MTL782 Data Mining Assignment Subsmission

Nitin Gautam[*]
Submitted to Prof. Bhawani Shankar Panda [†]

March 27th, 2022

- The project is solely done by me, as unable to find group member who were interested in dataset (Pregnant women Risk stratification in developing countries), which is area of interest in development economics. Similar studies are carried out using NFHS survey dataset in India (National Family Health survey - 5)

- Jupyter notebook python Code is appended in appendix in this report for both Question 1 and 2
- Github Repository Link for code submission: magenta https://github.com/Kafka-21/MTL782_Assignment

- Link for dataset magentaPregnant Women dataset, Python code download data from this url

[*]hes207035@hss.iitd.ac.in
[†]as part of the course requirement in MTL 782 Data Mining

Question 1

Choose a data set from UCI Machine Learning Repository for Multi class classification problems.

(i) Your first task is characterize the data set. Answer the following questions about the data:

Table 1: Summary Statistics of attributes

	Age	SystolicBP	DiastolicBP	BS	BodyTemp	HeartRate
count	1,014.00	1,014.00	1,014.00	1,014.00	1,014.00	1,014.00
mean	29.87	113.20	76.46	8.73	98.67	74.30
std	13.47	18.40	13.89	3.29	1.37	8.09
min	10.00	70.00	49.00	6.00	98.00	7.00
25%	19.00	100.00	65.00	6.90	98.00	70.00
50%	26.00	120.00	80.00	7.50	98.00	76.00
75%	39.00	120.00	90.00	8.00	98.00	80.00
max	70.00	160.00	100.00	19.00	103.00	90.00

Data set contains
dataset includes

- Medical data of pregnant women of Bangladesh where attributes were collected using Internet of things devices
  - **Age** : age of pregnant woman
  - **SystolicBP** : Systolic Blood pressure
  - **DiastolicBP** : Diastolic Blood pressure
  - **BS** : Blood Sugar
  - **BodyTemp** : Body Temperature
  - **HeartRate** : Heart Rate
- Outcome variable is about Risk stratification of the pregnant women based on observed attributes measured using IoT
  - Low Risk
  - Mid Risk
  - High Risk

- The classification and prediction of risk intensity during pregnancy is a multilayered problem
- Pregnant women are categorised into high and low risk based on these measured attributes.
- All these attributes are continuous variable and outcome variable is multi class categorical/indicator variable.

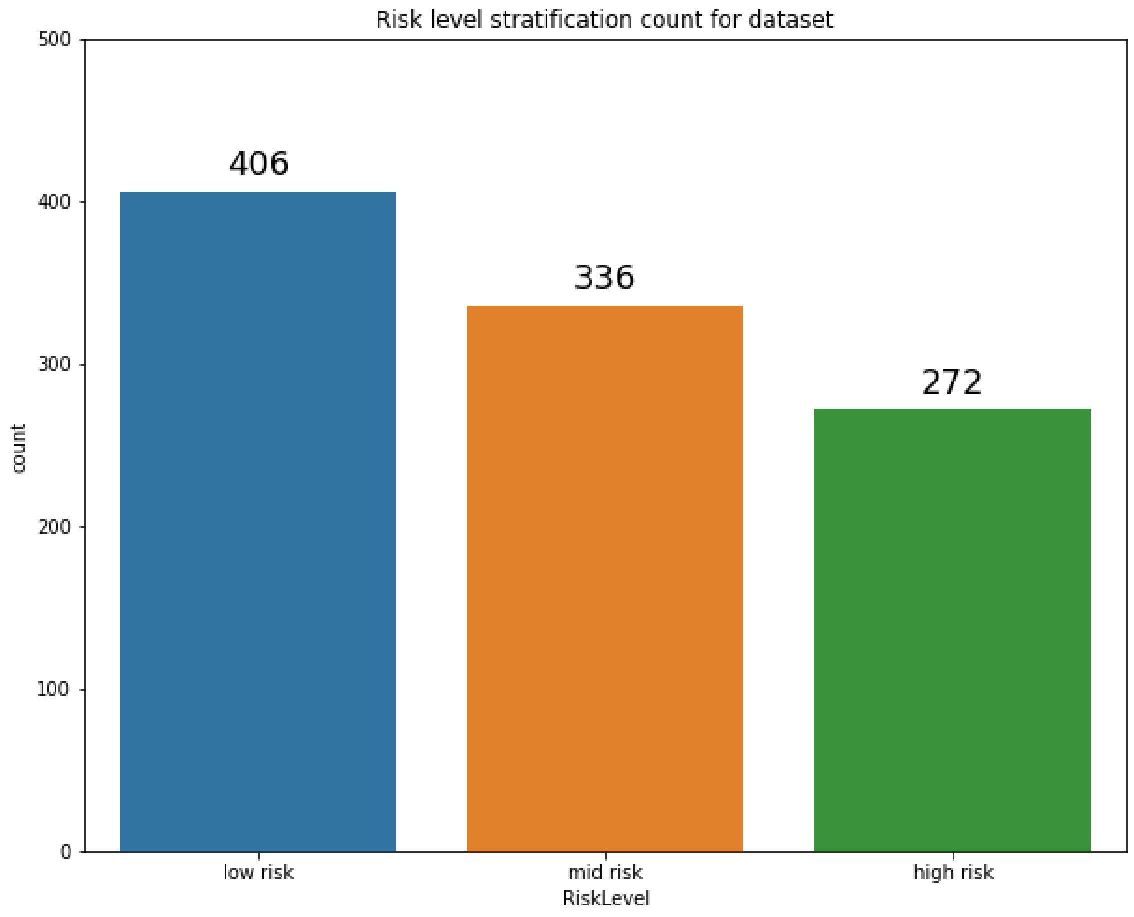


Figure 1: Risk level stratification count

2) What type of benefit you might hope to get from data mining.

- Data Mining study is to solve the maternal healthcare problem, mainly for the rural areas of a developing country.
- Study will help to reduce the problems of pregnant women as well as a newborn baby.
- Thus, risk stratification using data mining will increase the safety of maternal health as well as the unborn baby

3) Discuss data quality issues: For each attribute,

a) Are there problems with the data?

None of attributes is missing in dataset, hence minimum quality issues

b) What might be an appropriate response to the quality issues?

If quality issues existed, these can be resolved using

- If attribute data is missing for a row, dropping/discarding individual pregnant women data
- Filling missing dataset with most common values or mean of columns

(ii) Implement

(1) Decision Tree (2) Random Forest (3) Naïve Bayes Classifier (4) KNN classifier

compare the performances using k-fold cross validation and other tuning techniques.

1 Feature Selection

1.1 Chi-squared stats

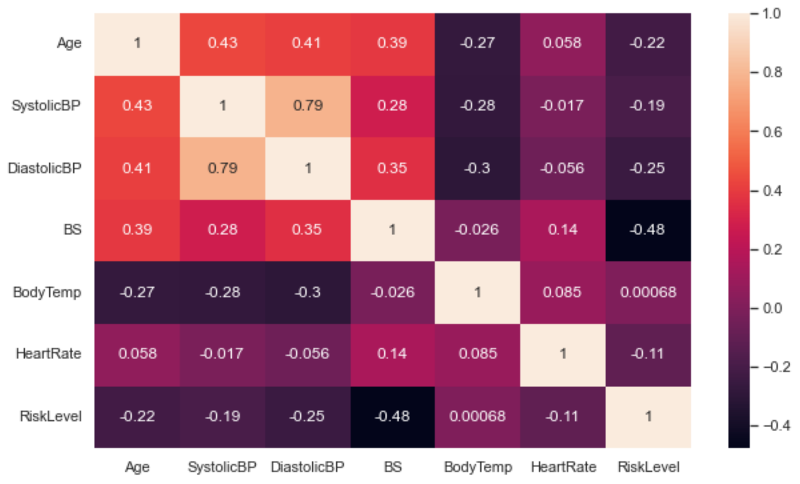
This score can be used to select the features with the highest values for the test chi-squared statistic from X, which must contain only non-negative features such as booleans or frequencies relative to the classes.

Chi-square test values to check whether or not a particular attribute is influential on the response reported in table 2

Table 2: Chi-square test to get influential attributes

	Age	SystolicBP	DiastolicBP	BS	BodyTemp	HeartRate
chi-square	500.92	556.24	323.26	663.13	61.96	212.91
p-value	0.00	0.00	0.00	0.00	0.00	0.00
Critical value	133.48	58.62	50.89	83.51	29.14	50.89
Influential	yes	yes	yes	yes	yes	yes

1.2 Correlation Heat Map



	Age	SystolicBP	DiastolicBP	BS	BodyTemp	HeartRate	RiskLevel
Age	1	0.43	0.41	0.39	-0.27	0.058	-0.22
SystolicBP	0.43	1	0.79	0.28	-0.28	-0.017	-0.19
DiastolicBP	0.41	0.79	1	0.35	-0.3	-0.056	-0.25
BS	0.39	0.28	0.35	1	-0.026	0.14	-0.48
BodyTemp	-0.27	-0.28	-0.3	-0.026	1	0.085	0.00068
HeartRate	0.058	-0.017	-0.056	0.14	0.085	1	-0.11
RiskLevel	-0.22	-0.19	-0.25	-0.48	0.00068	-0.11	1

Figure 2: Correlation Heat Map

- Blood Sugar is highly correlated with Risk Level
- Diastolic BP > Age > Systolic BP have significant correlation with Risk Level

1.3 Random Forest feature importance

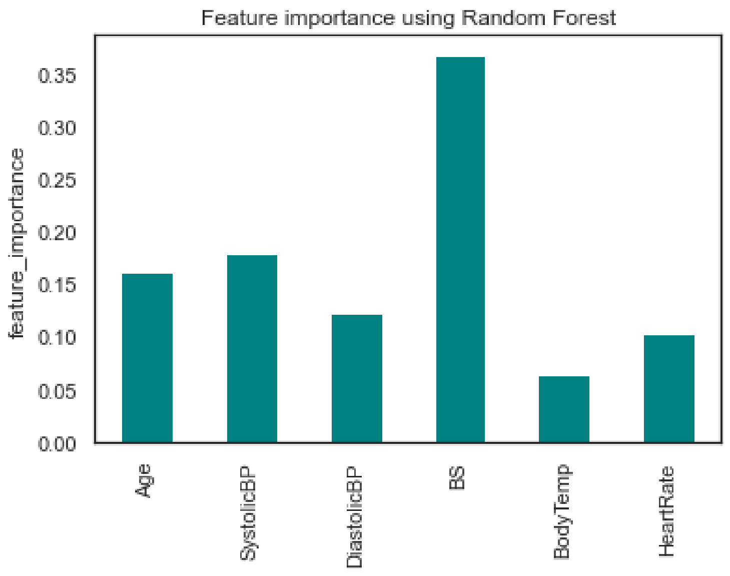


Figure 3: Feature selection using Random Forest

- Blood Sugar is most importance feature using Random forest feature selection

1.4 Coefficient importance

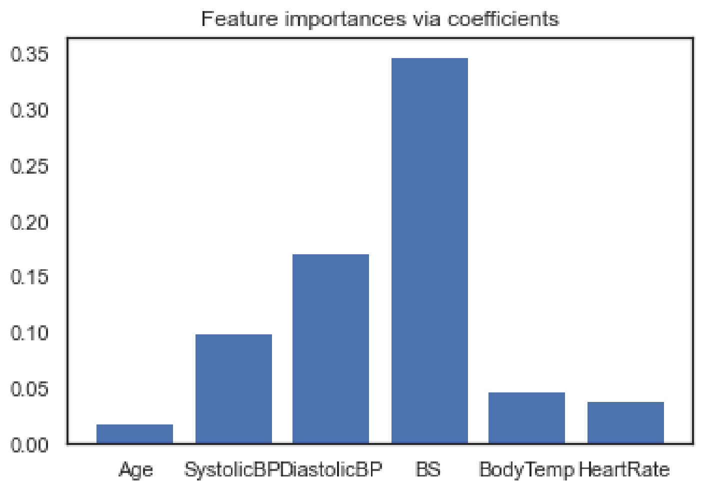


Figure 4: Feature selection using Coefficient importance

- Blood Sugar is most importance feature using Random forest feature selection

1.5 Mutual information classification

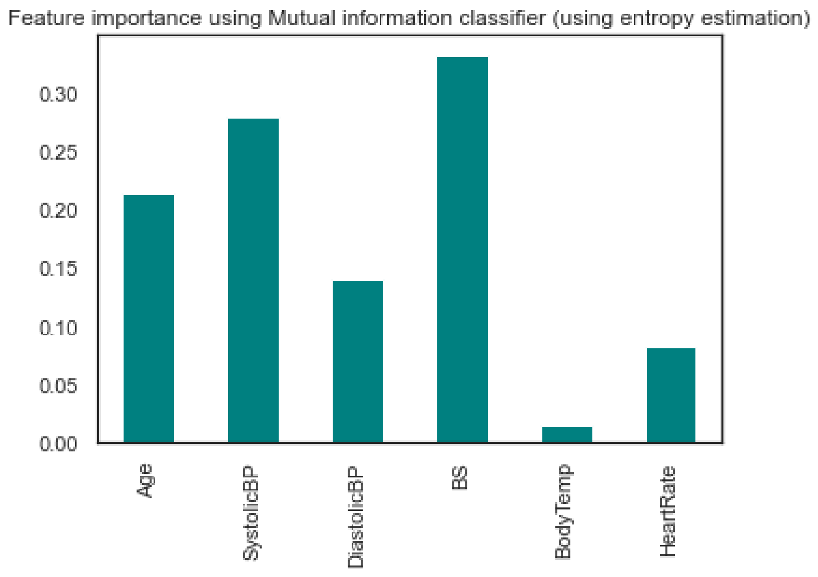


Figure 5: Feature selection using Mutual information classification

1.6 Recursive Feature selection using Decision Tree

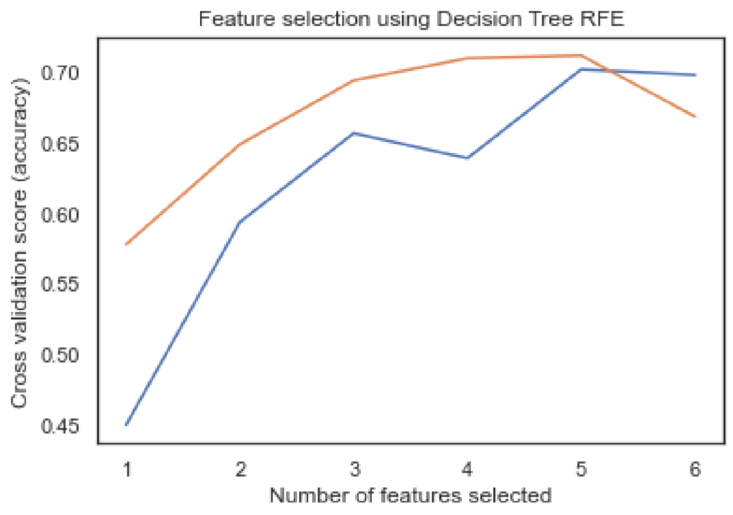


Figure 6: Feature selection using Decision Tree (RFE method)

1.7 Recursive Feature selection using SVM

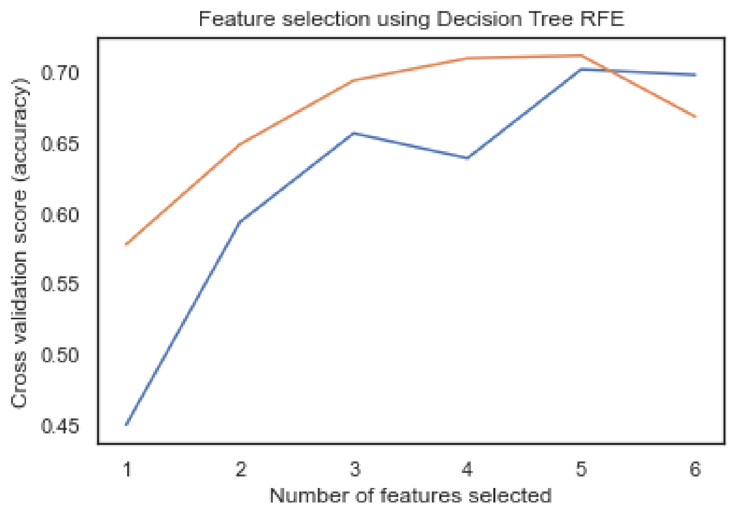


Figure 7: Feature selection using SVM (RFE method)

- recursive feature selection used
- Both in backward and forward RFE blood sugar and Diastolic BP are important features
- minimum 5 features to be selected for high accuracy, but 6 were retained as accuracy did not fall from 6th feature addition

2 Multi-collinearity check

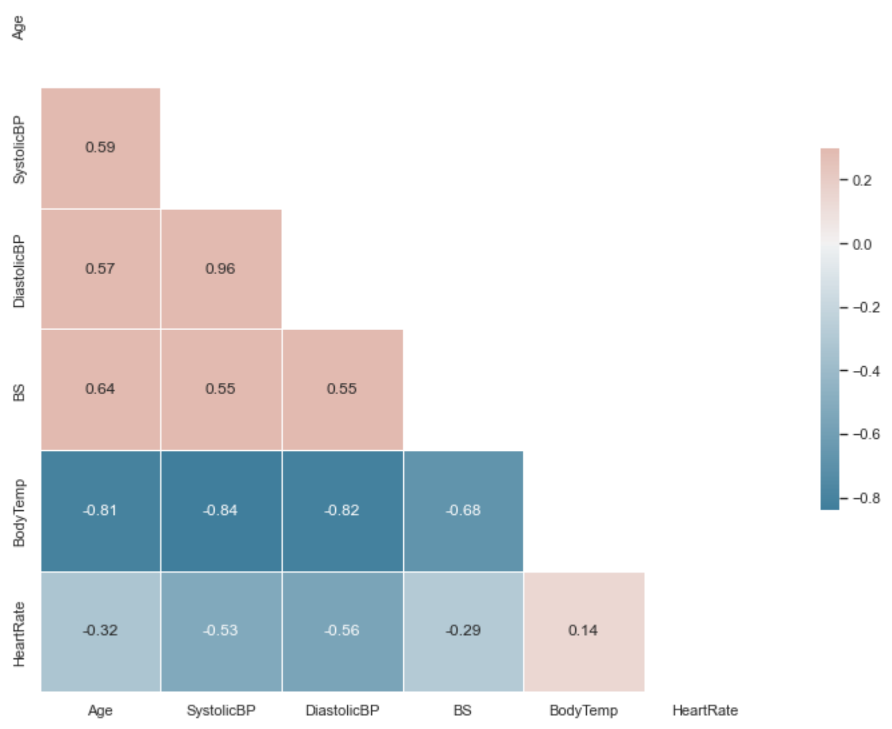


Figure 8: Checking for multicollinearity between features

- No perfect col-linearity observed, but correlation among attributes

Table 3: Model Performance

	Accuracy score	Precision	Recall	F-score	ROC-AUC score
Decision Tree	0.81	0.81	0.82	0.81	0.86
Random Forest	0.82	0.82	0.82	0.82	0.87
Naive Bayes	0.50	0.50	0.48	0.48	0.61
K-NN	0.80	0.80	0.80	0.80	0.85

3 Classification Model implementation and Result

3.1 Decision Tree

- Decision tree is trained using entropy criterion and initially with default hyper-parameters in sklearn python library
- data set is split into 70:30 ratio for training:testing

3.1.1 Result

- Accuracy score: 0.810 (High out of sample accuracy)
- Precision : 0.806
- Recall : 0.817
- F-score : 0.808
- ROC AUC of test set: 0.863 (suggests high performance)

3.1.2 Confusion Matrix

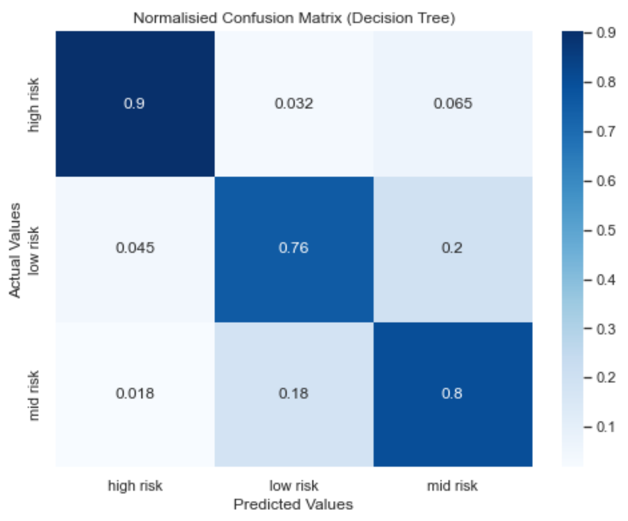


Figure 9: Normalised Confusion Matrix (Decision Tree)

3.1.3 ROC AUC Curve

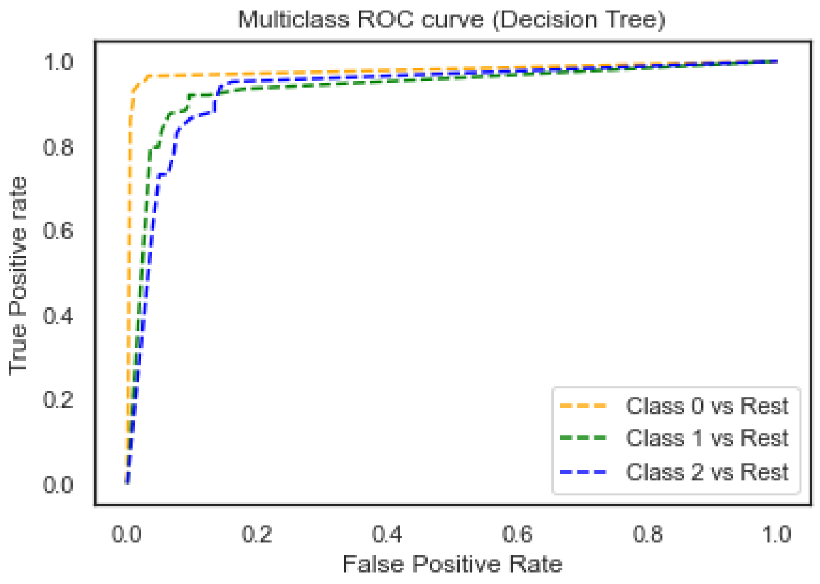


Figure 10: ROC AUC curve (Decision Tree)

3.1.4 Cross Validation Results

Data was divided into 15 equal-size partitions and accuracy score of 0.85 was achieved with a standard deviation of 0.04.

This was carried out at successively increasing height of decision tree and entropy criterion:

Table 4: Result using cross validation at different height

Tree Depth	Accuracy	Std Dev.
1	0.59	0.04
5	0.70	0.04
10	0.78	0.05
15	0.84	0.04
20	0.85	0.04

3.1.5 Hyper-parameter Tuning using GridSearchCV

Data was divided into 15 equal-size partitions and this was repeated 3 times using RepeatedStratifiedKFold function for cross validation.

For hyper-parameter tuning, GridSearchCV is used.

- Inputs for tuning is selection of criterion for division at node : Entropy vs Gini.
- Tree depth : range between 1 and 20
- Result from tuning : entropy as criterion and 19 as tree depth, but depth of 8 was selected as 0.8 accuracy was achieved at tree depth 8 and higher possiblity of overfitting at higher depth.
- decision tree is plot using Graphviz with tree depth 8.

3.2 Random Forest

- Random Forest is trained using 10 estimators and entropy as criterion for split. Initially with default hyper-parameters in sklearn python library
- data set is split into 70:30 ratio for training:testing

3.2.1 Result

- Accuracy score: 0.820 (High out of sample accuracy)
- Precision : 0.816
- Recall : 0.824
- F-score : 0.817
- ROC AUC of test set: 0.868 (suggests high performance)

3.2.2 Confusion Matrix

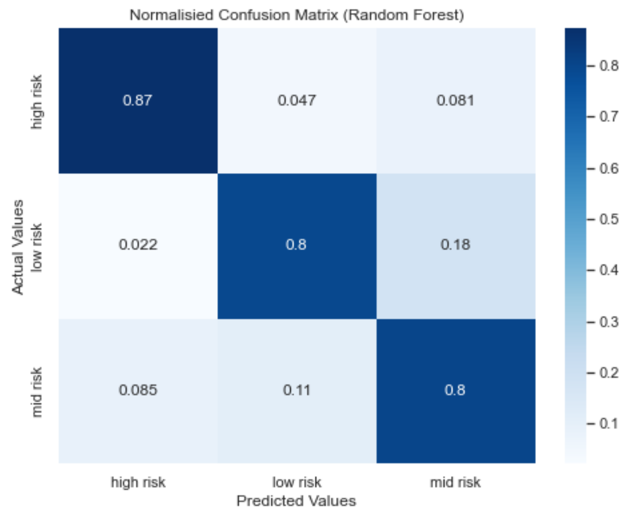


Figure 11: Normalised Confusion Matrix (Random Forest)

3.2.3 ROC AUC Curve

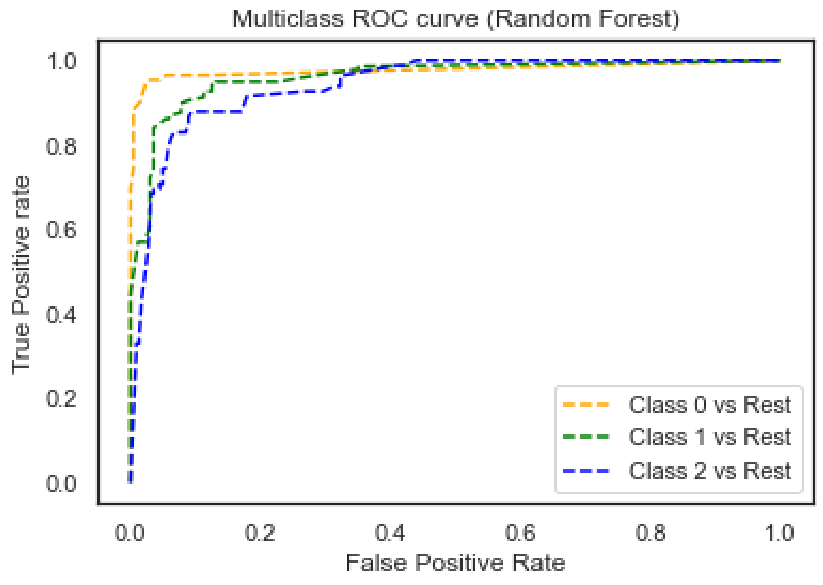


Figure 12: ROC AUC curve (Random Forest

3.2.4 Cross Validation Results

Data was divided into 15 equal-size partitions and accuracy score of 0.85 was achieved with a standard deviation of 0.13.

This was carried out at successively increasing height of decision tree and entropy criterion:

Table 5: Result using cross validation at different estimator size

Estimator size	Accuracy	Std Dev.
1	0.78	0.04
5	0.85	0.03
10	0.85	0.03
15	0.85	0.03
20	0.85	0.03

3.2.5 Hyper-parameter Tuning using GridSearchCV

Data was divided into 15 equal-size partitions and this was repeated 3 times using RepeatedStratifiedKFold function for cross validation.

For hyper-parameter tuning, GridSearchCV is used.

- Inputs for tuning is selection of criterion for division at node : Entropy vs Gini.
- Estimator size : range between 1 and 20
- Result from tuning : entropy as criterion and 20 as estimator size, but estimator size of 10 was selected as 0.85 accuracy was achieved at size 10 and higher possiblity of overfitting at higher estimator size.

3.3 Naive Bayes

- Random Forest is trained with default hyper-parameters in sklearn python library
- data set is split into 70:30 ratio for training:testing

3.3.1 Result

- Accuracy score: 0.518 (poor out of sample accuracy)
- Precision : 0.499
- Recall : 0.480
- F-score : 0.482
- ROC AUC of test set: 0.061 (suggests low performance)

3.3.2 Confusion Matrix

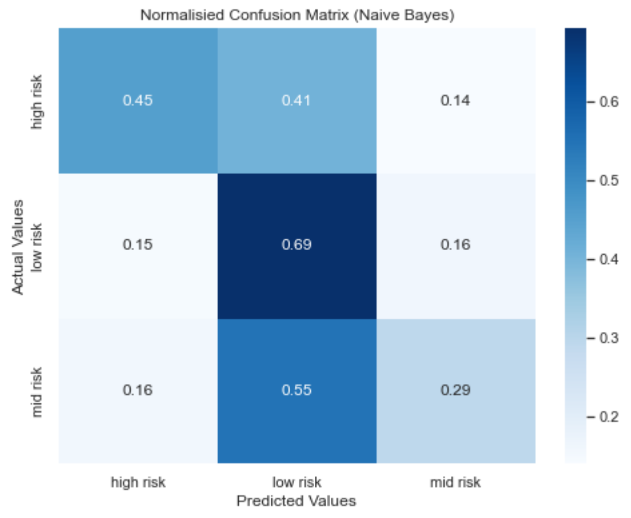


Figure 13: Normalised Confusion Matrix (Naive Bayes)

3.3.3 ROC AUC Curve

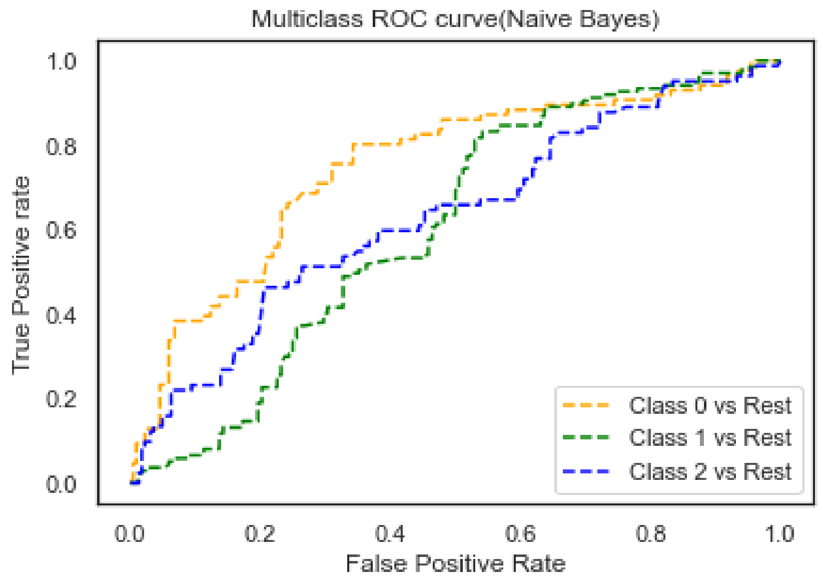


Figure 14: ROC AUC curve (Naive Bayes

3.3.4 Cross Validation Results

Data was divided into 15 equal-size partitions and accuracy score of 0.45 was achieved with a standard deviation of 0.10.

This was carried out at successively increasing alpha value from 0.05 to 0.95 with increment of 0.1 at each step and fit_prior = True :

Table 6: Result using cross validation at different alpha values

Alpha	Accuracy	Std Dev.
0.1	0.46	0.06
0.3	0.46	0.06
0.5	0.46	0.06
0.7	0.46	0.06
1.0	0.46	0.06

3.3.5 Hyper-parameter Tuning using GridSearchCV

Data was divided into 15 equal-size partitions and this was repeated 3 times using RepeatedStratifiedKFold function for cross validation.

For hyper-parameter tuning, GridSearchCV is used.

- Inputs for tuning is selection of criterion for division at node : alpha in range 0 to 1.
- fit prior boolean value True or False
- Result from tuning : alpha = 0.45 and fit_prior = True are best hyper-parameter

3.4 K Nearest Neighbour

- KNN is trained with nearest_neighbors = 1, metric = 'minkowski', p = 2 and with default hyper-parameters in sklearn python library
- data set is split into 70:30 ratio for training:testing

3.4.1 Result

- Accuracy score: 0.803 (good out of sample accuracy)
- Precision : 0.802
- Recall : 0.803
- F-score : 0.803
- ROC AUC of test set: 0.852 (suggests high performance)

3.4.2 Confusion Matrix

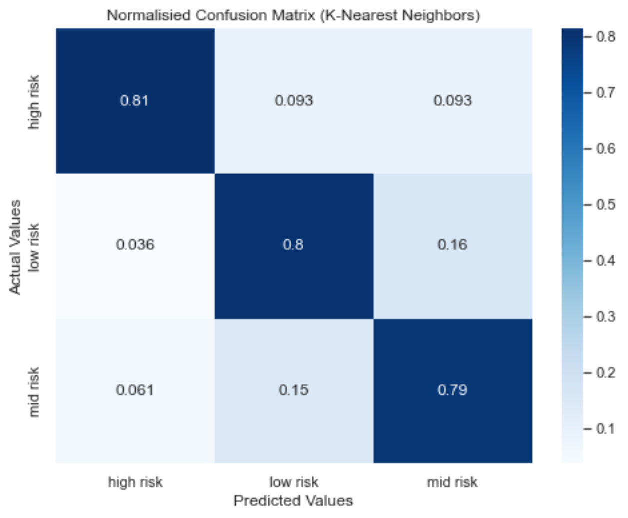


Figure 15: Normalised Confusion Matrix (KNN)

3.4.3 ROC AUC Curve

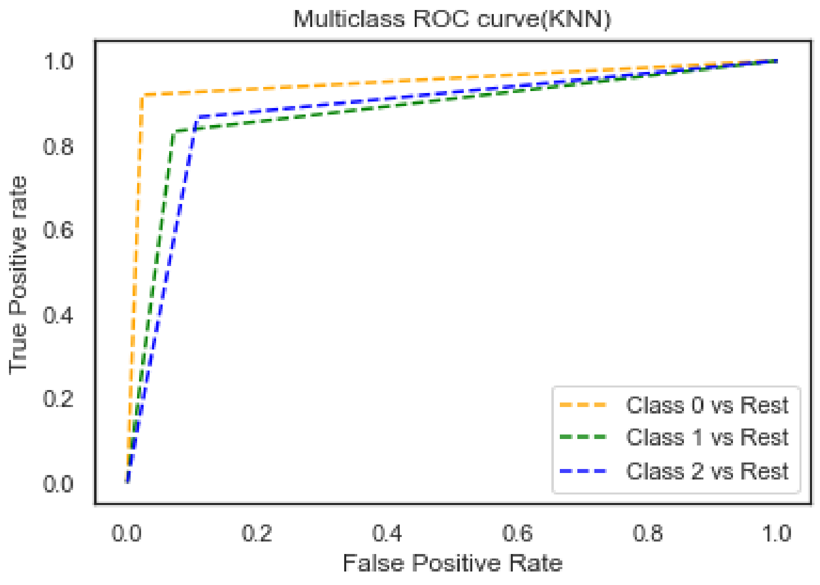


Figure 16: ROC AUC curve (KNN)

3.4.4 Cross Validation Results

Data was divided into 15 equal-size partitions and accuracy score of 0.84 was achieved with a standard deviation of 0.04.

This was carried out at successively decreasing nearest neighbour from 10 to 1 with increment of 0.1 at each step and minkowski metric is used with p = 2 which is euclidean distance :

Table 7: Result using cross validation at different n_neighbour values

n_neighbour	Accuracy	Std Dev.
10	0.69	0.05
7	0.70	0.05
5	0.71	0.05
3	0.71	0.05
1	0.84	0.04

3.4.5 Hyper-parameter Tuning using GridSearchCV

Data was divided into 15 equal-size partitions and this was repeated 3 times using RepeatedStratifiedKFold function for cross validation.

For hyper-parameter tuning, GridSearchCV is used.

- Inputs for tuning is selection of criterion: nearest neighbour in range 1 to 10. and p value in range of 1 to 5 for minokswi distance metric
- Result from tuning : nearest neighbour = 1 and p = 1 are best hyper-parameter but p = 2 was chosen as almost same accuracy

4 Performance of classification algorithm comparison

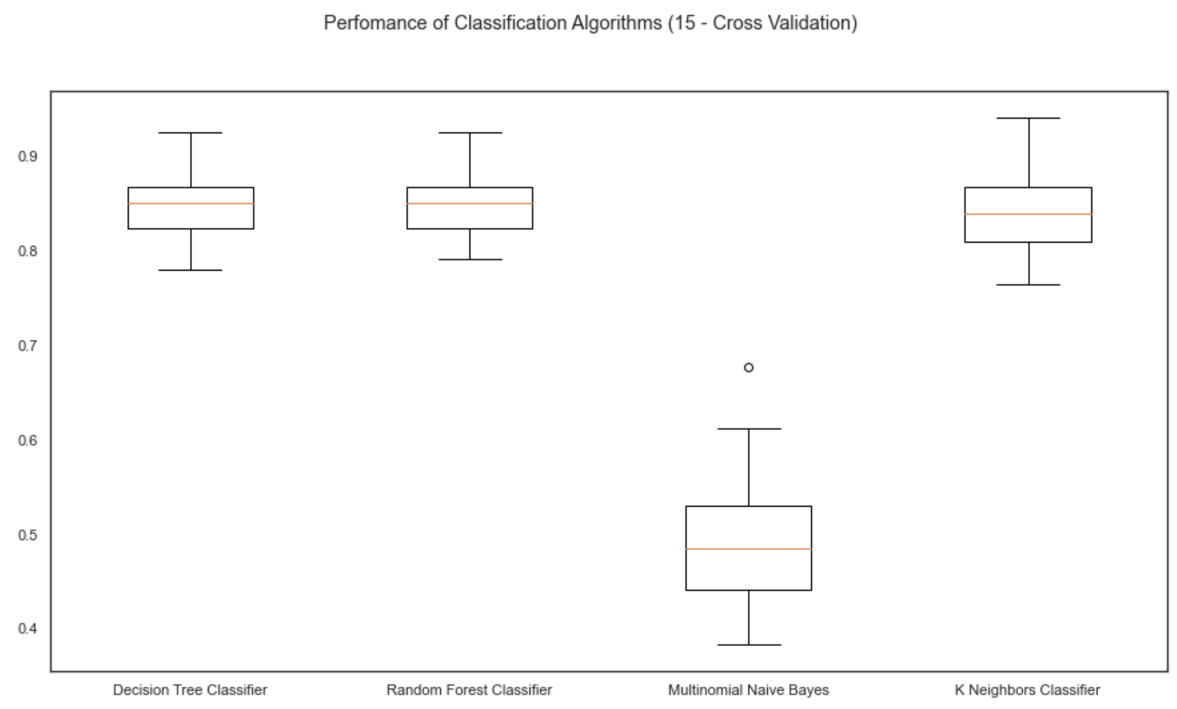


Figure 17: classification algorithm accuracy comparison using cross validation

Random Forest > Decision Tree > KNN Classifer > Multinominal Naive Bayes

Question 2

(a) Implement Apriori and FP-growth algorithm. Cite any sources helpful to you for implementing the algorithms.

Sol. Please refer jupyter notebook and github/gitlab repository for code.

(b) Modify the algorithms to achieve the same task (preferably with some improvement). Clearly mention the difference in the modified algorithm.

Implementation Apriori using Hashtree is modification in aprori alogrithm for improvement.

Sol. Please refer jupyter notebook and github/gitlab repository for code.