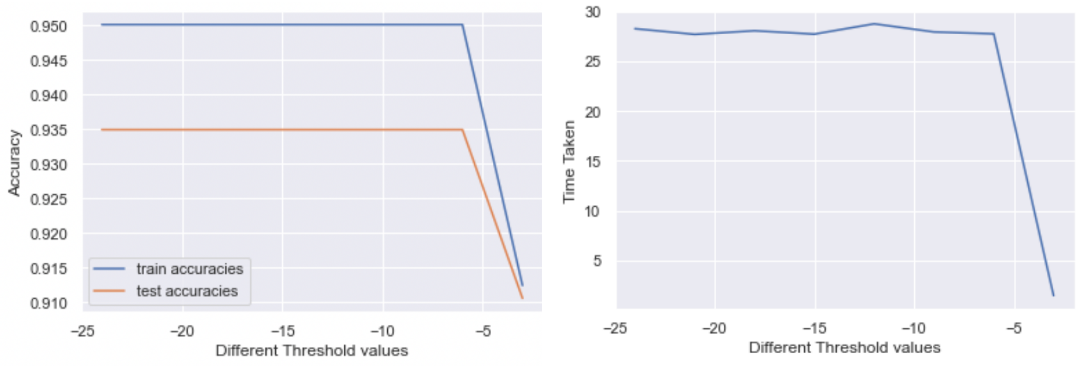
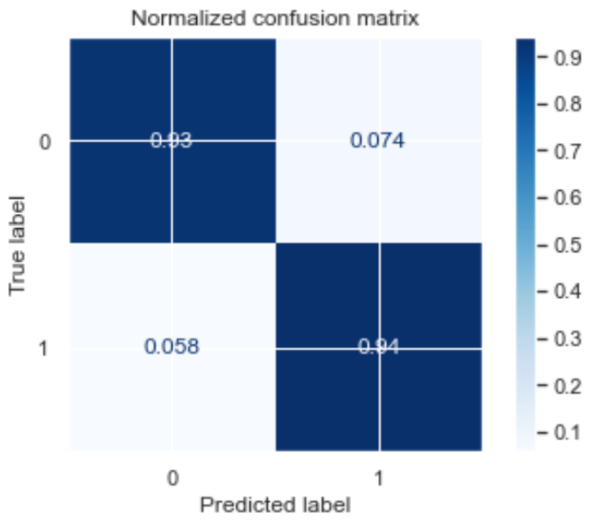


Fig 1.6 – Accuracy and time taken as function of thresholds.

From the above figure 1.6, we can see that there is no much improvement in accuracy or speed when threshold taken is less than 10^-6. Hence, we can consider 10^-4 as baseline.

Final fully tuned model: -

After getting the inputs from the experiments, final model is tuned with the above parameters with the below results



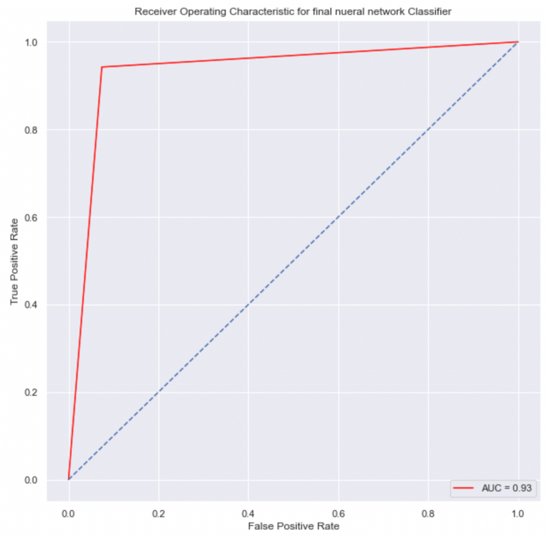


Fig 1.12 –confusion matrix and ROC curve for best tuned model.

The optimized model has train and test accuracies of 0.93 and 0.94 and ROC curve can be seen going further towards top left corner, which is a great indicator for a good classifier.

Cross validation:-

K- fold Cross validation can be used to understand how well the data performs on the new unseen data. Data is divided into k subsets and training is done on all the folds excluding 1 and testing is done on the excluded fold. This process is repeated till all the folds are tested and trained upon. Below experiment is done on different Ks' of cross validation.

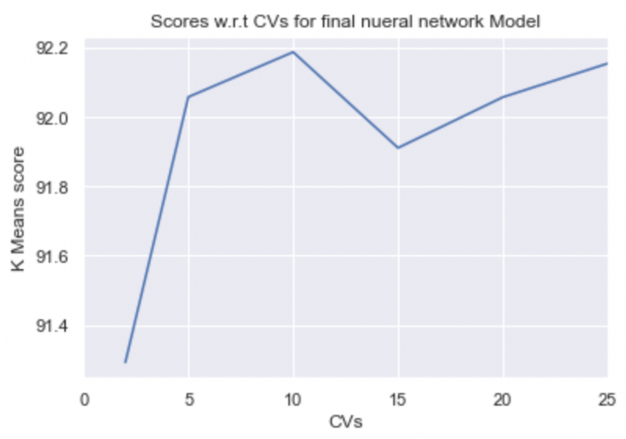
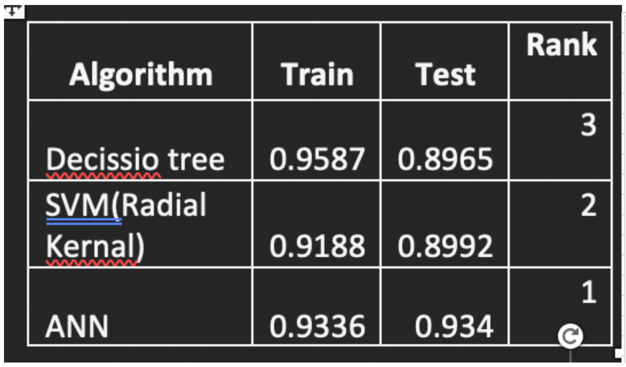


Fig 1.13 –CV score as function on number of Ks.

From the above figure, we can see that cross validation accuracy is between 91% to 92% which is in line with our optimized neural network model's accuracy. We can say that the model performs equally good on the new and unseen data.

Algorithm comparison: -

Following are the train and test accuracies of the models implemented in this this and previous assignment.



Algorithm	Train	Test	Rank
Decissio tree	0.9587	0.8965	3
SVM(Radial Kernal)	0.9188	0.8992	2
ANN	0.9336	0.934	1

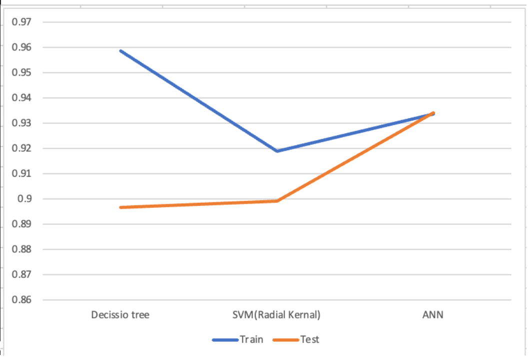


Fig 1.14- Model comparison.

From the above we can see that performance is best with neural networks with optimum accuracy on train and test. Though decision tree have higher train accuracy, test accuracy is low leading to case of high variance. SVM's train and test accuracies are lower than neural network model.