Objective:

The goal of this assignment is to implement 2 learning algorithms: Artificial Neural Networks and K Nearest Neighbors by using built-in function in python and also implementing cross-validation using the above algorithms. 2 datasets mentioned below are going to be used in this assignment.

Dataset 1: Seoul Bike Sharing Demand Data Set

Currently Rental bikes are introduced in many urban cities for the enhancement of mobility comfort. It is important to make the rental bike available and accessible to the public at the right time as it lessens the waiting time. Eventually, providing the city with a stable supply of rental bikes becomes a major concern. The crucial part is the prediction of bike count required at each hour for the stable supply of rental bikes. The dataset contains weather information (Temperature, Humidity, Windspeed, Visibility, Dewpoint, Solar radiation, Snowfall, Rainfall), the number of bikes rented per hour and date information.

Data Pre-processing:

1) Data has been converted into a binary classification by calculating the median value of column "Rented Bike Count" and at point in time, if the count of the rented bikes is greater than the median value (504.5) then, the column MedianBikes is 1 else 0. MedianBikes will act as the target variable.
2) Column Rented Bike Count has been deleted due to its exact collinearity with the new column MedianBikes.
3) All the categorical valued columns have been converted into their respective dummy variables.
4) Feature Scaling has been done for all independent variables.

Dataset 2: Cardiotocography Data Set

The dataset consists of measurements of fetal heart rate (FHR) and uterine contraction (UC) features on cardiotocograms classified by expert obstetricians. 2126 fetal cardiotocograms (CTGs) were automatically processed and the respective diagnostic features measured. The CTGs were also classified by three expert obstetricians and a consensus classification label assigned to each of them. Classification was both with respect to a morphologic pattern (A, B, C. ...) and to a fetal state (N, S, P). Therefore, the dataset can be used either for 10-class or 3-class experiments.

Data Pre-processing:

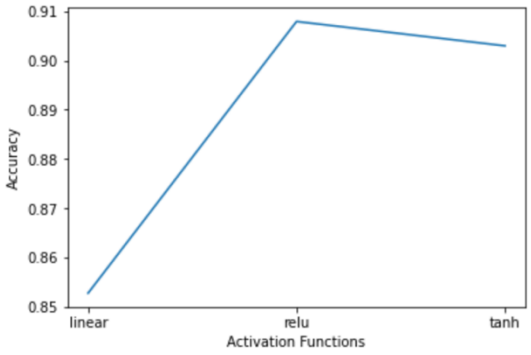
1) There are no missing values and categorical data values, so there was no need for dummy variables or data manipulation or dealing with null values.
2) Feature Scaling has been done for all independent variables.

Both the datasets have been divided into training and testing datasets in the ratio of 7:3.

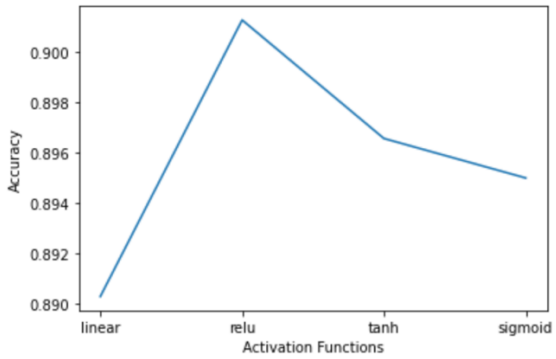
Tasks 1: ANN Model

Keras API with TensorFlow as it's backbone in python was used to carry out all the experiments.

i) **Finding the best activation function in the hidden layers:**
Initial Number of hidden layers = 2
Initial Number of neurons in each layer = 12
Output layer is set to sigmoid activation because we are doing a binary classification in Dataset – 1
Output layer is set to softmax activation because we are doing a multi class classification in Dataset - 2

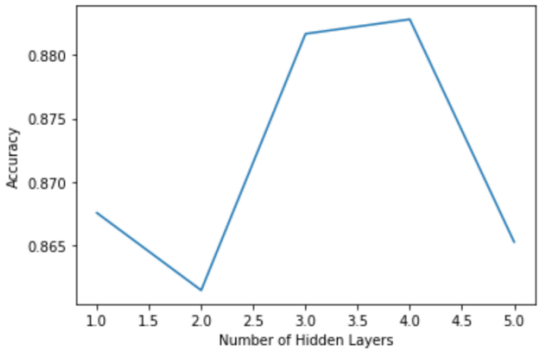


Dataset-1: We can observe that relu activation function yields the best accuracy of about 91%

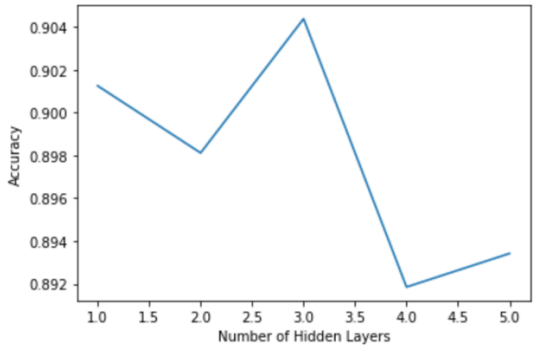


Dataset-2: We can observe that relu activation function yields the best accuracy of about 90%

ii) Finding best number of hidden layers

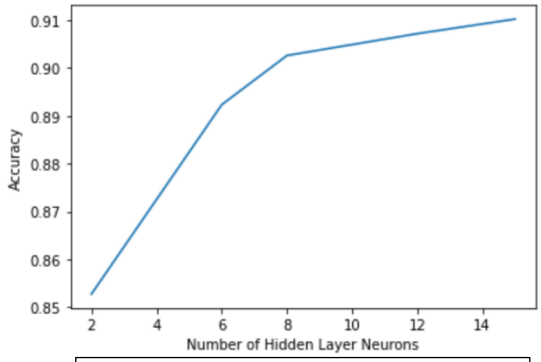


Dataset-1: A model with 4 hidden layers performs the best in classifying the data

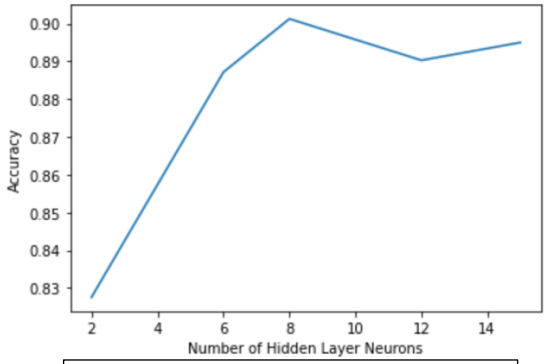


Dataset-2: A model with 3 hidden layers performs the best in classifying the data

iii) Finding best number of neurons in each hidden layer

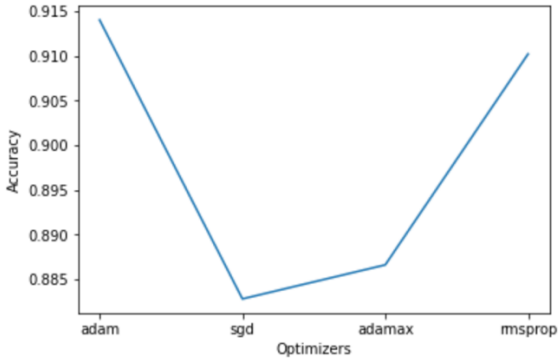


Dataset-1: 15 neurons in each of the hidden layers has the highest accuracy. However, in order to decreases overfitting, I would choose 12 neurons to be the best.

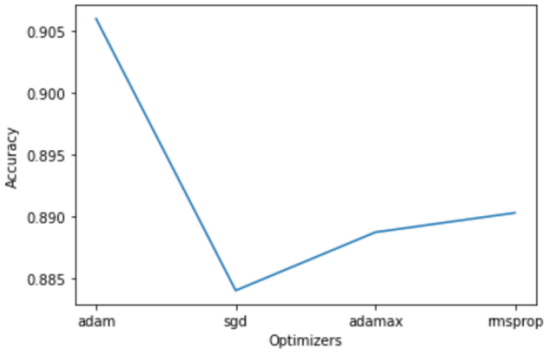


Dataset-2: 8 neurons in each of the hidden layers has the best accuracy of 90%

iv) Finding the best optimizer

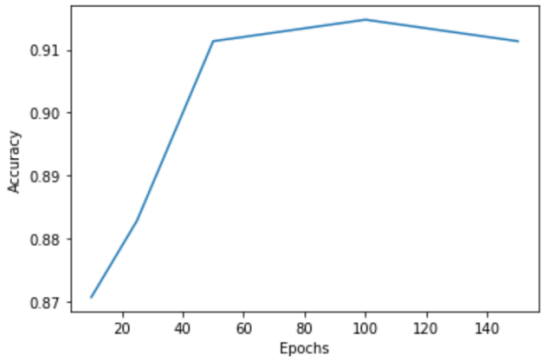


Dataset-1: We can observe that adam optimizer performs the best and sgd optimizer performs the worst

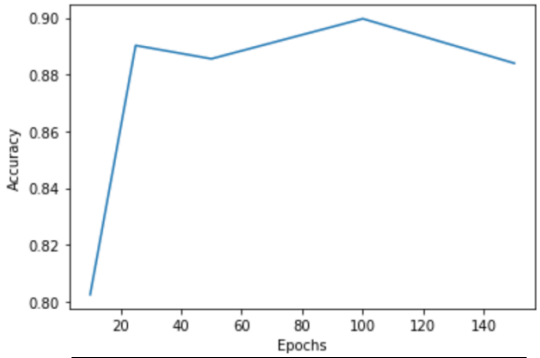


Dataset-2: We can observe that adam optimizer performs the best and sgd optimizer performs the worst

v) Finding best number of Epochs

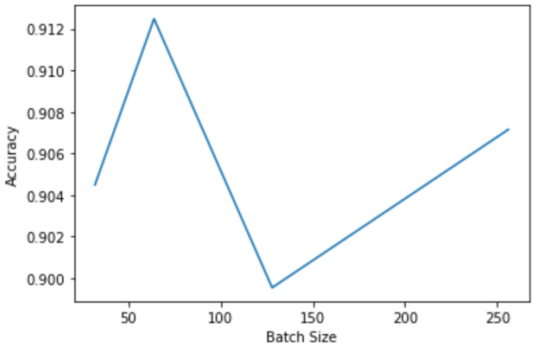


Dataset-1: Anywhere from 40 to 150 epochs gives us similar results however 100 epochs gives us best accuracy

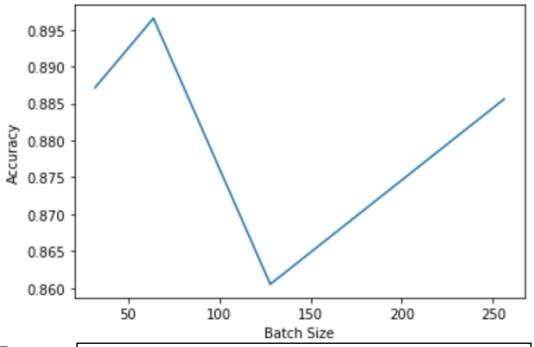


Dataset-2: Anywhere from 45 to 150 epochs gives us similar results however 100 epochs gives us best accuracy

vi) Finding best batch size



Dataset-1: Batch size of 64 yields the best accuracy results compared to other batches



Dataset-2: Batch size of 64 yields the best accuracy results compared to other batches

vii) Best Models Results with the above best parameters

Dataset	# Hidden Layers	# Neurons/Layer	Activation	Epochs	Optimizer	Batch Size
Dataset -1	4	12	relu	100	adam	64
Dataset-2	3	8	relu	100	adam	64

You can see the total execution time, model accuracy and the respective confusion matrix below for 2 datasets

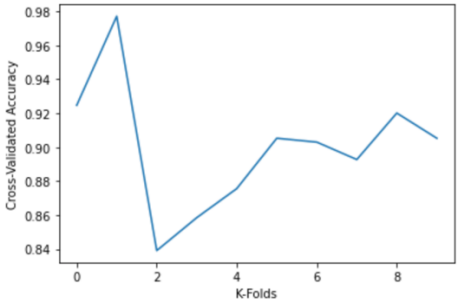
```
Time:  96.75039100646973
Accuracy:  0.9136225266362252

Confusion Matrix
[[1222  119]
 [ 108 1179]]
```

Dataset-1 Results

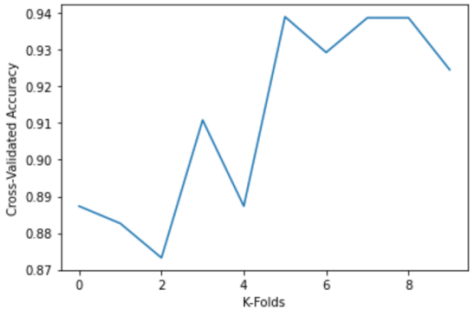
```
Time:  36.133424043655396
Accuracy:  0.896551724137931

Confusion Matrix
[[468  20   6]
 [ 25  57   4]
 [  2   9  47]]
```

Dataset-2 Results

viii) Cross Validation Results



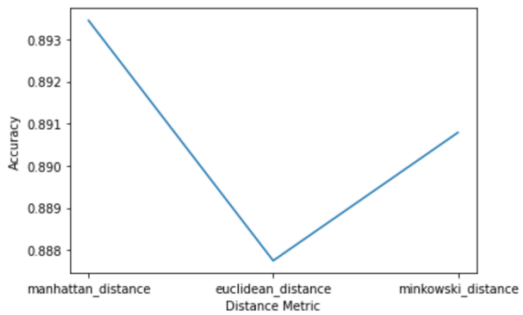
- Accuracies of 10-fold cross validation is plotted for Dataset-1
- The mean average accuracy came out to be 90.01% for this dataset with the best parameters found out through experimentation



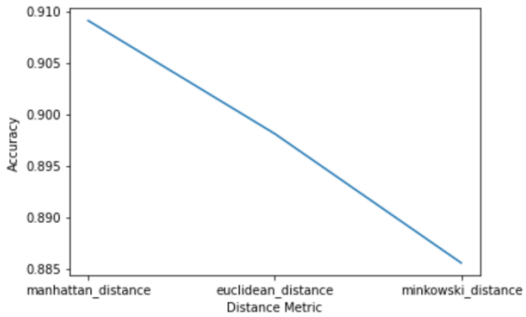
- Accuracies of 10-fold cross validation is plotted for Dataset-2
- The mean average accuracy came out to be 91.11% for this dataset with the best parameters found out through experimentation.

Tasks 2: K-NN Model

i) Finding the Best Distance Metric

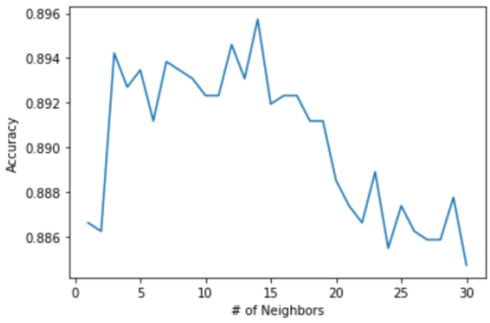


Dataset-1: We can observe that manhatten distance is able to give best accuracy results of around 89%

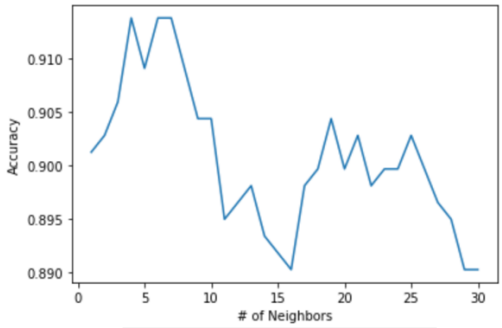


Dataset-2: We can observe that manhatten distance is able to give best accuracy results of around 91%

ii) Finding the best nearest neighbors



Dataset-1: K=14 achieves a highest accuracy of about 91%



Dataset-2: K=4 achieves a highest accuracy of about 91%

iii) Best K-NN Model with the above best parameters

For Dataset 1: K=14 with Manhattan distance gave us the best results from above experiments. Therefore, the final accuracy and confusion matrix which is considered the best for this train and test set split is given below

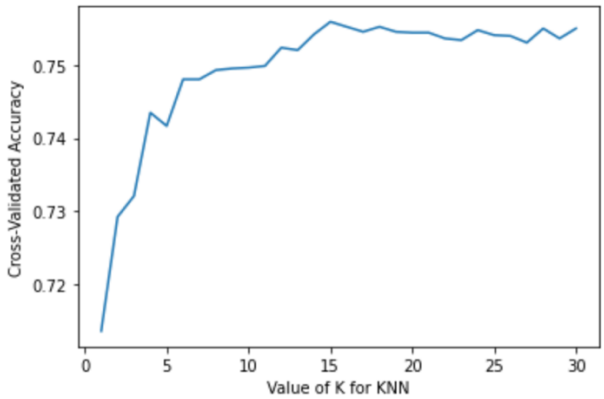
```
Accuracy: 0.895738203957382

Confusion Matrix
[[1257   84]
 [ 190 1097]]
```

For Dataset 1: K=4 with Manhattan distance gave us the best results from above experiments. Therefore, the final accuracy and confusion matrix which is considered the best for this train and test set split is given below

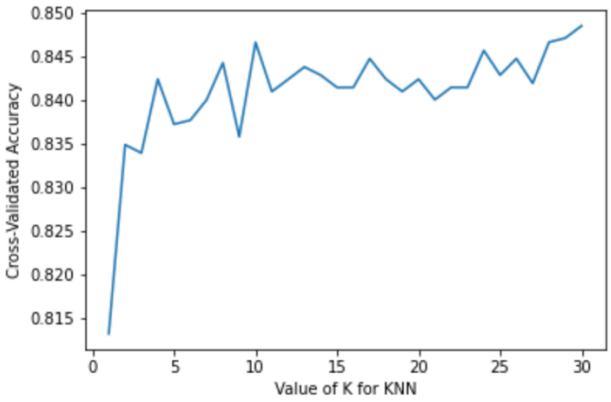
```
Accuracy: 0.9137931034482759

Confusion Matrix
[[489   5   0]
 [ 33  50   3]
 [  5   9  44]]
```

iv) Cross Validation Results

K-fold cross validation was performed on both the datasets with k=10 and below are the resulting mean accuracies for each of the 10 random data splits with different k nearest neighbors value ranging from 1 to 30.



- From this plot we can see that in fact k=15 has the best overall accuracy for the model rather than k=14 (in KNN)
- The K-NN model for Dataset-1 has an accuracy of about 75% which generalizes the data and reduces overfitting



- From this plot we can see that in fact k=10 has the best overall accuracy for the model rather than k=4 (in KNN)
- The K-NN model for Dataset-2 has an accuracy of about 85% which generalizes the data and reduces overfitting

Comparison with Previous Algorithms:

Test Accuracies of Algorithms with best parameters on the Datasets without Cross Validation

Algorithm	Dataset – 1 (Test Accuracy)	Dataset - 2 (Test Accuracy)
SVM	91%	89%
Decision Tree	91%	93.8%
AdaBoost	91.3%	87.9%
ANN	91.4%	89.7%
K-NN	89.6%	91.4%

Accuracies of Algorithms with best parameters on the Datasets with Cross Validation

Algorithm	Dataset – 1 (Test Accuracy)	Dataset – 2 (Test Accuracy)
SVM	77%	86%
Decision Tree	87%	87%
AdaBoost	87%	86.8%
ANN	90%	91.1%
K-NN	75.6%	84.7%

Ranking the Algorithms based on Cross Validation Accuracies

1. **ANN**
2. **Decision Tree**
3. **AdaBoost**
4. **SVM**
5. **KNN**

Conclusion

1) Both the datasets do reasonably well with ANN and KNN classification with around 90% test accuracy in the random split.
2) However, we can observe that ANN performs better for dataset 1 and K-NN performs better for dataset-2.
3) In dataset 1, we could see that there was about 2% decrease in the test accuracy when K-NN was used over ANN but there was about 2% increase in accuracy in dataset 2.
4) The above results are specific to the random test and train split I had initially, however these results do not conclude that none of the models is the best model for both the datasets. It is because, the information from the test data has been leaked into the training set while trying to find the best parameters for the models and I chose only those parameter values that gave best test accuracies in all the models. This also means that the models are overfitting.
5) We could observe that the test accuracy levels decrease when all the algorithms are run using cross validation. This is bound to happen because the best generalized model would have a good Bias and Variance tradeoff. This point has been clearly established with the experiments that I had conducted in my assignment.

6) Only in the case of ANN in Dataset-2 we see that there is an increase in the Cross-validation accuracy as compared to the random split. This might be a case of a pure random chance.
7) It is important to use cross validation for hyperparameter tuning rather than trying it on the same train and test data splits. This reduces overfitting in the model.
8) It is only after performing cross validation on datasets using K-NN model we realized that the actual number of nearest neighbors required to generalize the model is different from the initial experiments.
9) In dataset 1, from my random split I got k=14 as the best value but the cross validation proved that k=15 best generalizes the model with less overfitting and high accuracy.
10) In dataset 2, from my random split I got k=4 as the best value but the cross validation proved that k=10 best generalizes the model with less overfitting and high accuracy.
11) The models perform as per the theory. Naturally the neural networks perform the best for classifying both my datasets when compared to K-NN model.
12) We also see that as the number of epochs increase the accuracy also increase (suggesting overfitting). This also means that we need to invest more time and computational power.
13) In order to decrease the compilation time, we can play around with the batch_size parameter in ANN models.
14) K-NN algorithm is pretty straight forward. Determining the correct distance metric and the right number of neighbors will yield us relevant results.
15) Although K-NN performed well in the initial split, the K-fold cross validation results on both the algorithms in both datasets reveal that neural networks are the best models for classification among the other algorithms that are previously experimented.
16) **Based on the above results I would like to rank my algorithms that are specific to classification in both the datasets: (Based on Cross Validation Mean Accuracy) ANN> Decision Tree/AdaBoost > SVM > K-NN**