Assignment 3

A. General Guidelines about Models used in the Report:

Two different classification learning techniques are used in this report to predict the target classes, namely – k-Nearest Neighbors (kNN) and Artificial Neural Networks (ANN).

A1. k-NN Classifier

It is a parametric classifier where an object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k-nearest neighbors. If k = 1, then the object is simply assigned to the class of that single nearest neighbor.

Grid Search is used to select the best hyperparameters that leads to the best 5-Fold Cross-Validation score. The only hyper-parameter is k (number of neighbors) which is selected using Grid Search that leads to the best performance.

A2. ANN Classifier

Artificial neural networks are networks of neurons which mimic the neural structure of the brain. They process records one at a time and learn by comparing their classification of the record with the known classification of the record.

Neural networks have a lot of hyperparameters to search through to find the best parameters with best metric scores. 5-fold Grid Search Cross Validation is used to select the best hyperparameters in a series of models to construct the final neural network. Following are the key points to note before constructing the neural network:

- For learning algorithm (back propagation), Stochastic Gradient Descent (SGD) is used in all experimental models. SGD is an iterative learning algorithm that uses a training dataset to update a model.
- Sigmoid activation function is used as the output node.
- Binary Cross entropy is used as the loss function. Cross-entropy will calculate a score that summarizes the average difference between the actual and predicted probability distributions for predicting class 1. The score is minimized, and a perfect cross-entropy value is 0.

Following is the order of experimental models:

Model 1: Hyperparameters: Batch size and # of Epochs

The batch size is a hyperparameter of gradient descent that controls the number of training samples to work through before the model's internal parameters are updated. The number of epochs is a hyperparameter of gradient descent that controls the number of complete passes through the training dataset.

Model 2: Activation Function of the 1st hidden layer nodes

Four activation functions are used, namely – Linear or Identity, Sigmoid or Logistic, Tanh or Hyperbolic Tangent, ReLU or Rectified Linear Unit

Model 3: Learning Rate and Momentum of the 1st hidden layer nodes

Learning rate controls how much to update the weight at the end of each batch and the momentum controls how much to let the previous update influence the current weight update.

Model 4: Drop-out Rate and Weight Constraint of the 1st hidden layer nodes

Dropout rate is the percentage of randomly selected neurons which are ignored during training. Their contribution to the activation of downstream neurons is temporally removed on the forward pass and any weight updates are not applied to the neuron on the backward pass. This prevents units from co-adapting too much. In neural networks, co-adaptation refers to when different hidden units in a neural network have highly correlated behavior. This is a primary cause for overfitting. It is better for computational efficiency and the model's ability to learn a general representation if hidden units can detect features independently of each other. Weight constraint is the maximum norm value for the incoming weights. To get good results, dropout is best combined with a weight constraint such as the max norm constraint.

Model 5: # of Neurons in the 1st hidden layer

The number of neurons in a layer is an important parameter to tune. Generally, the number of neurons in a layer controls the representational capacity of the network, at least at that point in the topology.

Model 6: # of Neurons in the 2nd hidden layer

The 2nd hidden layer neurons contain the same properties as the 1st hidden layer neurons. Grid search is done only to find the optimum number of neurons in the 2nd hidden layer.

B. Bike Sharing Dataset:

The Seoul Bike dataset contains information regarding number of bikes rented on an hourly basis for a year and includes prevailing weather conditions. The aim is to predict whether the bike rentals on a particular hour of a day are higher than the median value.

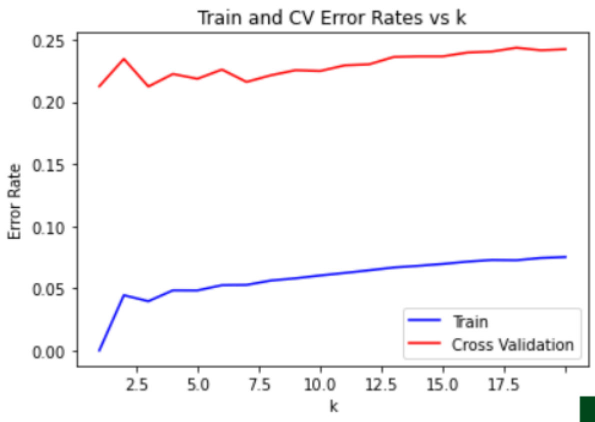
B1. EDA and Data Pre-processing:

Exploratory data analysis has resulted in following implications which are considered for modelling:

- The 'hour' variable is considered as a categorical variable and is included in our models.
- We do not need to include 'functioning_day' while training our models. On those days we do not to predict the number of bikes rented.
- The dataset does not contain day of week as variable. Hence a new variable is created to include in machine learning models.
- Dew point temperature is highly correlated with temperature (0.91) and including both these variables might lead to multi-collinearity issues while modelling. In addition, the VIF of all variables is less than 10 after removing dew-point temperature. Hence, dew point temperature is not included in the model building.
- The dataset is balanced.

B2. k-NN Classification

Hyperparameter: k = [1 to 20]

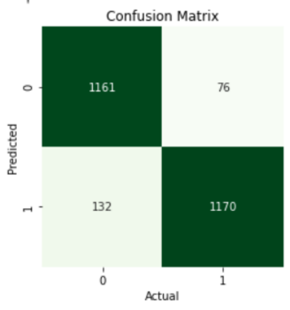


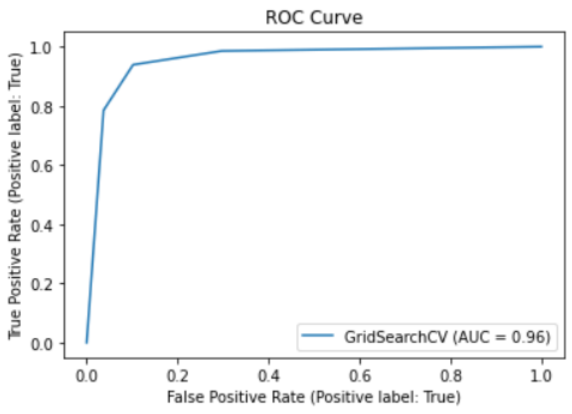
As k increases, the error rate of train set increases and the error rate of CV set remains the same, suggesting slight underfitting. k = 1 suggests that the model has zero train error. But this has high CV error rate.

Chosen parameters: k = 3. The average CV accuracy is 0.79.

The k-NN classifier does a pretty good job in classifying the target variable as test accuracy is about 0.92. The area under the curve suggests that it is a very good classifier.

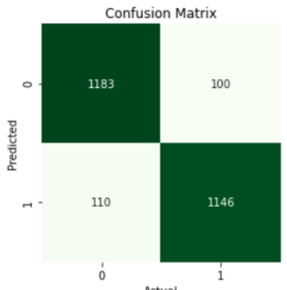
```
Accuracy: 0.92
Sensitivity: 0.94
Specificity: 0.90
```





B3. ANN Classification

Model 1: Hyperparameters: Batch size and # of Epochs



```
Accuracy: 0.92
Sensitivity: 0.92
Specificity: 0.91
```

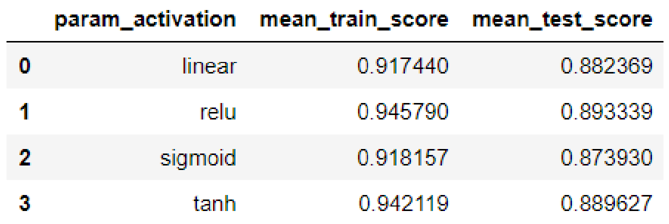
Hyperparameter: Batch Size = [10, 20, 50] and # of Epochs = [10, 50, 100]

The network has sigmoid function as the activation function for the 10 nodes in the first hidden layer and for the output node.

Chosen parameters: Batch Size = 10 and # of Epochs = 100. The average CV accuracy is 0.87.

Generally, as the number of epochs increase, the weights in the neural network are modified more frequently and the curve goes from underfitting to optimal to overfitting curve. But for this dataset, 100 epochs and a batch size of 10 is best suited.

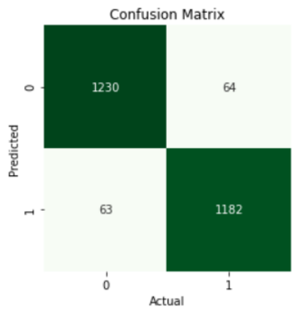
Model 2: Activation Function of the 1st hidden layer nodes



	param_activation	mean_train_score	mean_test_score
0	linear	0.917440	0.882369
1	relu	0.945790	0.893339
2	sigmoid	0.918157	0.873930
3	tanh	0.942119	0.889627

Hyperparameter: Activation = [Linear, Sigmoid, Tanh, ReLU]

The network has sigmoid function as the activation function for the 10 nodes in the first hidden layer and for the output node.



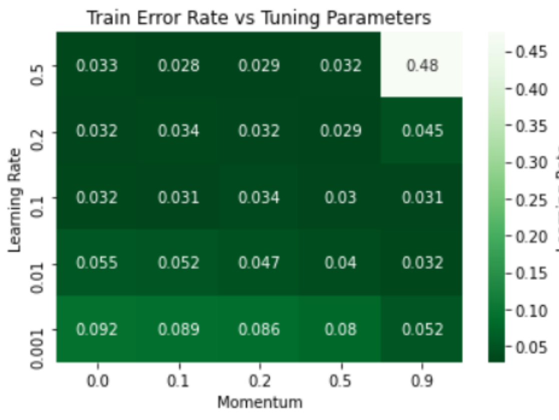
Chosen parameters: Activation Function = ReLU. The average CV accuracy is 0.89.

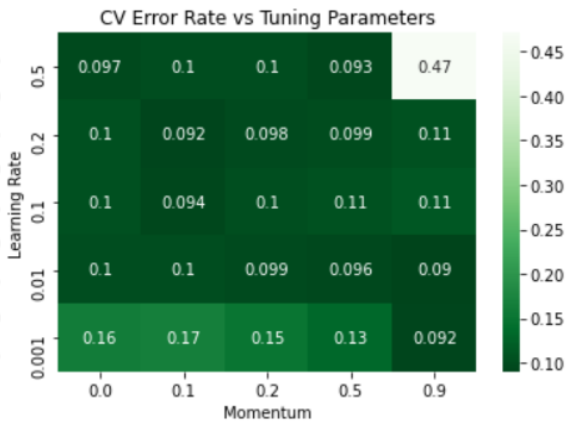
The performance on test set is very good with 0.95 as accuracy, sensitivity, and specificity. ReLU activation function in hidden layer generally leads to best classification results.

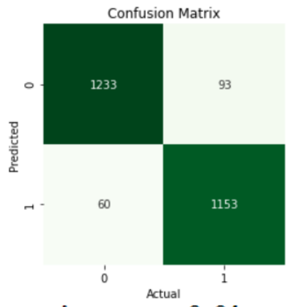
```
Accuracy: 0.95
Sensitivity: 0.95
Specificity: 0.95
```

Model 3: Learning Rate and Momentum of the 1st hidden layer nodes

Hyperparameter: Learning Rate = [0.001, 0.01, 0.1, 0.2, 0.5] and Momentum = [0.0, 0.1, 0.2, 0.5, 0.9]







```
Accuracy: 0.94
Sensitivity: 0.93
Specificity: 0.95
```

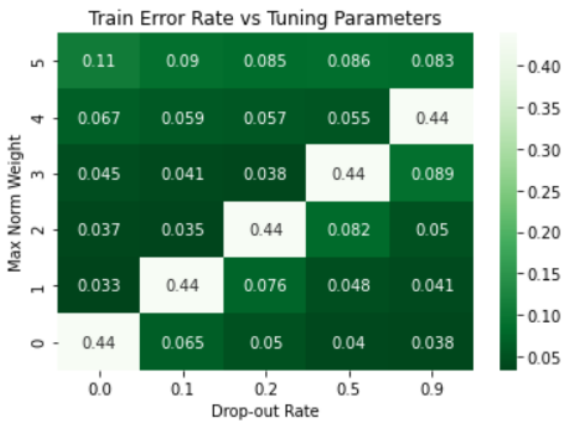
The network has ReLU function as the activation function for the 10 nodes in the first hidden layer and sigmoid function as output node.

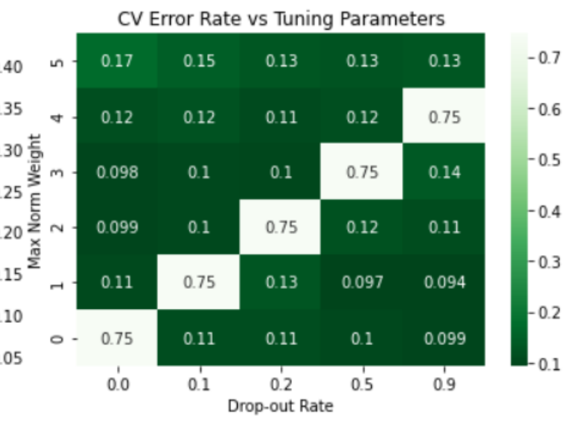
Chosen parameters: Learning Rate = 0.01 and Momentum = 0.9. The average CV accuracy is 0.91.

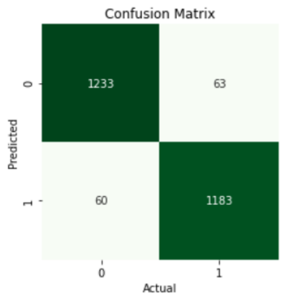
Generally, larger learning rates have a significant underfitting effect and smaller learning rates tend to converge after a greater number of iterations. Since the cost function involved in neural networks generally have local minima, a right amount of momentum is required to find the global minimum. The test set performance is comparable to Model 2.

Model 4: Drop-out Rate and Weight Constraint of the 1st hidden layer nodes

Hyperparameter: Drop-out Rate = [0.001, 0.01, 0.1, 0.2, 0.5] and Max Norm Weight = [0.0, 0.1, 0.2, 0.5, 0.9]







```
Accuracy: 0.95
Sensitivity: 0.95
Specificity: 0.95
```

The network has ReLU function as the activation function for the 10 nodes in the first hidden layer and sigmoid function as output node. The learning rate and momentum are 0.01 and 0.9, respectively.

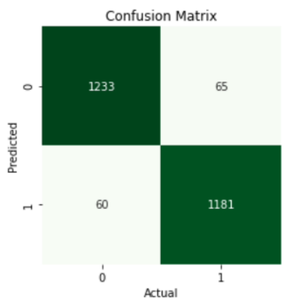
Chosen parameters: Drop-out Rate = 0.1 and Max Norm Weight = 3. The average CV accuracy is 0.91.

Compared to the previous models, the model with 10% of drop-out rate and max norm weight of 3 performed a little better in terms of creating a balance between sensitivity and specificity.

Model 5: # of Neurons in the 1st hidden layer

Hyperparameter: # of neurons = [1, 5, 10, 20, 25, 30]

	param_neurons	mean_train_score	mean_test_score
0	1	0.912293	0.879838
1	5	0.944482	0.888276
2	10	0.959163	0.898909
3	15	0.967010	0.910892
4	20	0.970806	0.894014
5	25	0.976797	0.901102
6	30	0.979750	0.907685



```
Accuracy: 0.95
Sensitivity: 0.95
Specificity: 0.95
```

The network has ReLU function as the activation function in the first hidden layer and sigmoid function as output node. The learning rate and momentum are 0.01 and 0.9, respectively. The network architecture has 10% of drop-out rate and weight constraint of max norm 3.

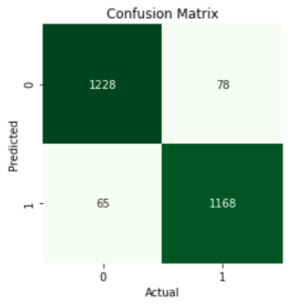
Chosen parameters: # of Neurons = 15. The average CV accuracy is 0.91.

Compared to the previous model with 10 nodes in the 1st hidden layer, the present model with 15 nodes resulted in similar overall performance. Number of neurons greater than optimum has resulted in overfitting and lower than optimum has resulted in underfitting.

Model 6: # of Neurons in the 2nd hidden layer

Hyperparameter: # of neurons = [1, 5, 10, 20, 25, 30]

	param_neurons	mean_train_score	mean_test_score
0	1	0.967009	0.902453
1	5	0.964689	0.897559
2	10	0.966335	0.901103
3	15	0.966756	0.903971
4	20	0.967305	0.900596
5	25	0.966967	0.910386
6	30	0.967221	0.896545



```
Accuracy: 0.94
Sensitivity: 0.94
Specificity: 0.95
```

The network has ReLU function as the activation function in the first hidden layer and sigmoid function as output node. The first hidden layer has 15 ReLU neurons. The learning rate and momentum are 0.01 and 0.9, respectively. The network architecture has 10% of drop-out rate and weight constraint of max norm 3.

Chosen parameters: # of Neurons = 25. The average CV accuracy is 0.91.

Comparing the CV accuracy in model 5 and model 6, adding a new hidden layer has not improved the performance significantly. Therefore, one hidden layer with 15 neurons is enough to produce the best classification results.

Final Model:

```
# def final_model():
#     model = Sequential()
#     model.add(Input(shape=(40, )))
#     model.add(Dense(neurons = 15, activation = 'relu', kernel_constraint = maxnorm(3)))
#     model.add(Dropout(0.1))
#     model.add(Dense(1, activation = 'sigmoid'))
#     optimizer = SGD(lr = 0.01, momentum = 0.9)
#     model.compile(loss = 'binary_crossentropy', optimizer = optimizer, metrics= ['accuracy'])
#     return model
# annc_model = KerasClassifier(build_fn = final_model, batch_size = 10, epochs = 100, verbose = 0)
```

B4. Comparison of Classifiers

Model	**Hyperparameters**	**5-fold CV Accuracy**	**Test Accuracy**	**Test Sensitivity**	**Test Specificity**
Logistic Regression			0.92		
SVM (Linear)	C=1.0	0.84	0.92	0.94	0.90
SVM (Radial)	C=10.0, gamma=0.01	0.85	0.92	0.94	0.90
SVM (Poly)	C=1.0, degree=1, gamma=1.0	0.84	0.92	0.94	0.90
Decision Tree	criterion=gini, max_depth=8, min_samples_leaf=1	0.82	0.92	0.95	0.90
CCP Decision Tree	ccp_alpha=0.0050	0.83	0.90	0.91	0.89
AdaBoost D-Tree	learning_rate=0.1, n_estimators=1000	0.86	0.92	0.93	0.91
k-NN	k=3	0.79	0.92	0.94	0.90
ANN	batch_size=10, epochs=100 **Hidden Layer 1:** neurons=15, activation=ReLU, dropout=0.1, weight_constraint=maxnorm(3), **Output Layer:** neurons=1, activation=sigmoid **SGD Optimizer:** learn_rate=0.01, momentum=0.9	**0.90**	**0.95**	**0.95**	**0.95**

C. Chronic Heart Diseases Dataset:

C1. EDA and Data Pre-processing:

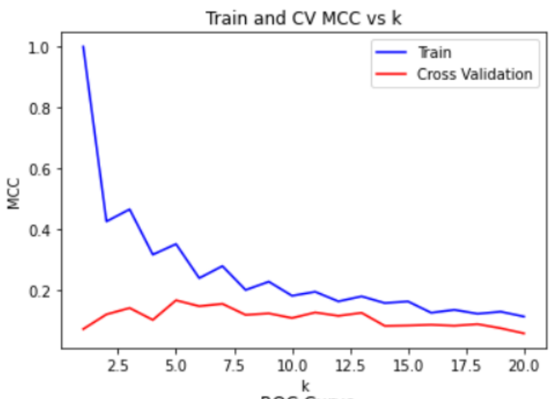
The chronic heart diseases dataset contains information regarding the chance of contracting a chronic heart disease after 10 years based on following parameters:

- Demographic: male, age, education
- Behavioral: current_smoker, cigs_per_day
- Medical (history): bp_meds, prevalent_stroke, prevalent_hyper, diabetes
- Medical (current): tot_chol, sys_bp, dia_bp, bmi, heart_rate, glucose

None of the continuous variables are correlated as the highest VIF is less than 10. Hence, we use all the variables in modelling. The distribution of most numerical variables looks the same for both the classes except 'age'.

The dataset is unbalanced with 3101 negative class records out of 3658. In such a case, model evaluation using accuracy would be misleading. One needs to investigate sensitivity and specificity or misclassification costs to evaluate and select models. For such scenarios, Matthews Correlation Coefficient (MCC) is a better model scoring method. It considers true and false positives and negatives and is generally regarded as a balanced measure which can be used even if the classes are of very different sizes. The MCC is in essence a correlation coefficient value between -1 and +1. A coefficient of +1 represents a perfect prediction, 0 an average random prediction and -1 an inverse prediction. The statistic is also known as the phi coefficient. Suitable class weights must be given to both target variables for better classification.

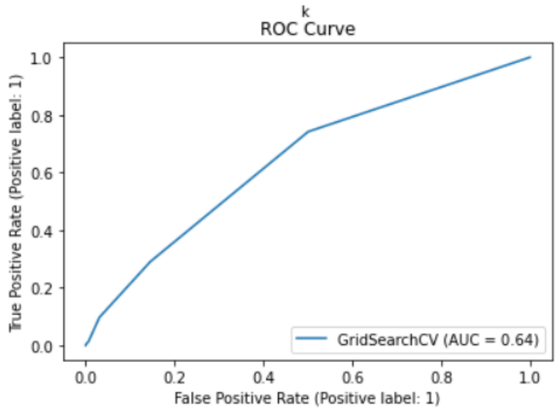
C2. k-NN Classification

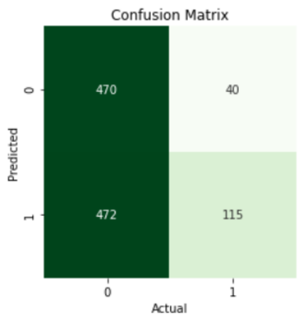


Hyperparameter: k = [1 to 20]

k = 1 suggests that the model has perfect prediction value and has overfit the data. As k increases, the MCC of train set reduces and the MCC of CV set increase slightly and then decreases.

Chosen parameters: k = 5. The average CV MCC is 0.17.



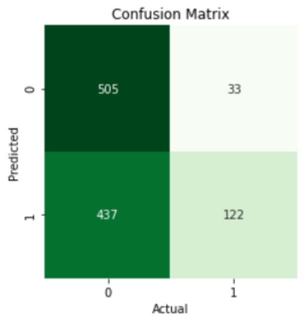


```
Accuracy: 0.53
Sensitivity: 0.74
Specificity: 0.50
```

The k-NN classifier does an okay job in classifying the target variable with an accuracy of 0.53. The area under the curve suggests that it is not a great classifier. This can be attributed to the unbalanced nature of the dataset and the minority class distribution is spread across the majority class distribution.

C3. ANN Classification

Model 1: Hyperparameters: Batch size and # of Epochs



Hyperparameter: Batch Size = [10, 20, 50] and # of Epochs = [10, 50, 100]

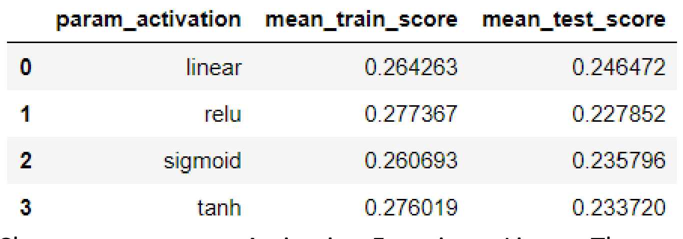
The network has sigmoid function as the activation function for the 5 nodes in the first hidden layer and for the output node.

```
Accuracy: 0.57
Sensitivity: 0.79
Specificity: 0.54
```

Chosen parameters: Batch Size = 50 and # of Epochs = 10. The average CV MCC is 0.25.

Model 2: Activation Function of the 1st hidden layer nodes

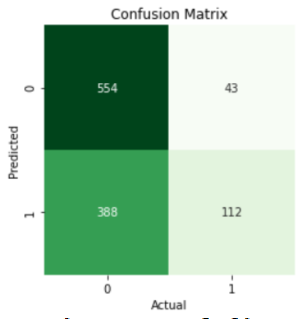
Hyperparameter: Activation = [Linear, Sigmoid, Tanh, ReLU]



	param_activation	mean_train_score	mean_test_score
0	linear	0.264263	0.246472
1	relu	0.277367	0.227852
2	sigmoid	0.260693	0.235796
3	tanh	0.276019	0.233720

The network has sigmoid function as the activation function for the 5 nodes in the first hidden layer and for the output node.

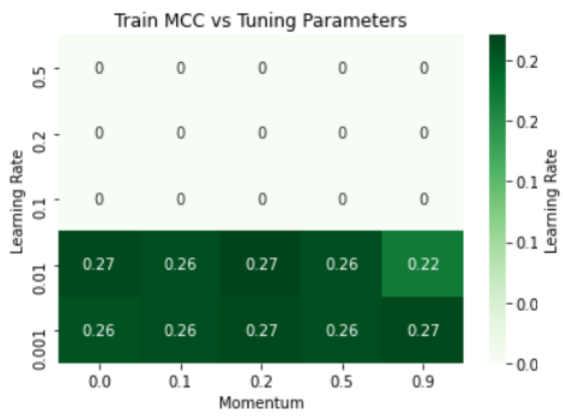
Chosen parameters: Activation Function = Linear. The average CV MCC is 0.25.

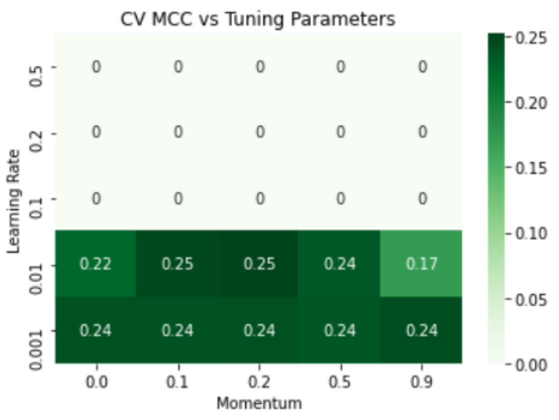


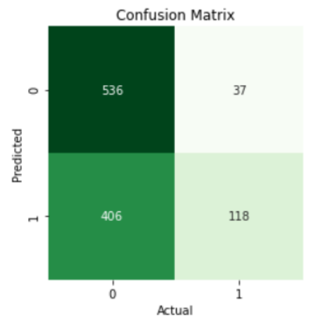
```
Accuracy: 0.61
Sensitivity: 0.72
Specificity: 0.59
```

Model 3: Learning Rate and Momentum of the 1st hidden layer nodes

Hyperparameter: Learning Rate = [0.001, 0.01, 0.1, 0.2, 0.5] and Momentum = [0.0, 0.1, 0.2, 0.5, 0.9]







```
Accuracy: 0.60
Sensitivity: 0.76
Specificity: 0.57
```

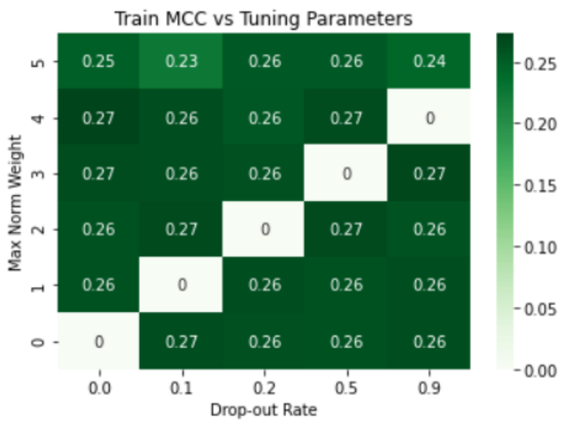
The network has linear function as the activation function for the 5 nodes in the first hidden layer and sigmoid function as output node.

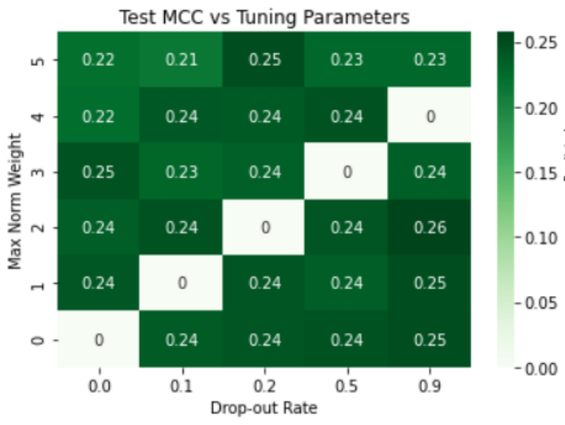
Chosen parameters: Learning Rate = 0.01 and Momentum = 0.2. The average CV MCC is 0.25.

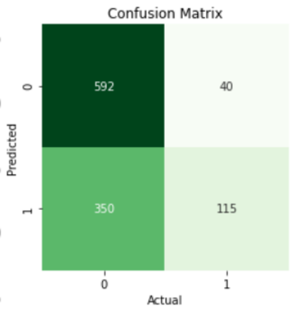
As the learning rate increases, the stochastic gradient descent will diverge from the global minima. Hence selecting lower learning rates have provided better classification metrics. Momentum is 0.2 which suggests that a small amount of push was required to move into global minimum.

Model 4: Drop-out Rate and Weight Constraint of the 1st hidden layer nodes

Hyperparameter: Drop-out Rate = [0.001, 0.01, 0.1, 0.2, 0.5] and Max Norm Weight = [0.0, 0.1, 0.2, 0.5, 0.9]







```
Accuracy: 0.64
Sensitivity: 0.74
Specificity: 0.63
```

The network has linear function as the activation function for the 5 nodes in the first hidden layer and sigmoid function as output node. The learning rate and momentum are 0.01 and 0.2, respectively.

Chosen parameters: Drop-out Rate = 0.2 and Max Norm Weight = 2. The average CV MCC is 0.26.

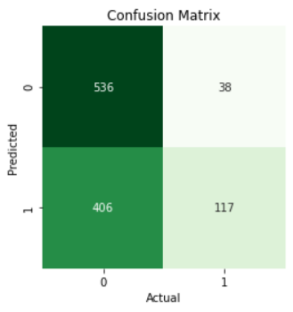
Model 5: # of Neurons in the 1st hidden layer

Hyperparameter: # of neurons = [1, 5, 10, 20, 25, 30]

The network has linear function as the activation function in the first hidden layer and sigmoid function as output node. The learning rate and momentum are 0.01 and 0.2, respectively. The network architecture has 20% of drop-out rate and weight constraint of max norm 2.

Chosen parameters: # of Neurons = 20. The average CV MCC is 0.25.

	param_neurons	mean_train_score	mean_test_score
0	1	0.260579	0.237628
1	5	0.264919	0.234425
2	10	0.268057	0.230486
3	15	0.263449	0.240203
4	20	0.268890	0.250125
5	25	0.260944	0.238977
6	30	0.270872	0.228097



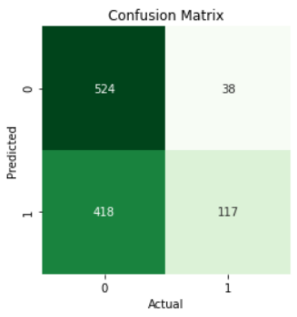
```
Accuracy: 0.60
Sensitivity: 0.75
Specificity: 0.57
```

There is not much improvement in the models with changing the number of neurons in the 1st hidden layer. The performance on test set does not change much.

Model 6: # of Neurons in the 2nd hidden layer

Hyperparameter: # of neurons = [1, 5, 10, 20, 25, 30]

	param_neurons	mean_train_score	mean_test_score
0	1	0.257163	0.241513
1	5	0.257065	0.248608
2	10	0.263906	0.246685
3	15	0.262105	0.256295
4	20	0.258843	0.252780
5	25	0.264989	0.239757
6	30	0.257528	0.243439



```
Accuracy: 0.58
Sensitivity: 0.75
Specificity: 0.56
```

The network has linear function as the activation function in the first hidden layer and sigmoid function as output node. The first hidden layer has 20 linear neurons. The learning rate and momentum are 0.01 and 0.2, respectively. The network architecture has 20% of drop-out rate and weight constraint of max norm 2.

Chosen parameters: # of Neurons = 15. The average CV MCC is 0.26.

Similar to the Bike Sharing dataset, adding a new hidden layer has not improved the performance significantly. Therefore, one hidden layer with 20 neurons is enough to produce the best classification results among the ANN models tested.

Final Model:

```
# def final_model():
#     model = Sequential()
#     model.add(Input(shape=(17, )))
#     model.add(Dense(neurons = 20, activation = 'linear', kernel_constraint = maxnorm(2)))
#     model.add(Dropout(0.2))
#     model.add(Dense(1, activation = 'sigmoid'))
#     optimizer = SGD(lr = 0.01, momentum = 0.2)
#     model.compile(loss = 'binary_crossentropy', optimizer = optimizer, metrics = [matthews_correlation])
#     return model
# annc_model = KerasClassifier(build_fn = final_model, batch_size = 50, epochs = 10, verbose = 0)
# annc_model.fit(X_train.values.astype('float64'), c_train.values.ravel(), class_weight = {0: 15.34, 1: 84.77})
```

C4. Comparison of Classifiers

Model	**Hyperparameters**	**5-fold CV MCC**	**Test Accuracy**	**Test Sensitivity**	**Test Specificity**
SVM (Linear)	C=0.01	**0.25**	**0.67**	**0.71**	**0.66**
SVM (Radial)	C=0.1, gamma=0.01	0.25	0.68	0.64	0.68
SVM (Poly)	C=0.01, degree=1, gamma=1.0	0.25	0.67	0.71	0.66
Decision Tree	criterion=gini, max_depth=2, min_samples_leaf=1	0.20	0.72	0.46	0.75
CCP Decision Tree	ccp_alpha=0.0059	0.21	0.73	0.43	0.78
AdaBoost D-Tree	learning_rate=1.0, n_estimators=100	0.15	0.61	0.62	0.60
k-NN	k=5	0.17	0.53	0.74	0.50

ANN	batch_size=50 , epochs=10 **Hidden Layer 1:** neurons=20, activation=linear, dropout=0.2 , weight_constraint=maxnorm(2), **Output Layer:** neurons=1, activation=sigmoid **SGD Optimizer:** learn_rate=0.01, momentum=0.2	0.25	0.60	0.75	0.57

D. Conclusion:

The two datasets have different properties. The Bike dataset is balanced and linearly separable. Therefore, all the classifiers produced great results. ANN produced the best results with highest accuracy, sensitivity, and specificity combination. The heart disease dataset is an unbalanced dataset and only few predictors are strong enough to explain the class labels. For such a scenario, model evaluation using accuracy would be misleading. Sensitivity and Specificity are better metrics. We have used Matthew's Correlation Coefficient as a scoring method which gives equal weightage to sensitivity and specificity. Support Vector Machine models from the previous assignments have produced the best results. To produce better and robust results, we need to have more samples to produce unbiased predictions.