Assignment 2
Ritik Garg | 2018305

Q1.

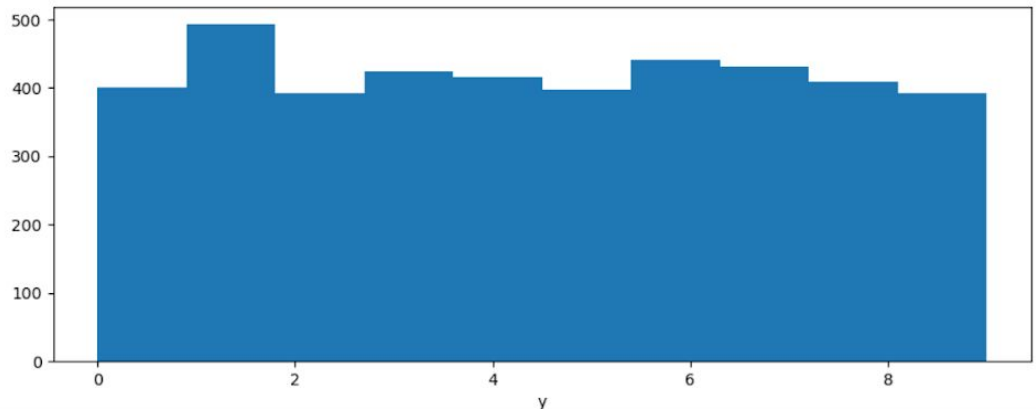
(a).**Principal Component Analysis (PCA)** is a dimensionality reduction technique that enables to identify correlation and patterns in the dataset so that it can be transformed into a dataset of significantly lower dimension without loss of any important information. It works on a probabilistic model. It takes the top eigenvectors with highest values and the ones with the lower values are dropped. It is a linear reduction technique. It maximises variance and preserves large pairwise distances.
It standardizes the dataset. Then the covariance is computed. Then we calculate the eigenvectors and rank them in descending order. The highest rank eigenvectors are chosen.

(b).**Singular Value Decomposition (SVD**) has a popular application of dimensionality reduction. It can be used to reduce the dimension of the data. Here we select the top k largest singular value Sigma to reduce the dimensions of the data.

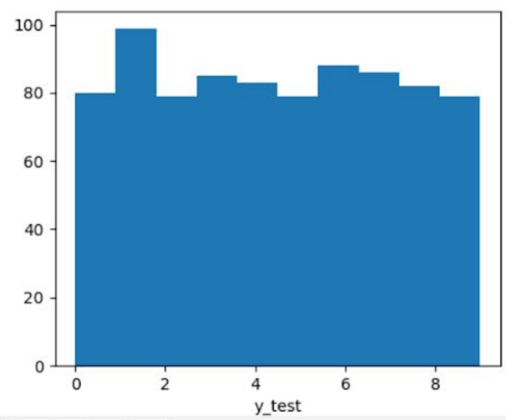
(c).**t-Distributed Stochastic Neighbor Embedding (t-SNE)** is used for data exploration and visualisation of high-dimensional data. It preserves small pairwise distances. It calculates the similarity between pairs of instances in high dimensional space and low dimensional space and tries to optimize them. It is a non-parametric and non-linear method.

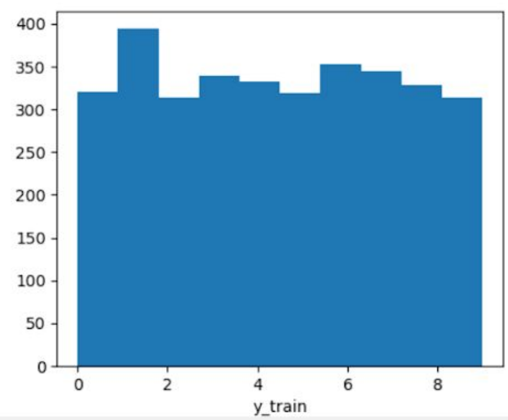
(d). **Stratified sampling** is a type of random sampling in which the samples are partitioned into subparts(strata). Here we do the proportionate allocation of samples. Data will be divided from all the classes in proportionate ratio.
Performing the stratified sampling on the Dataset A and breaking it into 80-20 splits. We can see the graphs of the y_test and y_train and the actual y. They all follow the same class distribution and reduce the chances of an unfortunate split.

Y:



Y_test and Y_train:

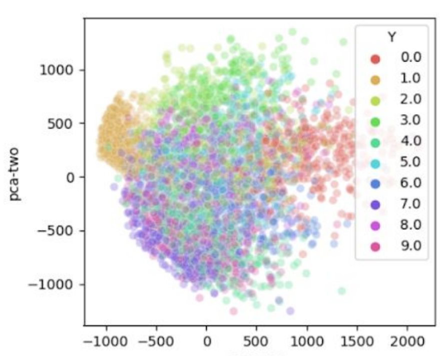


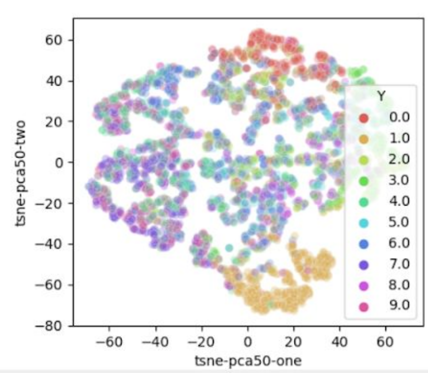


They are following the same class frequency distribution as the original class.

(e).Accuracy using PCA on the dataset: **0.8047619047619048**

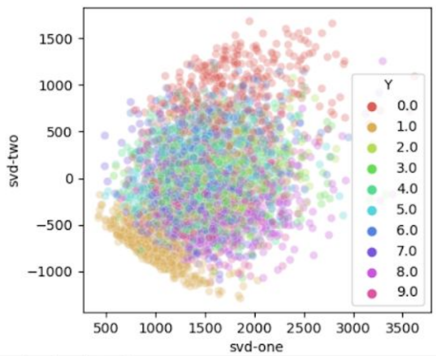
Further analysing using t-SNE:

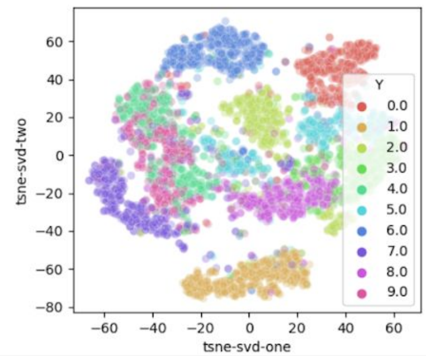




Here we can see how t-SNE has helped to further reduce and segregate the features.

(f).Accuracy using SVD on the dataset: **0.8142857142857143**
Further analysing t-SNE:





Here we can see how t-SNE has helped in further reducing and segregating the features.

(g). Accuracy using SVD is comparable to the accuracy obtained using PCA. PCA is a probabilistic model that reduces the dimensions of the data. SVD allows us to extract and untangle information. PCA reduction can be done in many ways and SVD is one of them.

Q2.

(a).Performed bootstrapping on the dataset C. (Code in the zip file).

```
C:\Users\Ritik garg\Desktop\MlAssignment\Assignment2>python Q2.py
[127.13699821 122.02676156 113.11464298 ... 143.60914042 183.11002246
 128.90027741]  Mean vector
41590310.22477637  Mean
[ 8.62398528 -0.53507317  6.33233295 ... 15.13382163 19.25756111
 15.25117473]  Bias Vector
317894.5455948077  Bias
[0.11048691 0.12767832 0.16211844 ... 0.06775697 0.04417445 0.10498747]  varience vector
18.331719207064864  Varience
Value of MSE - Bias**2 - Varience:  7.332687682946046e-05
```

(b). As the noise is zero, the value of MSE - Bias**2 - Variance = 0 (ideally) but on calculating, its value came out to be 7.33e-05. This value is tending to zero. This is the value of noise that cannot be avoided.

Q3.

(a). Implemented the k fold cv from scratch on Decision trees and Gaussian Naive bayes. For the gridSearchCv, i simply tried to find the best accuracy for all the depths from 1 to 50 and saved the model that has the maximum accuracy on the validation dataset using pickle. After that I used that saved model to predict the accuracy on the test dataset.

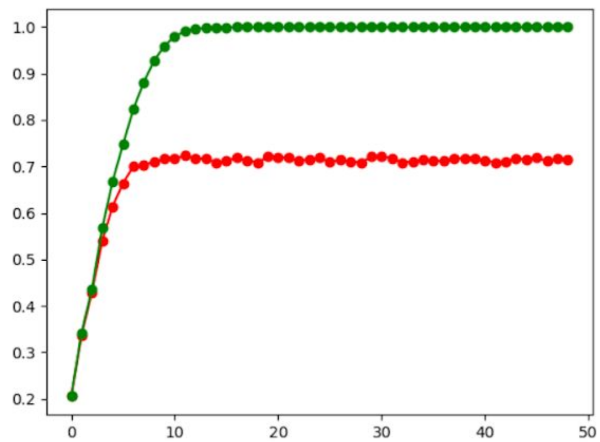
K fold from scratch for Decision tree and Grid fold CV:

```python
# depth = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25]
accuracies_val= []
accuracies_train = []
max_accuracy=-999
max_depth = 1
k = 4                                                               #Creating 4 folds in CV
X_folds = np.array_split(X_train,k)
y_folds = np.array_split(y_train,k)
for d in range(1,20):
  accu_fold_val=0
  accu_fold_train = 0
  for i in range(k):
    X_data_train = X_folds.copy()                                   # Creating the test and validation data
    y_data_train = y_folds.copy()
    X_data_test = X_folds[i]
    y_data_test = y_folds[i]
    del X_data_train[i]
    del y_data_train[i]
    X_data_train = np.concatenate(X_data_train)
    y_data_train = np.concatenate(y_data_train)  #done with data generation train and validate
    # print(len(X_data_test),len(X_data_train))
    '''apply Decision tree on train data and get the accuracy using a depth'''
    dt = tree.DecisionTreeClassifier(criterion='gini',max_depth=d)
    dt.fit(X_data_train,y_data_train)
    y_pred_val = dt.predict(X_data_test)
    accu_val = accuracy(y_data_test,y_pred_val)             #prediciting accuracy on the
    y_pred_train = dt.predict(X_data_train)
    accu_train = accuracy(y_data_train,y_pred_train)
    # accu = accuracy(y_data_test,y_pred)
    # print(accu)
    if(accu_val>max_accuracy):
      max_accuracy = accu_val
      max_depth = d
      saved_model = pickle.dumps(dt)
    accu_fold_val+=(accu_val)
    accu_fold_train+=(accu_train)
  accuracies_val.append(accu_fold_val/k)
  accuracies_train.append(accu_fold_train/k)
plt.plot(accuracies_val,'o-',color="r",
            label="Validation score")
plt.plot(accuracies_train,'o-',color="g",
            label="Train score")
plt.show()
print(accuracies_val,"Accuracy on validation data using Decision Tree")
print("Maximum accuracy and max depth: ",max_accuracy, max_depth)
best_model = pickle.loads(saved_model) #loading the best model to predict on test data
print(best_model.get_depth(),": Model depth equal to the max depth")
y_test_pred = best_model.predict(X_test)
accur_test = accuracy(y_test,y_test_pred)
print("Test accuracy uding the best model: ",accur_test)
```

K fold on Gaussian NB from scratch:

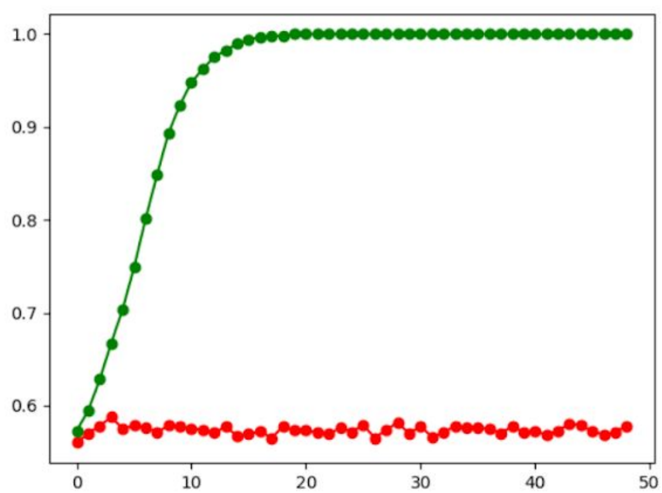
```python
accuracies_val= []
accuracies_train = []
max_accuracy=-999
# max_depth = 1
k = 4  #Creating 4 folds in CV
X_folds = np.array_split(X_train,k)
y_folds = np.array_split(y_train,k)
accu_fold_val=0
accu_fold_train = 0
for i in range(k):
  X_data_train = X_folds.copy()
  y_data_train = y_folds.copy()
  X_data_test = X_folds[i]
  y_data_test = y_folds[i]
  del X_data_train[i]
  del y_data_train[i]
  X_data_train = np.concatenate(X_data_train)
  y_data_train = np.concatenate(y_data_train)  #done with data generation train and validate
  # print(len(X_data_test),len(X_data_train))
  #apply Decision tree on train data and get the accuracy using a depth
  clf = GaussianNB()
  model = clf.fit(X_data_train,y_data_train)
  # y_pred = clf.predict(X_test)
  y_pred_val = clf.predict(X_data_test)
  accu_val = accuracy(y_data_test,y_pred_val)
  y_pred_train = clf.predict(X_data_train)
  accu_train = accuracy(y_data_train,y_pred_train)
  # accu = accuracy(y_data_test,y_pred)
  # print(accu)
  if(accu_val>max_accuracy):
    max_accuracy = accu_val
    # max_depth = d
    saved_model = pickle.dumps(clf)
  accu_fold_val+=(accu_val)
  accu_fold_train+=(accu_train)
accuracies_val.append(accu_fold_val/k)
accuracies_train.append(accu_fold_train/k)
# plt.plot(accuracies_val,'o-',color="r",
            # label="Validation score")
# plt.plot(accuracies_train,'o-',color="g",
            # label="Train score")
print("Accuracy on validate: ",accuracies_val)
print("Maximum Accuracy: ",max_accuracy)
best_model = pickle.loads(saved_model) #loading the best model to predict on test data
# print(best_model.get_depth(),"Model depth equal to the max depth")
y_test_pred = best_model.predict(X_test)
accur_test = accuracy(y_test,y_test_pred)
print("Test accuracy using GaussianNB: ",accur_test)
```

(b). Depth vs accuracy on Dataset A: green: X_train ; red: X_test
[x axis: depth, y axis: accuracy]



Here dataset is trending to overfit in the testing data

Depth vs accuracy on Dataset A: green: X_train ; red: X_test
[x axis: depth, y axis: accuracy]



Here dataset is trending to overfit in the testing data (c).Dataset A:

Decision Trees:

```
C:\Users\Ritik garg\Desktop\MlAssignment\Assignment2>python Q3.py
Enter 1 for Dataset A and 2 for Dataset B: 1
Enter 1 for Decision Tree and 2 for Gaussian NB: 1
Decision Tree
Applying 4 Fold CV to find the optimal depth of the decision tree with
[0.20565476190476195, 0.337202380952381, 0.42946428571428574, 0.5404761
714285714, 0.7089285714285715, 0.7101190476190476, 0.725, 0.70952380952
Maximum accuracy and max depth:  0.7440476190476191 14
14 : Model depth equal to the max depth
Test accuracy uding the best model:  0.7440476190476191

-------------------------Printing the confusion_matrix--------------
Accuracy of the data:  0.7440476190476191
Precision on the data:  0.875
Recall on the data:  0.8850574712643678
F1_score on the data:  0.8800000000000001
confusion_matrix:  [[60  0  1  3  4  6  0  0  1  3]
 [ 0 77  1  2  0  1  0  5  0  1]
 [ 0  4 63  6  4  0  5  2  4  3]
 [ 1  0  4 56  2  4  1  1  2  4]
 [ 1  1  1  2 64  1  6  2  2  7]
 [ 3  2  2  5  1 46  2  0  3  5]
 [ 2  1  3  1  6  8 77  0  4  0]
 [ 0  1  4  2  2  1  0 83  0  2]
 [ 1  2  5  7  2  1  3  2 44  9]
 [ 1  0  2  3  4  8  0  3  4 55]]

---------------------ROC Curves---------------------------------------
Predict_proba:  [[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 1. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 1.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 1. 0. ... 0. 0. 0.]]
```

Best model: Max acc: 0.744 Max Depth: 14

Accuracy on train data: 0.744

Gaussian NB:

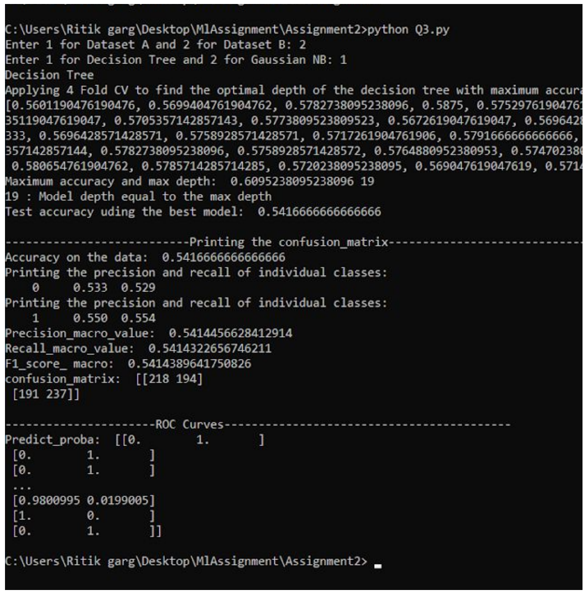
```
C:\Users\Ritik garg\Desktop\MlAssignment\Assignment2>python Q3.py
Enter 1 for Dataset A and 2 for Dataset B: 1
Enter 1 for Decision Tree and 2 for Gaussian NB: 2
Accuracy on validate:  [0.5752976190476191]
Maximum Accuracy:  0.6
Test accuracy using GaussianNB:  0.5666666666666667

-------------------------Printing the confusion_matrix-----------------------------
Accuracy of the data:  0.5666666666666667
Precision on the data:  0.7272727272727273
Recall on the data:  0.9195402298850575
F1_score on the data:  0.8121827411167513
confusion_matrix:  [[60  0  0  1  0  0  6  0  1 10]
 [ 1 80  0  2  0  1  0  0  2  1]
 [10  3 29 15  2  2 21  1  6  2]
 [ 5  2  6 43  0  0  4  1  3 11]
 [ 4  3  0  0 21  1  8  0 10 40]
 [ 8  5  2  8  1 12  4  0 23  6]
 [ 2  2  3  0  1  1 93  0  0  0]
 [ 0  1  0  1  1  0  0 34  0 58]
 [ 5 11  1  3  0  1  3  0 32 20]
 [ 0  3  0  0  1  0  0  4  0 72]]

--------------------ROC Curves---------------------------------------
Predict_proba:  [[0.00000000e+000 0.00000000e+000 0.00000000e+000 ... 0.00000000e+000
  0.00000000e+000 0.00000000e+000]
 [1.03708635e-219 0.00000000e+000 0.00000000e+000 ... 0.00000000e+000
  1.00000000e+000 0.00000000e+000]
 [0.00000000e+000 0.00000000e+000 0.00000000e+000 ... 2.85861052e-060
  0.00000000e+000 1.00000000e+000]
 ...
 [0.00000000e+000 0.00000000e+000 0.00000000e+000 ... 1.20576176e-050
  8.45975080e-210 1.00000000e+000]
 [0.00000000e+000 0.00000000e+000 5.58327383e-305 ... 0.00000000e+000
  0.00000000e+000 0.00000000e+000]
 [0.00000000e+000 1.00000000e+000 0.00000000e+000 ... 0.00000000e+000
  0.00000000e+000 0.00000000e+000]]
```

Best model: Max acc: 0.6

Accuracy on train data: 0.56

Dataset B:

Decision Tree:

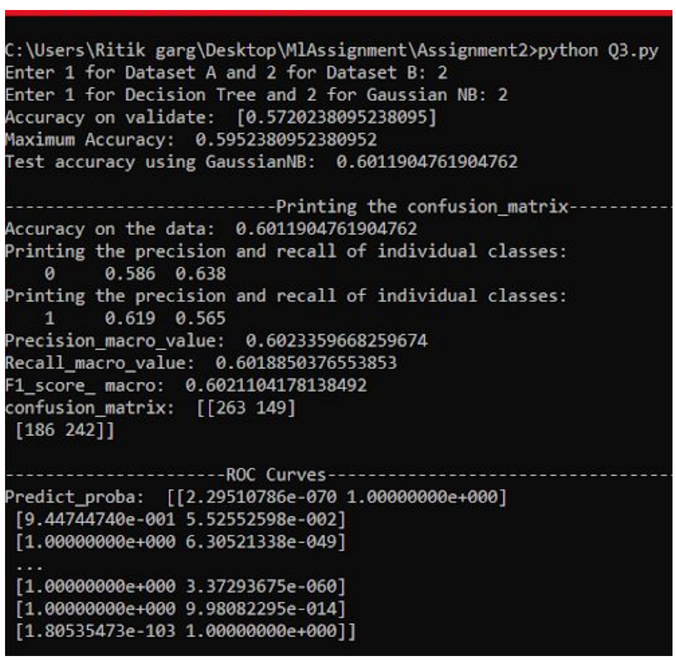


```
C:\Users\Ritik garg\Desktop\MlAssignment\Assignment2>python Q3.py
Enter 1 for Dataset A and 2 for Dataset B: 2
Enter 1 for Decision Tree and 2 for Gaussian NB: 1
Decision Tree
Applying 4 Fold CV to find the optimal depth of the decision tree with maximum accura
[0.5601190476190476, 0.5699404761904762, 0.5782738095238096, 0.5875, 0.57529761904761
35119047619047, 0.5705357142857143, 0.5773809523809523, 0.5672619047619047, 0.5696428
333, 0.5696428571428571, 0.5758928571428571, 0.5717261904761906, 0.5791666666666666,
357142857144, 0.5782738095238096, 0.5758928571428572, 0.5764880952380953, 0.574702380
 0.580654761904762, 0.5785714285714285, 0.5720238095238095, 0.569047619047619, 0.5714
Maximum accuracy and max depth:  0.6095238095238096 19
19 : Model depth equal to the max depth
Test accuracy uding the best model:  0.5416666666666666

--------------------------Printing the confusion_matrix-----------------------------
Accuracy on the data:  0.5416666666666666
Printing the precision and recall of individual classes:
    0     0.533  0.529
Printing the precision and recall of individual classes:
    1     0.550  0.554
Precision_macro_value:  0.5414456628412914
Recall_macro_value:  0.5414322656746211
F1_score_ macro:  0.5414389641750826
confusion_matrix:  [[218 194]
 [191 237]]

---------------------ROC Curves---------------------------------------
Predict_proba:  [[0.        1.        ]
 [0.        1.        ]
 [0.        1.        ]
 ...
 [0.9800995 0.0199005]
 [1.        0.        ]
 [0.        1.        ]]

C:\Users\Ritik garg\Desktop\MlAssignment\Assignment2>
```

Best model: Max acc: 0.60 Max Depth: 4

Accuracy on train data: 0.58

Gaussian NB:



```
C:\Users\Ritik garg\Desktop\MlAssignment\Assignment2>python Q3.py
Enter 1 for Dataset A and 2 for Dataset B: 2
Enter 1 for Decision Tree and 2 for Gaussian NB: 2
Accuracy on validate:  [0.5720238095238095]
Maximum Accuracy:  0.5952380952380952
Test accuracy using GaussianNB:  0.6011904761904762

--------------------------Printing the confusion_matrix----------
Accuracy on the data:  0.6011904761904762
Printing the precision and recall of individual classes:
    0     0.586  0.638
Printing the precision and recall of individual classes:
    1     0.619  0.565
Precision_macro_value:  0.6023359668259674
Recall_macro_value:  0.6018850376553853
F1_score_ macro:  0.6021104178138492
confusion_matrix:  [[263 149]
 [186 242]]

---------------------ROC Curves----------------------------------
Predict_proba:  [[2.29510786e-070 1.00000000e+000]
 [9.44744740e-001 5.52552598e-002]
 [1.00000000e+000 6.30521338e-049]
 ...
 [1.00000000e+000 3.37293675e-060]
 [1.00000000e+000 9.98082295e-014]
 [1.80535473e-103 1.00000000e+000]]
```

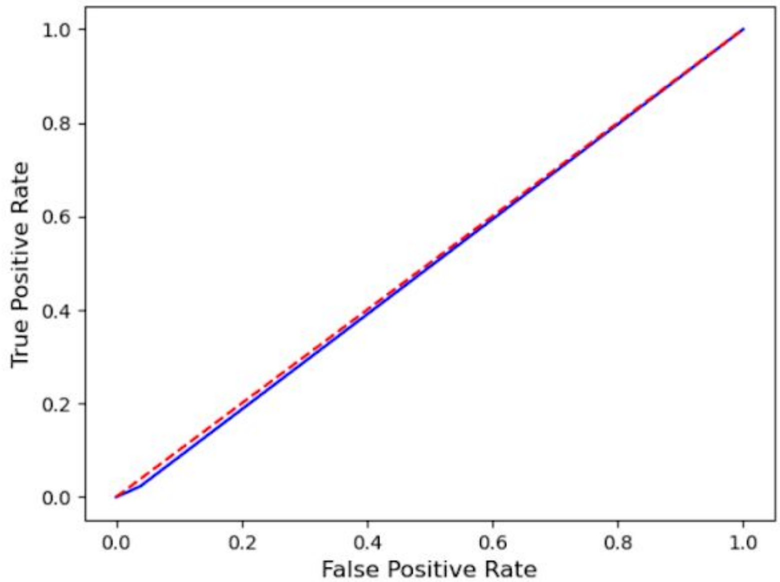
Best model: Max acc: 0.60

Accuracy on train data: 0.6011

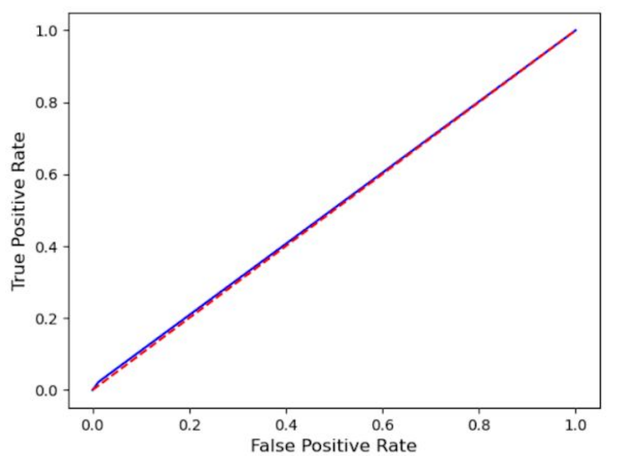
(d). **Precision, Accuracy, Recall values, F1_score values and the confusion matrix for both the datasets have been included in the above pictures for both the models and both the datasets.**

Dataset A: RoC Curves

Decision Trees:

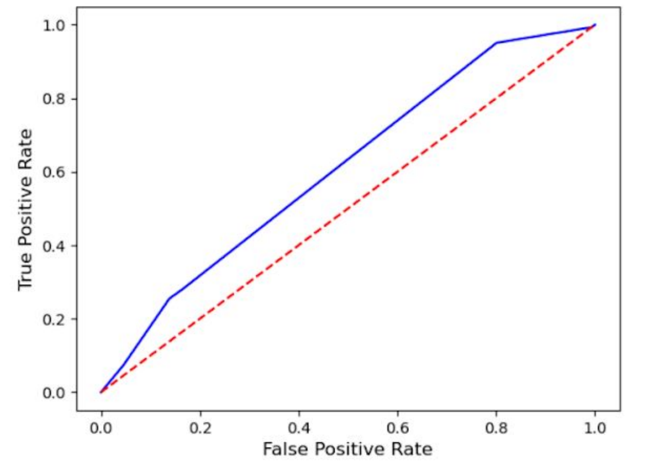


Gaussian NB:

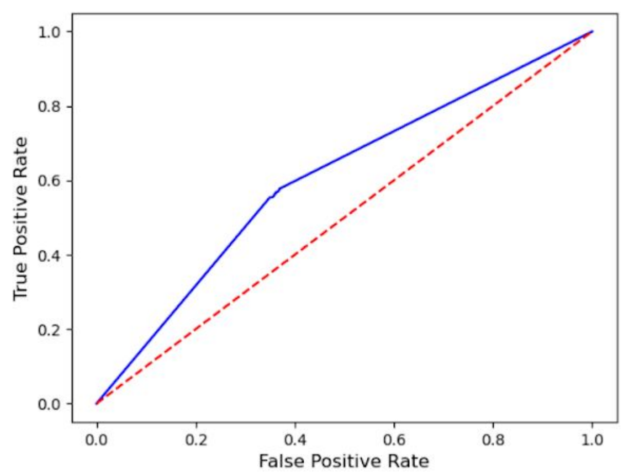


Dataset B: **ROC curves**

Decision Tree:



Gaussian NB:



Q4. Accuracy on Dataset 1:

Using Sklearn: 0.5952380952380952

Using mine implementation: 0.5416666666666666

Accuracy on Dataset 2:

Using Sklearn: 0.5721428571428572

Using mine implementation: 0.5833333333333334

To avoid the division by zero error, i had given variance a very small value (10(-6)) to remove the division by zero error and there is no significant change in the output.

Answer 5 to Answer 8 images are uploaded in the doc below.

Answer 5. Predicting IPL matches from outlook, climate, Humidity, Wind.

Entropy on training : $- P(Yes) * \log(P(Yes)) - P(NO) \log(P(NO))$

$$E(S) = -\left(\frac{5}{14}\log_2\frac{5}{14} + \frac{9}{14}\log_2\frac{9}{14}\right)$$

$$= \quad +$$

$$= 0.9403$$

We will choose the variable to split, which will have max. information gain.

$$\text{InfoGain} = E(S) - E(S/X)$$

- $\text{Info gain}(S, T) = (0.9403) - \left[\frac{6}{14}\left(-\left(\frac{2}{6}\log\frac{2}{6} + \frac{4}{6}\log\frac{4}{6}\right)\right) + \frac{4}{14}\left(-\left(\frac{1}{4}\log\frac{1}{4} + \frac{3}{4}\log\frac{3}{4}\right)\right)\right]$

$$= (0.9403) - [\quad - \quad - \quad]$$

$$= 0.9403 -$$

$$= \underline{0.0292}$$

- $\text{InfoGain}(S, H) = (0.9403) - \left[\frac{7}{14}\left(-\left(\frac{4}{7}\log\frac{4}{7} + \frac{3}{7}\log\frac{3}{7}\right)\right) + \frac{7}{14}\left(-\left(\frac{6}{7}\log\frac{6}{7} + \frac{1}{7}\log\frac{1}{7}\right)\right)\right]$

$$= (0.9403) - [\; - \quad - \;]$$

$$= 0.9403 +$$

$$= \underline{0.1518}$$

- $\text{Info gain}(S, W) = (0.9403) - \left[\frac{8}{14}\left(-\left(\frac{2}{8}\log\frac{2}{8} + \frac{6}{8}\log\frac{6}{8}\right)\right) + \frac{6}{14}\left(-\left(\frac{3}{6}\log\frac{3}{6} + \frac{3}{6}\log\frac{3}{6}\right)\right)\right]$

$$= 0.9403 - \left[\frac{8}{4}\left(- \quad\right) + \frac{6}{14}\left(\right.\right.$$

$$= 0.9403 -$$

$$= \underline{0.0481}$$

...... from each feature

$$\text{Infogain}(S, O) = 0.9403 - \left[\frac{5}{14}\left(-\frac{3}{5}\log\frac{3}{5} + \frac{2}{5}\log\frac{2}{5}\right) + \frac{5}{14}\left(-\frac{3}{5}\log\frac{3}{5} + \frac{2}{5}\log\frac{2}{5}\right) + \frac{4}{14}\left(-(0\log 0) - \frac{3}{3}\log\frac{3}{3}\right)\right]$$

$$= 0.9403 - [$$

$$=$$

$$= \underline{0.2468}$$

∴ The first attribute should be the <u>Outlook</u> as it has <u>max. InfoGain</u>.

Now, we need to choose an attribute to split

Outlook

Sunny / Overcast / Rain

Sunny: $E(P) = -\frac{2}{5}\log\frac{2}{5} - \frac{3}{5}\log\frac{3}{5} = E(2,3) = 0.971$

Overcast: $E(4,0) = 0$

Rain: $E(R) = -\frac{3}{5}\log\frac{3}{5} - \frac{2}{5}\log\frac{2}{5}$, $E(3,2) = 0.971$

finding Infogain

$\text{Info}(S, T) = 0.971 - \cancel{\frac{2}{5}\log\frac{2}{5}} - \frac{2}{5}\left(\frac{0}{2}\log\frac{0}{2} - \frac{2}{2}\log\frac{2}{2}\right) - \frac{1}{5}(0) - \frac{2}{5}\left(-\frac{1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2}\right)$

$\boxed{\text{Info}(S,T) = 0.571}$

$\text{Info}(S, H) = 0.971 - \frac{3}{5}\left(-\frac{3}{3}\log\frac{3}{3}\right) - \frac{2}{5}\left(-\frac{2}{2}\log\frac{2}{2}\right)$

$= 0.971 - \frac{3}{5}(0) - \frac{2}{5}(0)$

$\boxed{\text{Info}(S,H) = 0.971}$

$\text{Info}(S, W) = 0.971 - \frac{3}{5}\left(-\frac{1}{3}\log\frac{1}{3} - \frac{2}{3}\log\frac{2}{3}\right) - \frac{2}{5}\left(-\frac{1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2}\right)$

$\boxed{\text{Info}(S,W) = 0.02}$

∴ next att. will be <u>Humidity</u>

finding Infogain

$\text{Info}(R, C) = 0.971 - \frac{2}{5}(1) - \frac{3}{5}\left(\frac{1}{3}\log\frac{1}{3} - \frac{2}{3}\log\frac{2}{3}\right)$

$\boxed{\text{Info}(R,C) = 0.02}$

$\text{Info}(R, H) = 0.971 - \frac{2}{5}(1) - \frac{3}{5}\left(\frac{1}{3}\log\frac{1}{3} - \frac{2}{3}\log\frac{2}{3}\right)$

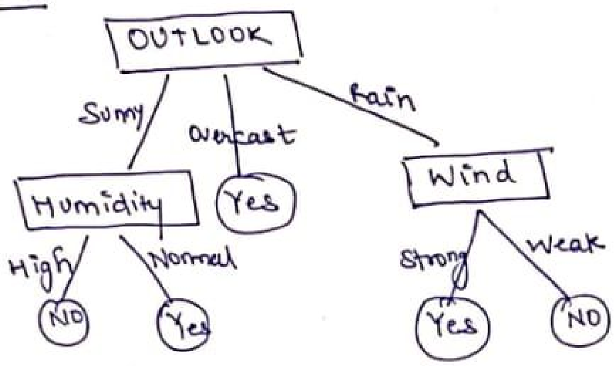
$\boxed{\text{Info}(R,H) = 0.02}$

$\text{Info}(R, W) = 0.971 - \frac{3}{5}(0) - \frac{2}{5}(0)$

$\boxed{\text{Info}(R,W) = 0.971}$

∴ next att. will be <u>Wind</u>

Tree will be

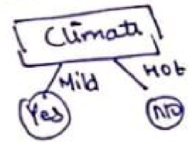


(b) Yes, we can find some set of traning examples that will get the algo. to include the attribute climate, even through the true target concept is independent of climate.

Example: subset: $\{D_1, D_2, D_4, D_{10}, D_{11}, D_{12}\}$.

$$\boxed{E(S,C) = -\frac{2}{6}(0) - \frac{4}{6}(0) = 0}$$

For this subset, climate attribute has entropy 0, it perfectly predicts play match. The tree would include climate only, as a single node.

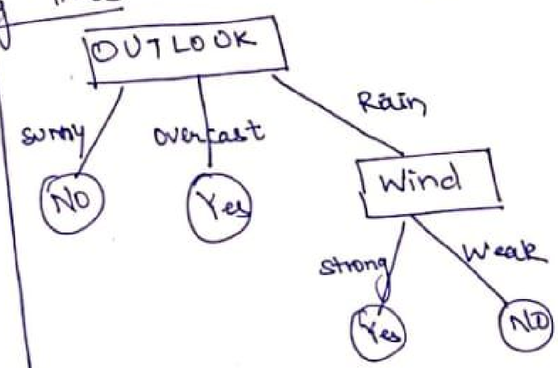


(c) Corresponding Tree:

on D8 - D14

	Pred	Actual	
D8	No	No	✓
D9	No	Yes	✗
D10	Yes	Yes	✓
D11	NO	Yes	✗
D12	Yes	Yes	✓
D13	Yes	Yes	✓
D14	NO	NO	✓

$\boxed{\text{Accuracy} = \frac{5}{7}}$



Here the test accuracy is $\frac{5}{7}$. The tree is too generic & training set didnot include sunny attribute. Hence, the model failed to classify that instance in test.

(d) We can add ~~contraints~~ on the minimal no. of training examples in a ~~to~~ leaf node as a pruning strategy.
Also, the tree can include upto 'k' pre-defined no. of levels as a simple tree is more generic.

Answer 6: We need to find $p(w_3|w_1, w_2, w_4)$
By markov's rule, we can ignore w_1
$\therefore$ we need to find $p(w_3|w_2, w_4)$
$\therefore$ we need to find $p(w_3 | w_2 = \text{course})$

Using Baye's rule: prior: $p(w_3 | w_2 = \text{course})$
likelihood: $p(w_4 = \text{course} | w_3)$

$$p(w_3|w_2, w_4) = \frac{1}{Z} p(w_3|w_2) p(w_4|w_2, w_3)$$

$$p(w_3|w_2,w_4) = \frac{1}{Z} p(w_3|w_2) p(w_4|w_3) \quad [\text{ignoring } w_2]$$

$$p(w_3|w_2,w_4) = \frac{1}{Z} p(w_3|w_2) p(\text{tough}|\text{course}) . p(\text{course}|\text{tough})$$

now, $p(?? = \text{tough}) = \frac{1}{Z} p(\text{tough}|\text{course})$
$= \frac{1}{Z}(0.5)(0.3) = \frac{(0.15)}{Z}$ — (i)

$p(?? = \text{course}) = \frac{1}{Z} p(\text{course}|\text{course}) p(\text{course}|\text{cou}$
$= \frac{1}{Z}(0.5)(0.5) = \frac{0.25}{Z}$ (ii)

we know, $p(?? = \text{tough}) + p(?? = \text{course}) = 1$

$$\therefore 1 = \frac{(0.15)}{Z} + \frac{(0.25)}{Z}$$

$$\Rightarrow Z = 0.4$$

$$\therefore p(?? = \text{tough}) = \frac{0.15}{4}$$

$$p(?? = \text{course}) = \frac{0.25}{4}$$

Answer (7) (a) Logistic regression treats each feature independently while the Decision trees donot assume independence of input features. They can thus encode complicated formulas related the variables.

(b) We associate only one feature/parameter with each feature in logistic regression ~~which~~ while decision trees can split on many different combination of features. Hence, decision trees tends to overfit the data.

(c)
$$y = \begin{cases} +1 & w^T x + b > 0 \\ -1 & w^T x + b \leq 0 \end{cases}$$

Yes, decision tree can classify correctly. We can split the points according to their X_1 values as they are linearly separable. Also, for each value of x_1, there will be cuttoff on X_2 and the values above it are in class 1 & below it in class -1.

Upper bound on depth = $O(\log n)$

As splitting on values of x_1, can be done with a $\log(n)$ depth and we need at most ~~more~~ more node for classification

$\therefore$ depth $= O(\log n) + 1$
$= O(\log n)$

(d) Yes, ~~we can~~ decision tree can classify correctly. We can split the points on the values of x_1. Now there cannot be any cuttoff for x_2 as the data is not linearly separable.

Upper bound on depth = $O(\log n)$

After x_1, we need to classify x_2, that can be done in $\log n$ depth.

$\therefore$ total depth $= 2\log n$
$= O(\log n)$

Answer (7) (a) Logistic regression treats each feature independently while the Decision trees donot assume independence of input features. They can thus encode complicated formulas related the variables.

(b) We associate only one feature/parameter with each feature in logistic regression ~~which~~ while decision trees can split on many different combination of features. Hence, decision trees tends to overfit the data.

(c)

$$y = \begin{cases} +1 & w^T x + b > 0 \\ -1 & w^T x + b \leq 0 \end{cases}$$

Yes, decision tree can classify correctly. We can split the points according to their x_1 values as they are linearly separable. Also, for each value of x_1, there will be cuttoff on x_2 and the values above it are in class 1 & below it in class -1.

Upper bound on depth $= O(\log n)$

As splitting on values of x_1, can be done with a $\log(n)$ depth and we need at most ~~more~~ more node for classification

$\therefore$ depth $= O(\log n) + 1$

$= O(\log n)$

(d) Yes, ~~we~~ ~~can~~ decision tree can classify correctly. We can split the points on the values of x_1. Now there cannot be any cuttoff for x_2 as the data is not linearly separable.

Upper bound on depth $= O(\log n)$

After x_1, we need to classify x_2, that can be done in $\log n$ depth.

$\therefore$ total depth $= 2\log n$

$= O(\log n)$

Answer (7) (a) Logistic regression treats each feature independently while the Decision trees donot assume independence of input features. They can thus encode complicated formulas related the variables.

(b) We associate only one feature/parameter with each feature in logistic regression while decision trees can split on many different combination of features. Hence, decision trees tends to overfit the data.

(c)
$$y = \begin{cases} +1 & w^T x + b > 0 \\ -1 & w^T x + b \leq 0 \end{cases}$$

Yes, decision tree can classify correctly. We can split the points according to their x_1 values as they are linearly separable. Also, for each value of x_1, there will be cuttoff on x_2 and the values above it are in class 1 & below it in class -1.

Upper bound on depth = $O(\log n)$

As splitting on values of x_1, can be done with a $\log(n)$ depth and we need at most more node for classification

$\therefore$ depth $= O(\log n) + 1$

$= O(\log n)$

(d) Yes, decision tree can classify correctly. We can split the points on the values of x_1. Now there cannot be any cuttoff for x_2 as the data is not linearly separable.

Upper bound on depth = $O(\log n)$

After x_1, we need to classify x_2, that can be done in $\log n$ depth.

$\therefore$ total depth $= 2\log n$

$= O(\log n)$

Ans (B) : ~~Given~~: ~~priors~~

$X = \langle X_1, \cdots X_n \rangle$

Assumptions:

→ Given Y is a boolean, it can have bernoulli distribution

$$P(Y=1) = \pi \quad \text{(given)}$$

→ $P(X_i|Y=y_k)$ is a Gaussian Distribution $\forall X_i$

→ X_i & X_j are conditionally independent $\forall i,j$

By Baye's rule

$$P(Y=1|X) = \frac{P(Y=1).P(X|Y=1)}{P(Y=1)P(X|Y=1)+P(Y=0)P(X|Y=0)}$$

Dividing N^r & D^r by N^r

$$P(Y=1|X) = \frac{1}{1+\frac{P(Y=0)P(X|Y=0)}{P(Y=1)P(X|Y=1)}}$$

[using $a = e^{\ln a}$ in D^r]

$$P(Y=1|X) = \frac{1}{1+e\left(\ln\left(\frac{P(Y=0)P(X|Y=0)}{P(Y=1)P(X|Y=1)}\right)\right)}$$

[Now $\ln(AB) = \ln A + \ln B$]

$$P(Y=1|X) = \frac{1}{1+e\left(\ln\frac{P(Y=0)}{P(Y=1)} + \ln\frac{P(X|Y=0)}{P(X|Y=1)}\right)}$$

$\boxed{P(Y=1) = \pi}$ $(P(Y=0) = 1-\pi)$

$$P(Y=1|X) = \frac{1}{1+e\left(\ln\left(\frac{1-\pi}{\pi}\right) + \sum\ln\left(\frac{P(X_i|Y=0)}{P(X_i|Y=1)}\right)\right)}$$

→ Applying Gaussian Distr.

$$\sum \ln\left(\frac{P(X_i|Y=0)}{P(X_i|Y=1)}\right) = \sum_i \ln \frac{\cancel{\frac{1}{\sqrt{2\pi\sigma^2}}}\exp\left(\frac{-(X_i-\mu_{i0})^2}{2\sigma_i^2}\right)}{\cancel{\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)}\exp\left(\frac{-(X_i-\mu_{i1})^2}{2\sigma_i^2}\right)}$$

$$= \sum_i \ln\left(\exp\left(\frac{(X_i-\mu_{i1})^2 - (X_i-\mu_{i0})^2}{2\sigma_i^2}\right)\right)$$

$$= \sum_i \left(\frac{(X_i-\mu_{i1})^2 - (X_i-\mu_{i0})^2}{2\sigma_i^2}\right)$$

$$= \sum_i \left(\frac{(x_i^2 - 2x_i\mu_{i1} - \mu_{i1}^2) - (x_i^2 - 2x_i\mu_{i0} - \mu_{i0}^2)}{2\sigma_i^2} \right)$$

$$= \sum_i \left(\frac{2x_i(\mu_{0i} - \mu_{i1}) + \mu_{i1}^2 - \mu_{i0}^2}{2\sigma_i^2} \right)$$

$$= \sum_i \left(\frac{\mu_{i0} - \mu_{i1}}{\sigma_i^2} x_i + \frac{\mu_{i1}^2 - \mu_{i0}^2}{2\sigma_i^2} \right)$$

$$\therefore\ P(Y=1|X) = \frac{1}{1 + \exp\left(\ln\frac{(1-\pi)}{\pi} + \sum_i \left(\frac{\mu_{i0} - \mu_{i1}}{\sigma_i^2} X_1 + \frac{\mu_{i1}^2 - \mu_{i0}^2}{2\sigma_i^2} \right) \right)}$$

$$= \frac{1}{1 + \exp\left(\left(\ln\left(\frac{1-\pi}{\pi}\right) + \frac{\mu_{i1}^2 - \mu_{i0}^2}{2\sigma_i^2} \right) + \sum_i \left(\frac{\mu_{i0} - \mu_{i1}}{\sigma_i^2} \right) X_i \right)}$$

$$P(Y=1|X) = \frac{1}{1 + \exp(w_0 + \sum_i w_1 X_i)}$$

where, $w_0 = \left(\ln\left(\frac{1-\pi}{\pi}\right) + \frac{\mu_{i1}^2 - \mu_{i0}^2}{2\sigma_i^2} \right)$ &

$$w_1 = \frac{\mu_{i0} - \mu_{i1}}{\sigma_i^2}$$

Hence, it can be expressed in the form of logistic regression.