IFN 509: Data Exploration and Mining

Assessment 2

Team Name: [6 is not acceptable, guys!]

Group No. [Assignment 2. 3]

Student Name	Student Id
Shaun Benny Kurian	N11443715
Ngo Chi Khang Nguyen	N11424893
Danny Jeong	N11491205

	Shaun B.	Khang N.	Danny J.
Shaun B.	<100 %>	<100 %>	< 100%>
Khang N.	<100 %>	<100 %>	<100 %>
Danny J.	<100 %>	<100 %>	<100 %>

Replace the % contribution with an appropriate number if it is not an equal contribution.

Contents

Case Study 1

1. What pre-processing was required on the dataset before building the association mining model? What variables did you include in the analysis? Justify your choice.

1) As we want to generate association rules from items purchased by each invoice number, we need variables like InvoiceNo, Description, and Quantity.
2) InvoiceDate will be used to sort the data in order to implement the sequential mining.
3) CustomerID has 3,985 missing data, so we decided to use InvoiceNo, which has no missing values. The description has also 588 missing data, which is a relatively small number, so we decided to remove the rows.

```
missing_values = df.isna()
print(missing_values.sum())
```

```
InvoiceNo         0
StockCode         0
Description     588
Quantity          0
InvoiceDate       0
UnitPrice         0
CustomerID     3985
Country           0
dtype: int64
```

4) StockCode, Quantity, UnitPrice, CustomerID, and Country are not used in this analysis, so we have dropped them to reduce the computational complexity.
5) The data type of InvoiceNo has been changed to String type, which was int64.

2. Conduct association mining and answer the following:

a. What ‘min_support’ and `min_confidence’ thresholds were set for this mining exercise? Rationalize why these values were chosen.

1) We have set min_support=0.025 and min_confidence=0.20. With this condition, the outcomes show some meaningful association rules with balanced values.
2) When min_support=0.20 was set, only three rules are generated, which cannot be a meaningful analysis.
3) When min_support=0.15 was set, only six rules are generated, which is still less outcome.
4) When min_support=0.10 was set, left_side (antecedents) had no values, which does not give any meaningful associations rules.
5) When min_support=0.05 was set, left_side (antecedents) had only 6 values, which gave some meaningful associations rules, but not enough.
6) When min_support=0.01 was set, the support values of outcomes were only 1.3%, which is too rare, and some of the confidence values less than 10% as well. These values won’t give any significant meaning.

b. Report the top-5 (interesting) rules and interpret them.

1) We initially sorted the rules based on their 'Lift' values. From the top-ranked rules, we then identified the most compelling ones, emphasizing those with a high 'Confidence'.
2) 2.6% of the order that purchased ‘JUMBO BAG PINK POLKADOT & LUNCH BAG RED RETROSPOT’ also bought ‘JUMBO BAG RED RETROSPOT’. This usually happens in 77.1% of the total transactions. These items are 3.5 times as likely to be bought together than if they were chosen independently. (Lift 3.5: A strong rule)

3) 8.6% of the order that purchased ‘JUMBO BAG PINK POLKADOT’ also bought ‘JUMBO BAG RED RETROSPOT’. This usually happens in 67.7% of the total transactions. These items are 3.1 times as likely to be bought together than if they were chosen independently. (Lift 3.1: A strong rule)
4) 6.7% of the order that purchased ‘LUNCH BAG BLACK SKULL’ also bought ‘LUNCH BAG RED RETROSPOT’. This usually happens in 50.3% of the total transactions. These items are 3.08 times as likely to be bought together than if they were chosen independently. (Lift 3.08: A strong rule)
5) 6.7% of the order that purchased ‘LUNCH BAG RED RETROSPOT’ also bought ‘LUNCH BAG BLACK SKULL’. This usually happens in 40.9% of the total transactions. These items are 3.08 times as likely to be bought together than if they were chosen independently. (Lift 3.08: A strong rule)
6) 8.6% of the order that purchased ‘JUMBO BAG RED RETROSPOT’ also bought ‘JUMBO BAG PINK POLKADOT’. This usually happens in 39.4% of the total transactions. These items are 3.1 times as likely to be bought together than if they were chosen independently. (Lift 3.1: A strong rule)

3. List top-5 common items that customers have purchased along with ‘JUMBO BAG PINK POLKADOT’.

```
ans3 = result3_df.loc[result3_df['Left_side']=='JUMBO BAG PINK POLKADOT']
ans3.sort_values(by = 'Lift', ascending = False).head(5)
```

	Left_side	Right_side	Support	Confidence	Lift
124	JUMBO BAG PINK POLKADOT	LUNCH BAG RED RETROSPOT,JUMBO BAG RED RETROSPOT	0.026024	0.204433	3.378272
28	JUMBO BAG PINK POLKADOT	JUMBO BAG RED RETROSPOT	0.086225	0.677340	3.097891
34	JUMBO BAG PINK POLKADOT	LUNCH BAG RED RETROSPOT	0.033758	0.265189	1.622332
31	JUMBO BAG PINK POLKADOT	LUNCH BAG BLACK SKULL.	0.026756	0.210181	1.579739
37	JUMBO BAG PINK POLKADOT	WHITE HANGING HEART T-LIGHT HOLDER	0.026860	0.211002	0.893303

4. Can you perform sequence analysis on this dataset? If yes, present your results. If not, rationalize why.

```
get_association_rules(sequences, 0.01,0.01)
```

	Left_rule	Right_rule	Support	Confidence
0	[WHITE HANGING HEART T-LIGHT HOLDER]	[ASSORTED COLOUR BIRD ORNAMENT]	0.016618	0.070354
1	[ASSORTED COLOUR BIRD ORNAMENT]	[WHITE HANGING HEART T-LIGHT HOLDER]	0.017454	0.114777
2	[WHITE HANGING HEART T-LIGHT HOLDER]	[JUMBO BAG PINK POLKADOT]	0.014632	0.061947
3	[JUMBO BAG PINK POLKADOT]	[WHITE HANGING HEART T-LIGHT HOLDER]	0.012960	0.101806
4	[WHITE HANGING HEART T-LIGHT HOLDER]	[LUNCH BAG RED RETROSPOT]	0.020276	0.085841
...	...	...	...	...
98	[PARTY BUNTING]	[REGENCY CAKESTAND 3 TIER]	0.020276	0.115065
99	[REGENCY CAKESTAND 3 TIER]	[SET OF 3 CAKE TINS PANTRY DESIGN]	0.016513	0.079437
100	[SET OF 3 CAKE TINS PANTRY DESIGN]	[REGENCY CAKESTAND 3 TIER]	0.016095	0.111191
101	[PARTY BUNTING]	[SET OF 3 CAKE TINS PANTRY DESIGN]	0.013482	0.076512
102	[SET OF 3 CAKE TINS PANTRY DESIGN]	[PARTY BUNTING]	0.012542	0.086643

103 rows × 4 columns

5. In what ways can the results of this task be utilized by the relevant decision-makers?

Association and sequential mining techniques offer valuable insights for decision-makers in online retail. These methods uncover patterns of frequently co-purchased items, which can be leveraged for optimizing various aspects of the online store.

One key application is enhancing product placement on store pages. By identifying strong product associations, decision-makers can strategically position items to boost cross-selling and upselling opportunities. For example, our analysis revealed that featuring jumbo bags with pink polka dots alongside those with red retro spots increases the likelihood of customers purchasing both items by over 67%.

Furthermore, this insight can be integrated into recommendation systems. For instance, products like party bunting, cake stands, and a set of 3 cake tins with a pantry design theme can be grouped together on a single page, potentially boosting sales.

Online store owners can also fine-tune their recommendation algorithms. For instance, after a customer purchases a 'natural slate heart chalkboard,' the algorithm can suggest adding a 'heart of wicker small,' showing 24% of confidence value, encouraging customers to purchase both items.

Case Study 2

1. What pre-processing was required on the dataset (D2.csv) before building the clustering model?

1) The variables 'race,' 'age,' and 'chlorpropamide' contain missing values. Fortunately, the quantity of missing data is relatively small compared to the overall dataset. As a straightforward solution, we decided to remove the rows containing these missing values.
2) The data type of the variables 'change' and 'diabetesMed' is boolean. To facilitate further analysis, we have changed the value 'True' to '1' and 'False' to '0'.
3) Since we are focusing on diabetic patients only, we have selected the data where "diabetesMed" value is true.

```
df['change'] = df['change'].astype(int)
df['diabetesMed'] = df['diabetesMed'].astype(int)
```

```
print(len(df))
diabetic_patients = df[df['diabetesMed'] == 1]
print(len(diabetic_patients))
```

```
35318
27575
```

2. Build a clustering model to profile the characteristics of diabetic patients. Answer the following:

a. What clustering algorithm have you used?

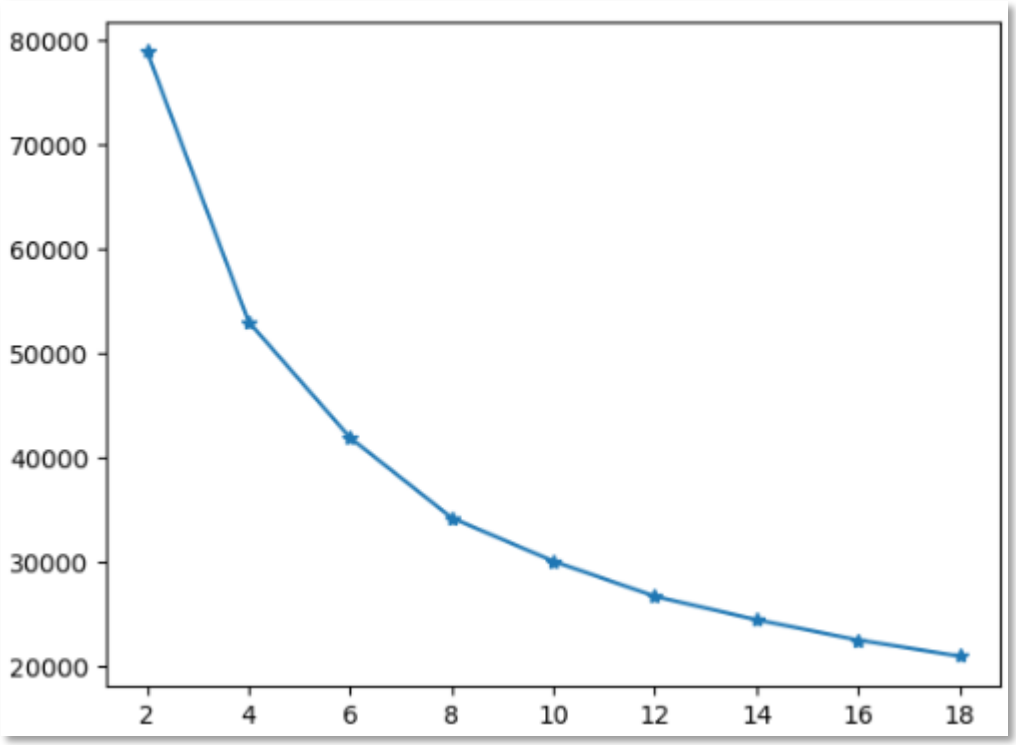
We have used K-means clustering algorithm because mainly we deal with numerical variables to analyse.

b. List the attributes used in this analysis.

1) time_in_hospital
2) num_lab_procedures
3) num_medications
4) number_outpatient

c. What is the optimal number of clusters identified? How did you reach this optimal number?

1) The optimal number of clusters is 4.
2) To identify the optimal number of clusters (K), we used both the elbow method and the silhouette method. The elbow method graph demonstrates a decreasing trend in the sum of squared distances as the number of clusters increases. (Thokagevistk, K., Millier, A., Lenert, L., Sadikhov, S., Moreno, S., & Toumi, M., 2016) The 'elbow point' marks where this decrease starts to slow down, indicating the onset of potential overfitting. In our case, the elbow point falls somewhere between 4 and 6 clusters.



3) As the elbow method may not always yield a distinct K, we computed silhouette scores. The silhouette score, ranging from -1 to 1, measures how well data points match their own clusters compared to neighbouring clusters. A higher silhouette score indicates better clustering. Our analysis revealed that the silhouette score for k=4 is higher than that for k=6, making k=4 the optimal choice for our model.

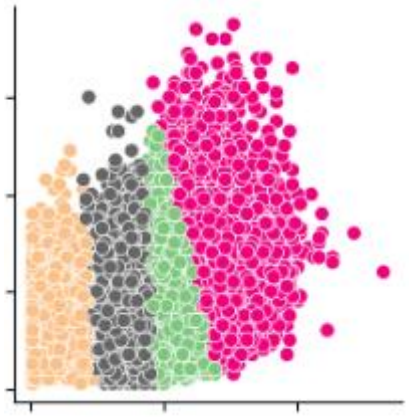
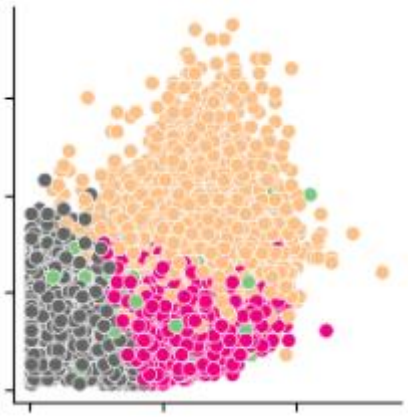
```
from sklearn.metrics import silhouette_score
print(clusters[1])
print("Silhouette score for k=4", silhouette_score(X, clusters[1].predict(X)))
print(clusters[2])
print("Silhouette score for k=6", silhouette_score(X, clusters[2].predict(X)))
```

```
KMeans(n_clusters=4, random_state=42)
Silhouette score for k=4 0.26761692852225544
KMeans(n_clusters=6, random_state=42)
Silhouette score for k=6 0.24544727935752794
```

d. Did you normalize/standardize the variables? What was its effect on the model – does the variable normalization/standardize process enable a better clustering solution?

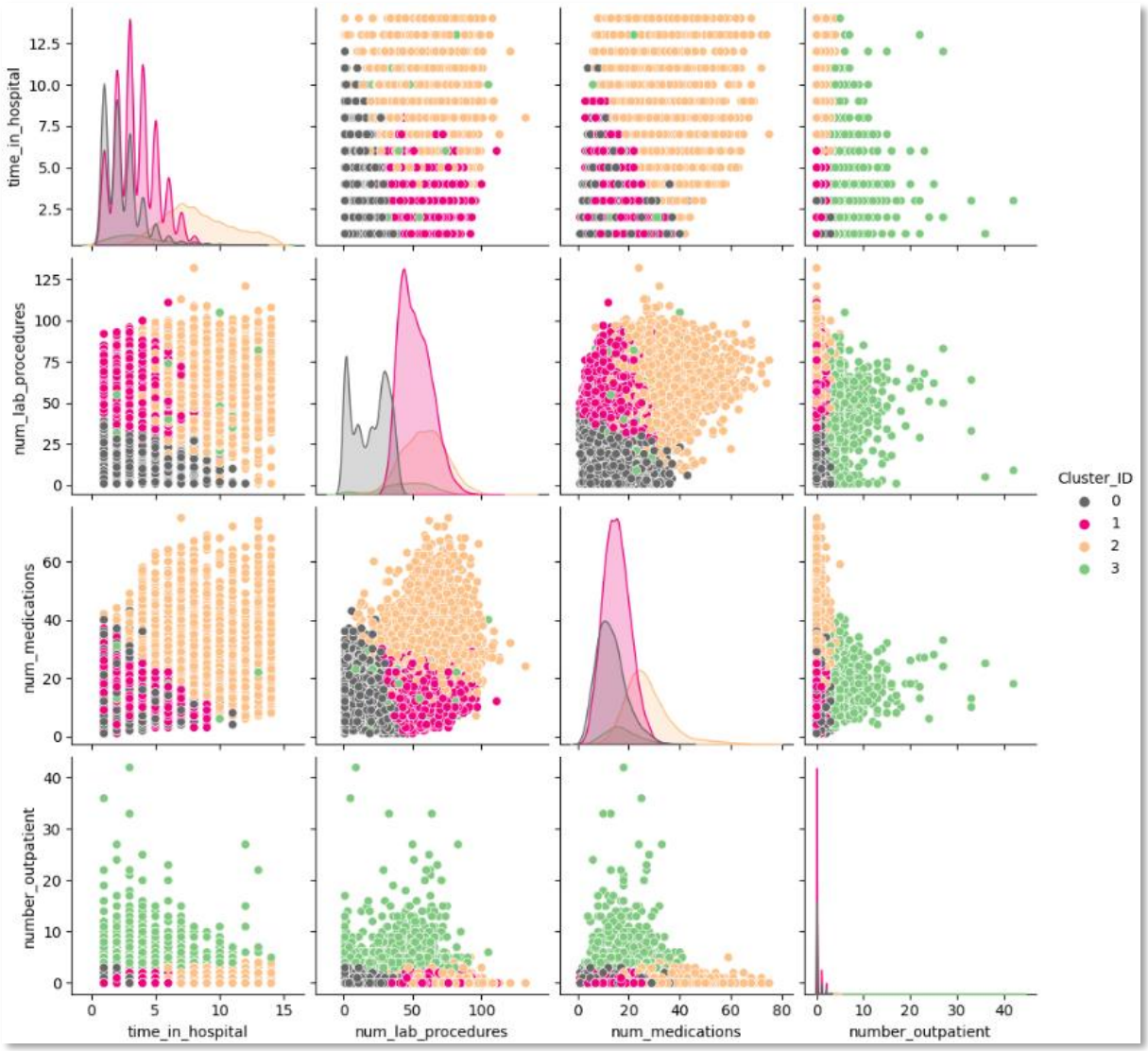
Clustering without Normalization	Clustering after Normalization
Sum of intra-cluster distance: 2987895.688452355 Centroid locations: [3.55117863 36.33913043 15.9398638 0.3810371 9] [6.57219481 71.53250375 24.20274619 0.3833941 2] [2.6587473 8.78401728 13.33141349 0.42980562] [4.44903282 54.17333188 16.73929581 0.4164312 1]	Sum of intra-cluster distance: 52966.91187710244 Centroid locations: [-0.63809869 -1.23452365 -0.49477365 -0.1337435 4] [-0.25721237 0.37986766 -0.24641294 -0.1678044 4] [1.42224304 0.70182168 1.19647144 -0.1431119 8] [-0.10991967 0.049874 -0.018819 3.48957199]

The Sum of Intra-cluster Distance represents the total sum of squared distances between data points and their respective cluster centroids. It serves as a measure of how tightly the data points are grouped within each cluster. When analyzing the clustering without normalization, it becomes apparent that data points within the same cluster are spread out over larger distances, resulting in less compact clusters. In contrast, in the clustering performed after normalization, the Sum of Intra-cluster Distance is significantly smaller. Normalization facilitates the algorithm in creating more tightly packed clusters within the standardized feature space. It is also confirmed by the visualized plot.

Clustering without Normalization	Clustering after Normalization
	

3. For the model with the optimal number of clusters,

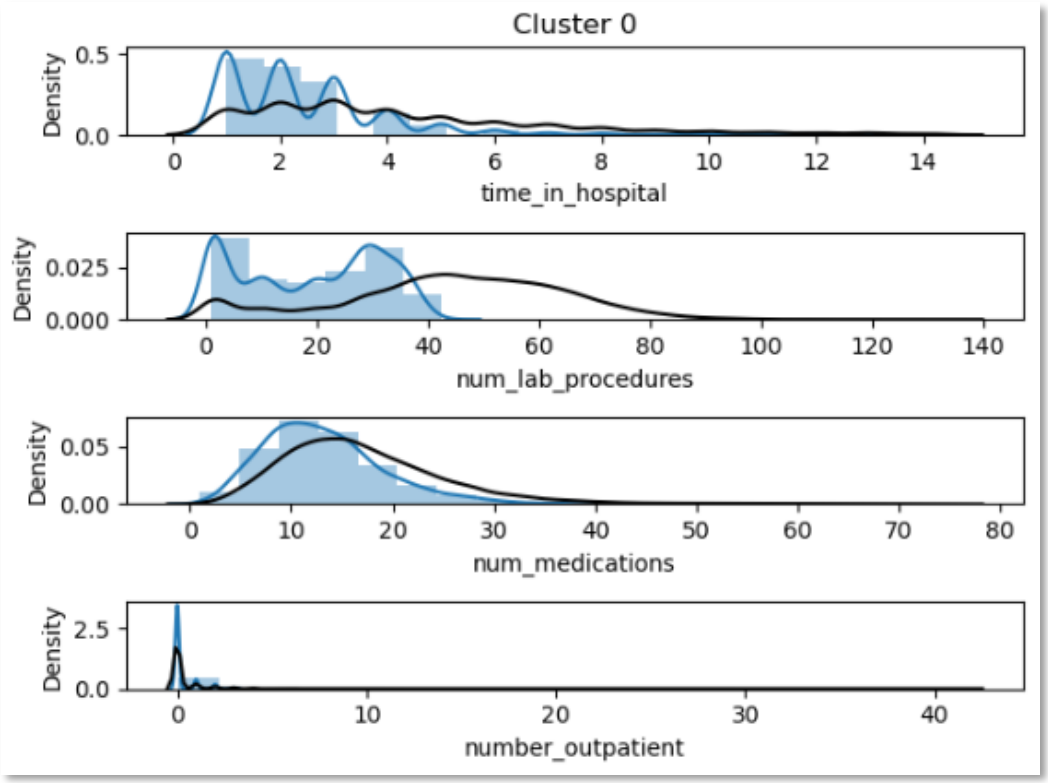
a. Visualize the clusters using ‘pairplot’ and interpret the visualization.



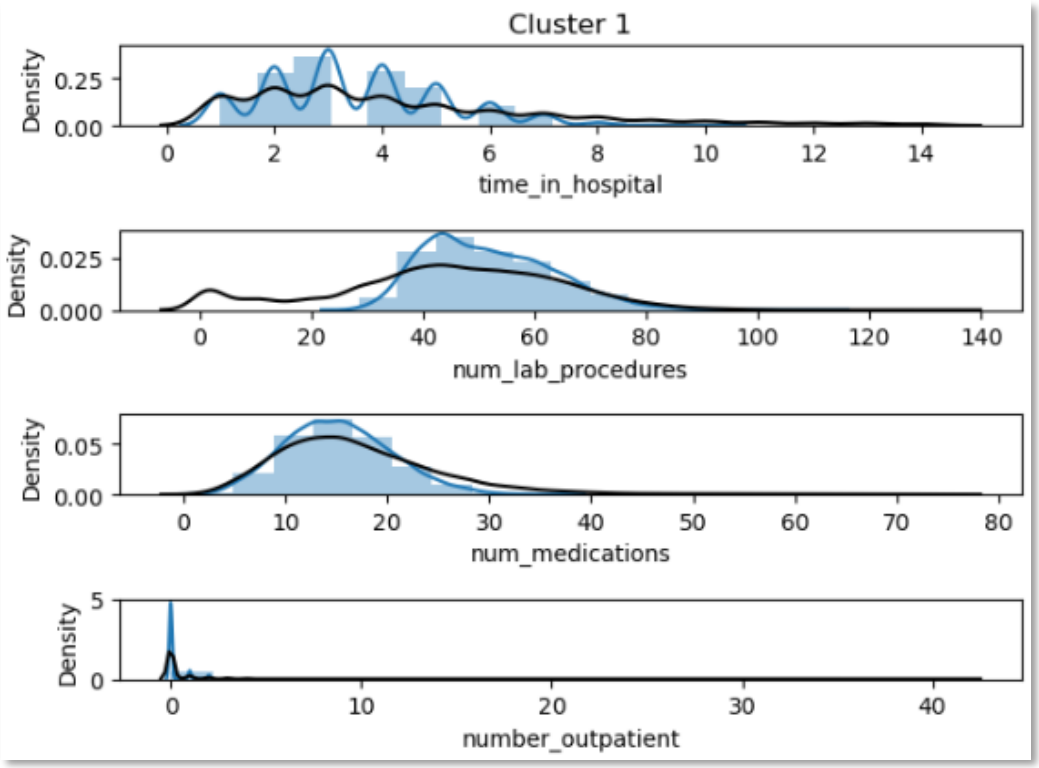
1) Cluster 0 represents diabetic patients who spent less time in the hospital and underwent fewer lab procedures. These patients also exhibit a trend of having a small number of lab procedures and medications.
2) Cluster 1 represents diabetic patients who spent less time in the hospital but had a larger number of lab procedures. These patients are characterized by a high number of lab procedures and a relatively small number of medications.
3) Cluster 2 represents diabetic patients who had longer hospital stays, underwent a larger number of lab procedures and medications. These patients are characterized by a high number of lab procedures and medications.
4) Cluster 3 represents diabetic patients who exhibit a phenomenon where medication, lab procedures, and the time spent in the hospital decrease as the number of outpatient visits to the hospital increases.

b. Characterize the nature of each cluster by giving it a descriptive label and a brief description. Hint: Use cluster distribution.

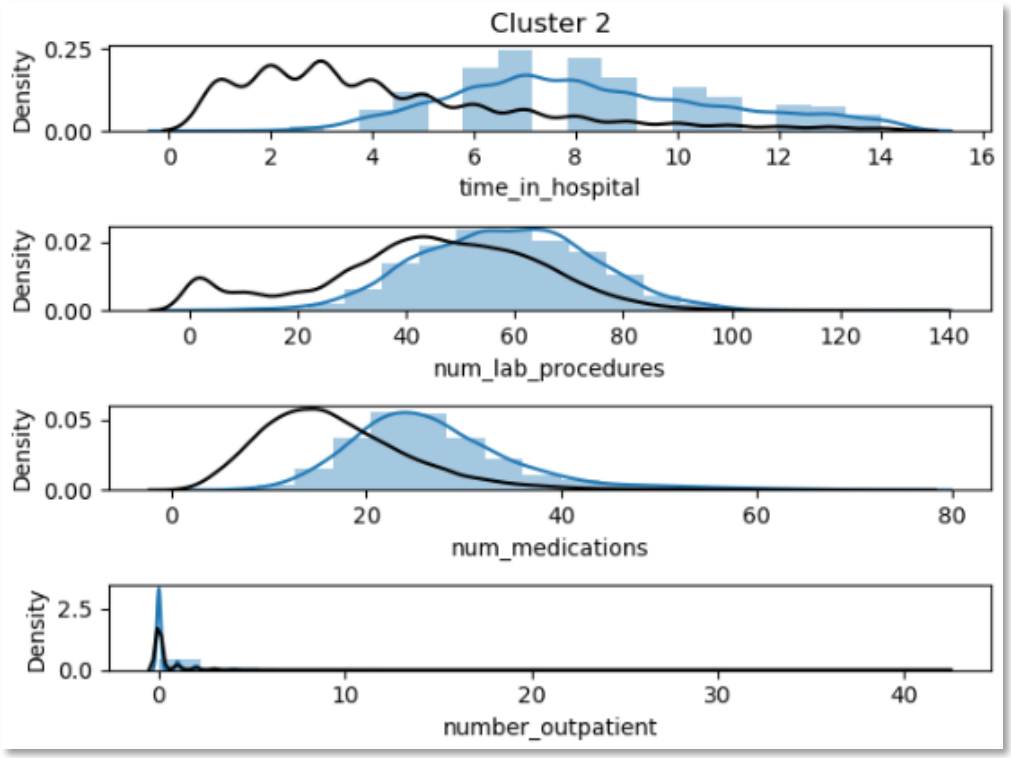
1) Diabetic patients in Cluster 0 spent time in hospital relatively shorter than other patients, underwent slightly fewer lab procedures and took fewer medications.



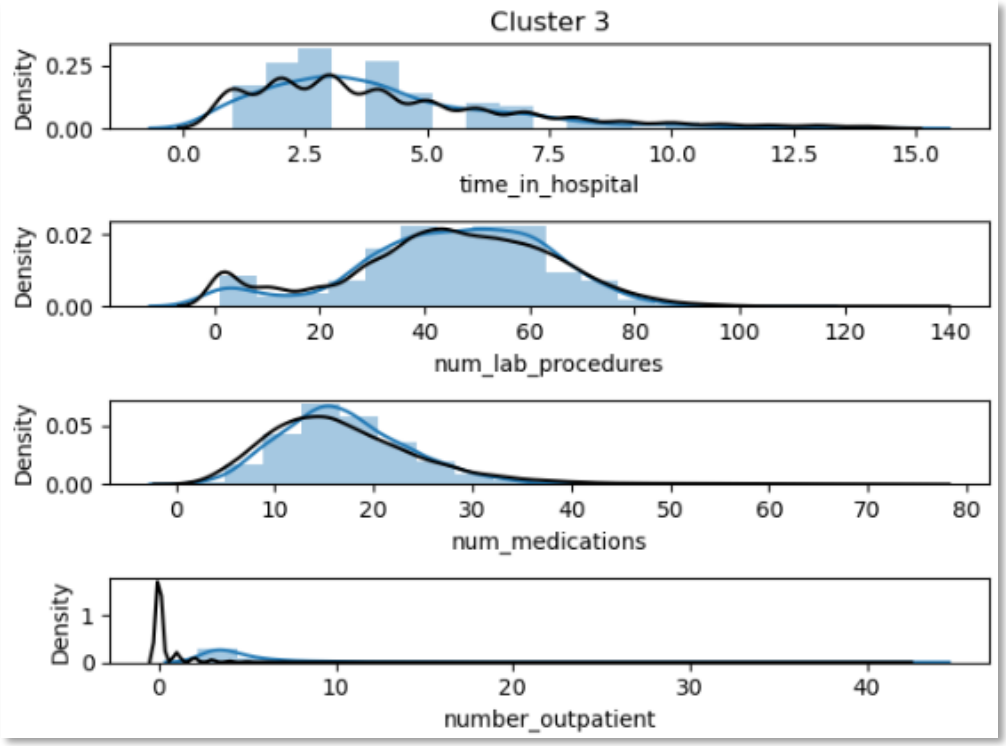
2) Cluster 1 mostly follows the pattern of distributions of all data. Diabetic patients in Cluster 1 are the median patients showing the common trends.



3) Cluster 2 shows right leaning time_in_hospital, higher num_lab_procedures, and right leaning num_medications. Diabetic patients in Cluster 2 spent more time in hospital, underwent more lab procedures, and took more medications.



4) Cluster 3 shows almost similar distributions from all records. Diabetic patients in Cluster 3 visited hospital as an outpatient more times and took a slightly greater number of medications.



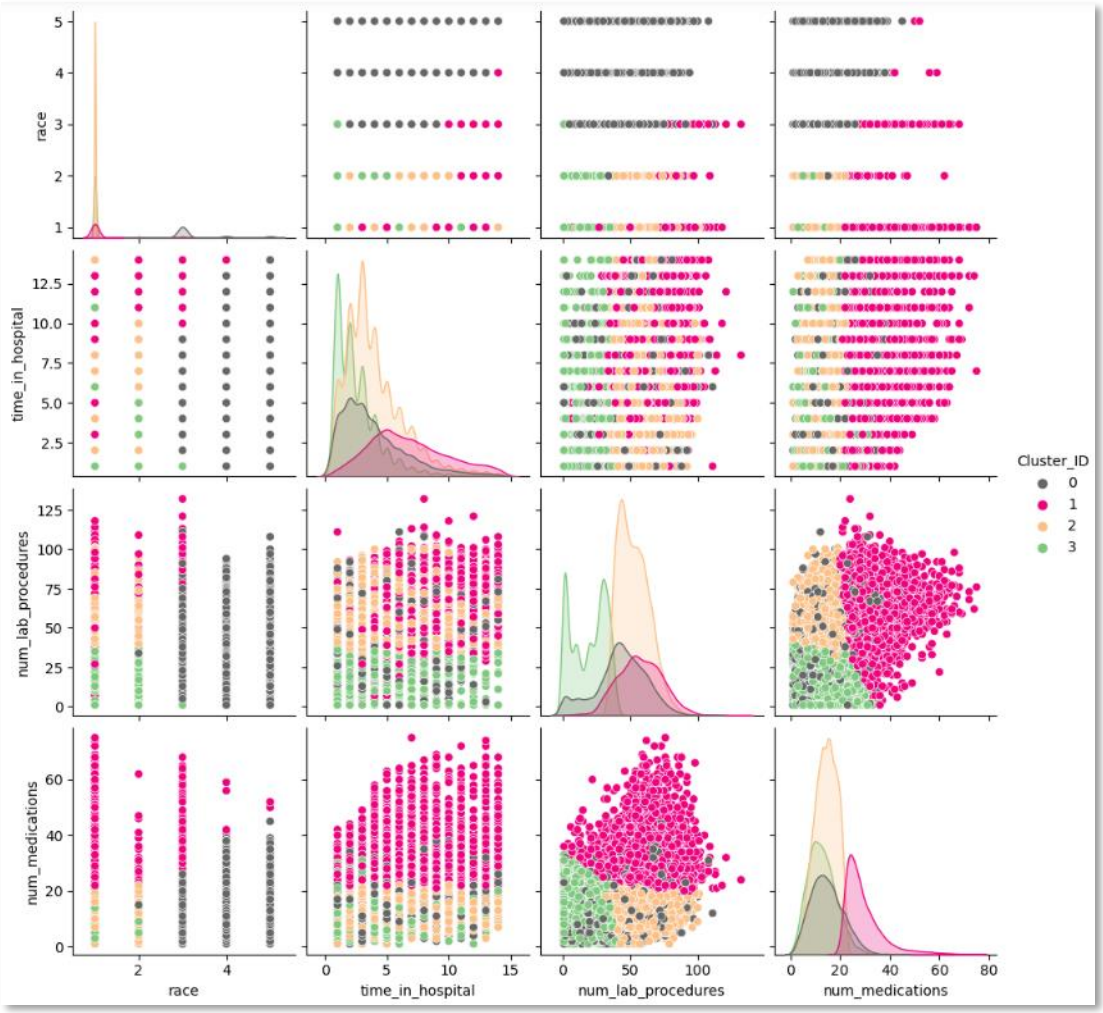
4. Build another clustering model using an algorithm that helps to profile the patients of specific races such as Asian and Caucasian.
Use the best setting (e.g., variable standardization, optimal K, etc.) obtained in the previous model.
Answer the following:

a. List the attributes used in this analysis.

race, time_in_hospital, num_lab_procedures, num_medications

b. What difference do you see in this clustering interpretation when compared to the previous one (task 3)?

1) In task 3, we were unable to analyze the categorical variables such as 'race,' 'age,' etc. However, with the K-prototypes algorithm, we were able to incorporate the 'race' categorical variable into the model.
2) In task 3, the clusters revealed the relationship between diabetic patients and other variables. Now, in this task, the clusters demonstrate the relationship between races and other variables.
3) Approximately two-thirds of Caucasians received a larger number of medications, whereas a majority of Hispanics received fewer medications.
4) The analysis indicates that most Caucasians underwent a significant number of lab procedures, while a smaller proportion of African Americans underwent lab procedures, and Hispanics had the lowest rate of lab procedures.



5. In what ways can the results of this task be utilized by the relevant decision-makers?

1) Decision-makers can allocate resources more efficiently by tailoring hospital resources and medical staff based on the specific needs of each cluster. For example, Cluster 2 in the task 3, which represents patients with longer hospital stays and a higher number of lab procedures and medications, may require more resources and specialized care.
2) Decision-makers can decide to create personalized treatment plans for diabetic patients based on their cluster. For example, patients in Cluster 3 in the task 3, who visit the hospital frequently with decreasing medication and procedures, may need tailored treatment plans with careful cares.
3) Decision-makers can design targeted patient education programs. For instance, they can focus on educating patients in Cluster 1 in the task 3 about the importance of adherence to medication, as they tend to have a high number of lab procedures but a relatively low number of medications.
4) Decision-makers can enhance outpatient services to cater to the needs of patients in Cluster 3 in the task 3. They may need more accessible outpatient care and support to reduce the frequency of hospital visits.

Case Study 3

Predictive modelling using Decision Tree

1. What pre-processing was required on the dataset (D3.csv) before decision tree modelling? What distribution split between training and test datasets have you used?

1) 'Tolbutamide' and 'acetohexamide' each have only one value, 'No,' making them unary variables that are not necessary for our further analysis. Consequently, we have removed these variables from our dataset.
2) 'admission_type_id', 'discharge_disposition_id', and 'admission_source_id' have been deleted since these variables are ID that do not give any information to build a model.
3) The data type of 'readmitted' variable was set to 'int64'. Its value 1 and 0 represent 'True' and 'False', which is Boolean data type. So, we have changed 'readmitted' data type as Bool.
4) The variables 'race,' 'age,' and 'chlorpropamide' contain missing values. Fortunately, the quantity of missing data is relatively small compared to the overall dataset. As a straightforward solution, we decided to remove the rows containing these missing values.
5) We have designated variable roles, such as 'readmitted,' as our target variable, while categorizing the remaining variables as input variables.
6) The following is the information of our dataset after pre-processing.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 50732 entries, 0 to 51765
Columns: 112 entries, age to insulin_Up
dtypes: bool(3), int64(9), uint8(100)
memory usage: 8.9 MB
```

7) We used a 70-30 split, with 70% of the data allocated to the training dataset and 30% to the test dataset.

```
rs = 10
x_train, x_test, y_train, y_test = train_test_split(X_mat, y, test_size=0.3, stratify=y, random_state=rs)
```

2. Build a decision tree using the default setting. Answer the following:

a. What is the classification accuracy of training and test datasets?

The train accuracy is closed to 1, but the test accuracy is only 0.549, which means this tree overfits on the training dataset and we need to tune the hyperparameters.

```
print("Train accuracy:", model.score(X_train, y_train))
```

```
Train accuracy: 0.9999436810092364
```

```
print("Test accuracy:", model.score(X_test, y_test))
```

```
Test accuracy: 0.549802890932983
```

b. What is the size of the tree (number of nodes and rules)?

The number of nodes is 20,817 and the number of leaves is 10,409.

```
treeObj = model.tree_
print(treeObj.node_count)
print(treeObj.n_leaves)
```

```
20817
10409
```

c. Which variable is used for the first split?

The variable used for the first split is 'number_inpatient'.

```
first_split_feature_name = X.columns[model.tree_.feature[0]]
print("The first split is made on the feature:", first_split_feature_name)
```

```
The first split is made on the feature: number_inpatient
```

d. What are the 5 important variables (in the order) in building the tree?

The 5 important variables are sorted by 'importances' and the result is as follows.

```
importances = model.feature_importances_
feature_names = X.columns
indices = np.argsort(importances)
indices = np.flip(indices, axis=0)
indices = indices[:10]
for i in indices:
    print(feature_names[i], ':', importances[i])
```

```
num_lab_procedures : 0.17683239849533594
num_medications : 0.1449038880209787
time_in_hospital : 0.09083801246483884
number_inpatient : 0.0681882639992299
age : 0.05429642322752682
```

e. What parameters have been used in building the tree? Detail them.

1) **criterion**: The function to measure the quality of a split. We have used the default value “gini” among “gini”, “entropy”, “log_loss”. (Scikit Learn, 2023)
2) **spliter**: The strategy used to choose the split at each node. We have used the default value “best” among “best”, “random”
3) **max_depth**: The maximum depth of the tree. We have used the default value “None”.
4) **min_samples_split**: The minimum number of samples required to split an internal node. We have used the default value “2”.
5) **min_samples_leafint**: The minimum number of samples required to be at a leaf node. We have used the default value “1”.
6) **min_weight_fraction_leaffloat**: The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node. We have used the default value “0.0”.
7) **max_featuresint**: The number of features to consider when looking for the best split. We have used the default value "None" among float, “auto”, “sqrt”, “log2”
8) **random_state**: Controls the randomness of the estimator. We have set it as "10".
9) **max_leaf_nodes**: Grow a tree with max_leaf_nodes in best-first fashion. We have used the default value “None”.
10) **min_impurity_decrease**: A node will be split if this split induces a decrease of the impurity greater than or equal to this value. We have used the default value “0.0”.
11) **class_weight**: Weights associated with classes in the form {class_label: weight}. If None, all classes are supposed to have weight one. We have used the default value “None”.
12) **ccp_alpha**: Complexity parameter used for Minimal Cost-Complexity Pruning. We have used the default value “0.0”.

```
model.get_params(deep = True)
```

```
{'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': None,
 'max_leaf_nodes': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'random_state': 10,
 'splitter': 'best'}
```

3. Build another decision tree tuned with GridSearchCV. Answer the following:

a. What is the classification accuracy of training and test datasets?

The training accuracy has decreased from 0.99 in the previous model to 0.63 in the current model. Conversely, the test accuracy has increased from 0.55 in the previous model to 0.63 in the current model.

```
print("Train accuracy:", cv_1.score(X_train, y_train))
print("Test accuracy:", cv_1.score(X_test, y_test))
```

```
Train accuracy: 0.6342644739806262
Test accuracy: 0.6295006570302234
```

b. What is the size of the tree (i.e. number of nodes and rules)?

The number of nodes has reduced to 125, and the number of leaves is now 63. These values are significantly lower compared to the previous tree.

```
cv_1_best = cv_1.best_estimator_
print(cv_1_best)
print('Nodes: ', cv_1_best.tree_.node_count)
print('Leaves', cv_1_best.tree_.n_leaves)
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=6, min_samples_leaf=5,
                       random_state=10)
Nodes:  125
Leaves 63
```

c. Which variable is used for the first split?

For the grid search CV, once again the variable used for the first split is 'number_inpatient'

```
first_split_feature_name2 = X.columns[cv_1_best.tree_.feature[0]]
print("The first split is made on the feature:", first_split_feature_name2)
```

```
The first split is made on the feature: number_inpatient
```

d. What are the 5 important variables (in the order) in building the tree?

The 5 important variables of the current tree are significantly changed as compared to the previous tree.

Ranking	Tree with Grid Search CV	Tree with default (Previous one)
1	number_inpatient	num_lab_procedures
2	age	num_medications
3	number_emergency	time_in_hospital
4	number_outpatient	number_inpatient
5	diabetesMed	age

```
importances = cv_1_best.feature_importances_
feature_names = X.columns
indices = np.argsort(importances)
indices = np.flip(indices, axis=0)
indices = indices[:10]
for i in indices:
    print(feature_names[i], ':', importances[i])
```

```
number_inpatient : 0.6381981338908856
age : 0.07327214295078206
number_emergency : 0.06782710488619006
number_outpatient : 0.04473199817364742
diabetesMed : 0.026753088800269036
```

e. Report if you see any evidence of model overfitting.

1) The initial tree, generated with default settings, had an extremely high train accuracy of approximately 0.9999 but a much lower test accuracy of approximately 0.5498. This discrepancy between training and test accuracy is a typical sign of overfitting.
2) The tree generated using GridSearchCV appears to have better generalization performance. The train accuracy, while still decent at 0.634, is noticeably lower than the initial tree's train accuracy. The test accuracy has significantly improved to approximately 0.629. The smaller gap between the train and test accuracies suggests that this tree is less overfit than the initial one.

4. What differences do you observe between these two decision tree models (with and without fine-tuning)? How do they compare performance-wise? Produce the ROC curve for both DTs. Explain why those changes may have happened.

1) The most notable difference is the significant reduction in the overfitting issue, as evidenced by the decrease in the accuracy gap between the training and test datasets when transitioning from the tree with default settings to the fine-tuned model.
2) Another remarkable difference is the significant reduction in the complexity of the decision tree. The fine-tuned model is much simpler and less complex, indicating that the fine-tuned model is likely to perform better on unseen data and is less likely to overfit the training data.
3) To compare performances of these two tree models, we have used classification evaluation metrics.

Tree without fine-tuning

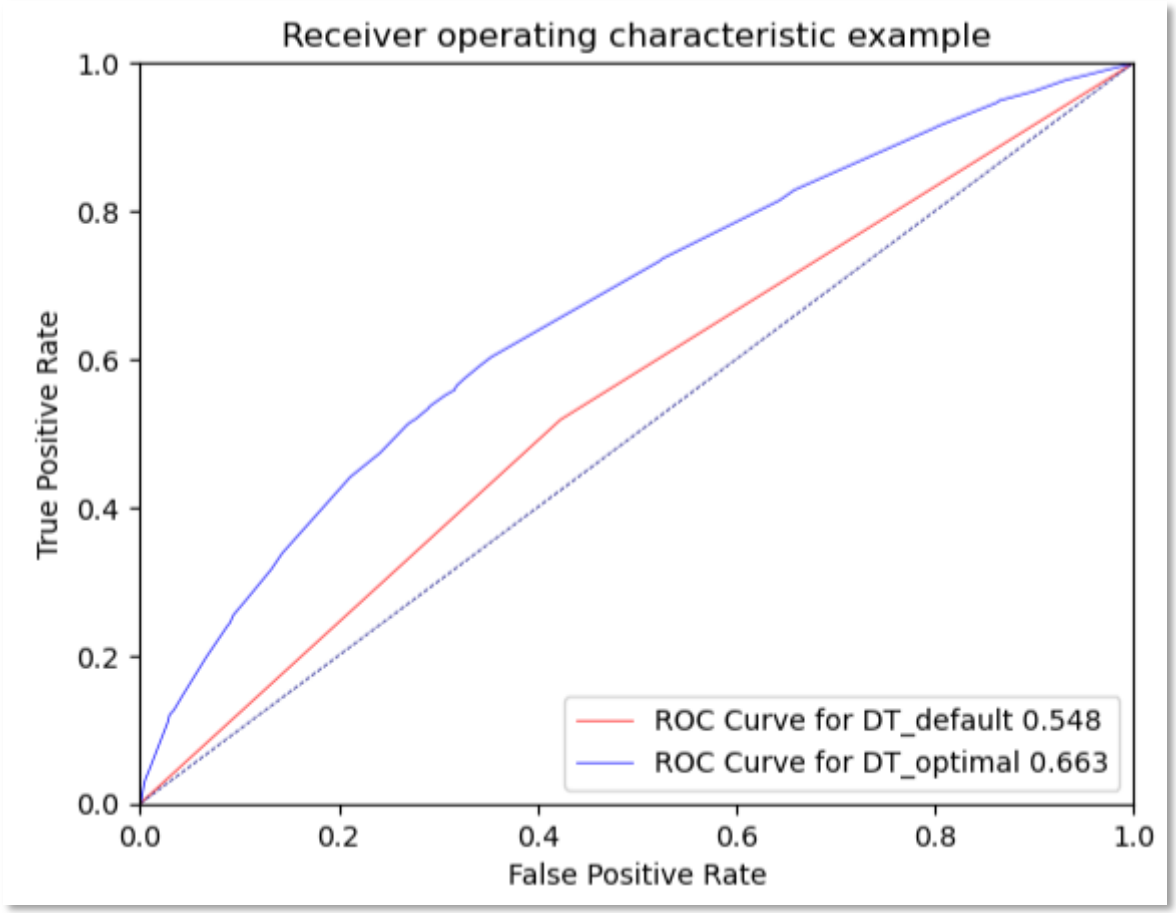
	precision	recall	f1-score	support
False	0.58	0.58	0.58	8184
True	0.51	0.52	0.52	7036
accuracy			0.55	15220
macro avg	0.55	0.55	0.55	15220
weighted avg	0.55	0.55	0.55	15220

Tree with fine-tunning

	precision	recall	f1-score	support
False	0.64	0.73	0.68	8184
True	0.62	0.51	0.56	7036
accuracy			0.63	15220
macro avg	0.63	0.62	0.62	15220
weighted avg	0.63	0.63	0.62	15220

4) **Accuracy**: The accuracy of the model generated by GridSearchCV (0.63) is higher than that of the default model (0.55). This indicates that the fine-tuned model is better at making correct predictions on the test data.
5) **Precision**: Precision measures how many of the positive predictions made by the model were correct. In the fine-tuned model, precision for the "True" class is higher (0.62) compared to the default model (0.51). This means the fine-tuned model has fewer false positives.
6) **Recall**: Recall measures how many of the actual positive or negative cases the model correctly identifies. In the fine-tuned model, recall for the "False" class is higher (0.73) compared to the default model (0.58). This means the fine-tuned model captures a higher proportion of the actual negative cases.
7) **F1-Score**: The F1-score is a metric that combines both precision and recall into a single value and is commonly used in binary classification tasks. It's calculated as the harmonic mean of precision and recall. The tree with fine-tuning generally outperforms the tree without fine-tuning for both classes, as indicated by higher F1-scores. A higher F1-score suggests a better balance between precision and recall.
8) The changes in the ROC curve and the ROC index values indicate that the fine-tuned DT_optimal model has better discriminatory power and is more effective at distinguishing between donors and non-donors compared to the default DT_default model. This improvement

in performance has been achieved due to the optimizations of hyperparameters through GridSearchCV.



5. From the better model, can you identify which patients could potentially be "readmitted"? Can you provide the general characteristics of those patients?

The most important variables are the key predictors for readmission. These are the "number_inpatient" variable, followed by "age" and "number_emergency." This can be interpreted as patients who experience more inpatient visits, fall within a specific age range, and have a history of emergency visits are more likely to be readmitted.

In other words, patients who are older, have had multiple inpatient visits, had a history of emergency visits, and had a specific medical specialty have a higher possibility of readmission. These characteristics may indicate that these patients have more complex health conditions or require ongoing medical care.

Patients have	multiple inpatient visits a history of emergency visits a specific medical specialty	a higher possibility of readmission

Predictive modelling using Regression

1. What pre-processing was required on the dataset before regression modelling? What distribution split between training and test datasets have you used?

1) The pre-processing was performed in the same manner as it was for the decision tree.
2) Deleted rows having missing data.
3) Deleted ID variables and unary variables.
4) Applied mapping into 'age' variable.
5) Changed the data type of 'readmitted' variable.
6) Applied one-hot encoding.
7) Designated variable roles: target variable(y) = readmitted, input variable(x) = other variables.
8) Set random state. (rs = 10)
9) Set distribution split. (test_size=0.3)
10) Standardized the variables used in the model.
11) We used a 70-30 split, with 70% of the data allocated to the training dataset and 30% to the test dataset.

2. Build a regression model using the default regression method with all inputs. Build another regression model tuned with GridSearchCV. Now, choose a better model to answer the following:

a. Explain why you chose that model.

Logistic Regression with Default Settings: Train Accuracy: 0.633 | Test Accuracy: 0.628

Logistic Regression with GridSearchCV: Train Accuracy: 0.633 | Test Accuracy: 0.628

In this case, both models have very similar performance in terms of accuracy, indicating that hyperparameter tuning using GridSearchCV did not significantly impact the model's performance based on accuracy. Consequently, we have selected the default regression model because it might be more practical for simplicity and computational efficiency.

b. Name the regression function used.

We have used the logistic regression, which is suitable for our purpose. Our objective for the case is to classify if a patient will be readmitted or not. The logistic regression function is used for binary classification problems where the target variable is categorical and represents two classes. It models the probability that a given input belongs to one of the two classes.

c. Did you apply standardization of variables? Why would you standardize the variables for regression mining?

We have standardized the variables for regression. In our dataset, Variable #4 (num_medications) has a wide range of values, ranging from a minimum of 1 to a maximum of 75, whereas Variable #3 (num_procedures) has a narrower range, with values ranging from 0 to 6. Due to the significant range difference between these two variables, a value of '1' can carry different weightings in the prediction process. This disparity in the variable scales may lead to inaccurate predictions. Therefore, we applied standardization to ensure that each feature has a mean of 0 and a standard deviation of 1.

```
Before scaling
-------------
Variable #0: min 1, max 9, mean 6.91 and std dev 1.82
Variable #1: min 1, max 14, mean 4.24 and std dev 2.87
Variable #2: min 1, max 121, mean 43.89 and std dev 19.99
Variable #3: min 0, max 6, mean 1.31 and std dev 1.72
Variable #4: min 1, max 75, mean 16.74 and std dev 8.08
After scaling
-------------
Variable #0: min -3.238106030760721, max 1.1483339751280162, mean -0.00 and std dev 1.00
Variable #1: min -1.1297799677610392, max 3.404898958986864, mean -0.00 and std dev 1.00
Variable #2: min -2.1451034047124797, max 3.856792422546743, mean 0.00 and std dev 1.00
Variable #3: min -0.7636830905148733, max 2.728451072550305, mean -0.00 and std dev 1.00
Variable #4: min -1.9486845888843802, max 7.214428458345791, mean -0.00 and std dev 1.00
```

d. Report the variables included in the regression model.

The following photo shows the variables included in the regression model. We have limited the number of variables to 20.

```
age : 0.092482264447527
time_in_hospital : -0.009865511596213504
num_lab_procedures : 0.04026981389565926
num_procedures : -0.06360293856516573
num_medications : 0.04523010783957481
number_outpatient : 0.11538635656198322
number_emergency : 0.26965584177963764
number_inpatient : 0.493025969157044
number_diagnoses : 0.09630118096106631
change : 0.02835840832443629
diabetesMed : 0.1658817932170065
race_AfricanAmerican : 0.0010343676245731473
race_Asian : -0.035329413612108646
race_Caucasian : 0.03399313401978708
race_Hispanic : -0.05768750327015372
race_Other : -0.023292788075029533
gender_Female : 0.019058798401231217
gender_Male : -0.018585221834064324
gender_Unknown/Invalid : -0.044521480776837756
medical_specialty_Anesthesiology : 0.0
```

e. Report the top-5 important variables (in the order) in the model.

The following shows the top 5 important variables based on the absolute values of their coefficients and sorts them in descending order.

```
number_inpatient : 0.493025969157044
number_emergency : 0.26965584177963764
diabetesMed : 0.1658817932170065
number_outpatient : 0.11538635656198322
number_diagnoses : 0.09630118096106631
```

f. What is the classification accuracy on training and test datasets?

The train accuracy is around 0.633 and the test accuracy is approximately 0.628.

```
Train accuracy: 0.633053615679207
Test accuracy: 0.6283837056504599
              precision    recall  f1-score   support

       False       0.62      0.79      0.70      8184
        True       0.64      0.44      0.52      7036

    accuracy                           0.63     15220
   macro avg       0.63      0.62      0.61     15220
weighted avg       0.63      0.63      0.62     15220
```

g. Report any sign of overfitting in this model.

We have not found any sign of overfitting. As we have seen the values of accuracies in the above question, the difference between the two accuracies is not substantial, suggesting that the model may not be severely overfitting.

3. Build another regression model on the reduced variables set. Perform dimensionality reduction with Recursive feature elimination. Tune the model with GridSearchCV to find the best parameter setting. Answer the following:

a. Was dimensionality reduction found useful in identifying a good feature set for building an accurate model?

The result shows that 88 of features have been remarkably reduced from the original 111 features to 23 features after elimination.

```
Original feature set 111
Number of features after elimination 23
```

b. What is the classification accuracy on training and test datasets?

The train accuracy is around 0.632 and the test accuracy is approximately 0.627. As compared to the previous models, both accuracies have been slightly increased.

```
Train accuracy: 0.6317582788916423
Test accuracy: 0.6273324572930354
              precision    recall  f1-score   support

       False       0.62      0.80      0.70      8184
        True       0.65      0.43      0.52      7036

    accuracy                           0.63     15220
   macro avg       0.63      0.61      0.61     15220
weighted avg       0.63      0.63      0.61     15220

{'C': 0.01}
```

c. Report any sign of overfitting.

We have not found any sign of overfitting.

d. Report the top-3 important variables (in the order) in the model.

Compared to the previous models, there have been changes in the top important variables in this current model. In the current model, 'number_outpatient' has become the most important variable, whereas in the previous model, 'number_inpatient' held that position. Additionally, 'num_medications' has now emerged as the second most important variable.

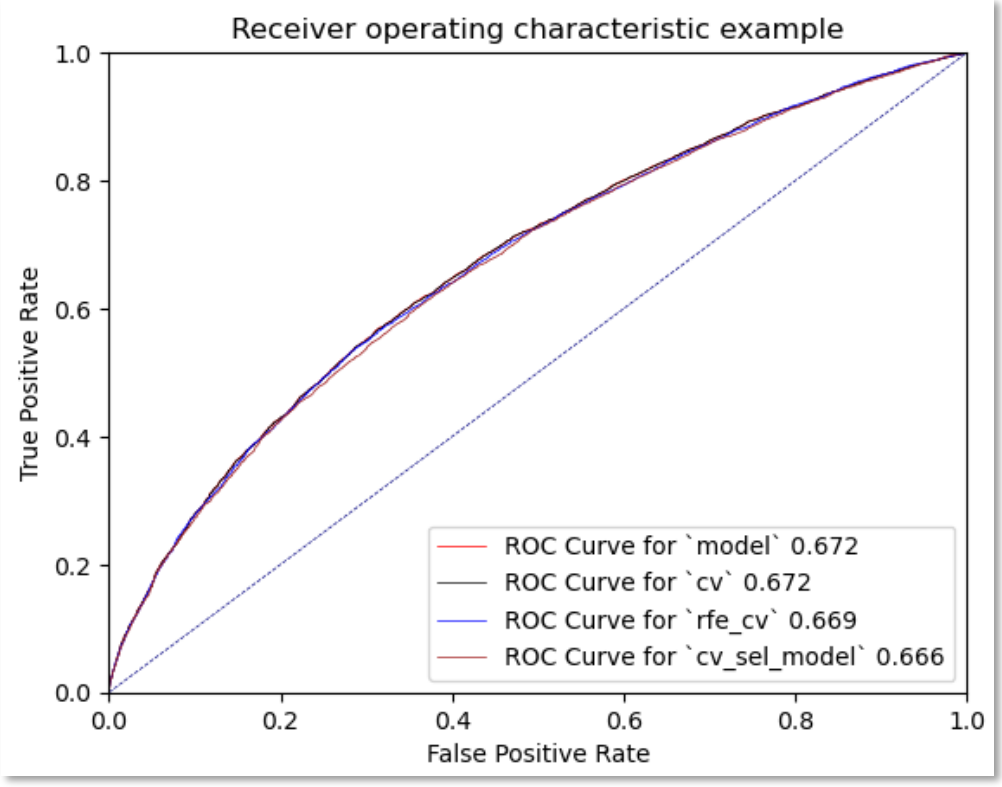
```
number_outpatient, Coefficient: 0.4908566129481026
num_medications, Coefficient: 0.26575299661816
number_inpatient, Coefficient: 0.1807099537039395
```

4. Produce the ROC curve for all different regression models. Using the best regression model, can you identify which patients could potentially be "readmitted"? Can you provide the general characteristics of those patients?

Based on our analysis, the best regression model for our case is the one with default setting showing the highest ROC index 0.672. The top 3 important variables in the model with default are number_inpatient, number_emergency, and diabetesMed in order. It indicates that a patient who has more number of inpatient, has more number of emergency, and has diabetes medication is more likely to be readmitted.

In other words, a patient who has visited hospital more as inpatient and emergency, and suffers by diabetes has the highest probability to be readmitted.

Patients have	multiple inpatient visits multiple emergency visits diabetes medications	a higher possibility of readmission



Predictive modelling using Neural Networks

1. What pre-processing was required on the dataset before neural network modelling? What distribution split between training and test datasets have you used?

1) All the pre-processing was done same as done in the decision tree.
2) Deleted rows having missing data.
3) Deleted ID variables and unary variables.
4) Applied mapping into 'age' variable.
5) Changed the data type of 'readmitted' variable.
6) Applied one-hot encoding.
7) Designated variable roles: target variable(y) = readmitted, input variable(x) = other variables.
8) Set random state. (rs = 10)
9) Set distribution split. (test_size=0.3)
10) Standardized the variables used in the model.
11) We used a 70-30 split, with 70% of the data allocated to the training dataset and 30% to the test dataset.

2. Build a Neural Network model using the default setting. Answer the following:

a. Explain the parameters used in building this model, e.g., network architecture, iterations, activation function, etc.

1) **hidden_layer_sizes:** The ith element represents the number of neurons in the ith hidden layer. We have used the default value '**100**'. (Scikit Learn, 2023)
2) **max_iter:** Maximum number of iterations. We have set the value as '**700**'.
3) **activation:** Activation function for the hidden layer. We have used the default value '**relu**', the rectified linear unit function.
4) **solver:** The solver for weight optimization. We have used the default value '**adam**', a stochastic gradient-based optimizer.
5) **random_state:** Determines random number generation for weights and bias initialization, train-test split if early stopping is used, and batch sampling when solver='sgd' or 'adam'. We have set the value as **10**.
6) **alpha:** Size of minibatches for stochastic optimizers. We have used the default value '**0.0001**'.
7) **batch_size:** Size of minibatches for stochastic optimizers. We have used the default value '**auto**'.
8) **earning_rate_init:** The initial learning rate used. It controls the step-size in updating the weights. We have used the default value '**0.001**'.
9) **momentum:** Momentum for gradient descent update. Should be between 0 and 1. We have used the default value '**0.9**'.

```
{'activation': 'relu',
 'alpha': 0.0001,
 'batch_size': 'auto',
 'beta_1': 0.9,
 'beta_2': 0.999,
 'early_stopping': False,
 'epsilon': 1e-08,
 'hidden_layer_sizes': (100,),
 'learning_rate': 'constant',
 'learning_rate_init': 0.001,
 'max_fun': 15000,
 'max_iter': 700,
 'momentum': 0.9,
 'n_iter_no_change': 10,
 'nesterovs_momentum': True,
 'power_t': 0.5,
 'random_state': 10,
 'shuffle': True,
 'solver': 'adam',
 'tol': 0.0001,
 'validation_fraction': 0.1,
 'verbose': False,
 'warm_start': False}
```

b. What is the classification accuracy on training and test datasets?

The train accuracy is around 0.728 and the test accuracy is approximately 0.596. Based on two accuracies, it appears that the model is overfitting data.

```
Train accuracy: 0.7281763910790718
Test accuracy: 0.5962549277266754
              precision    recall  f1-score   support

       False       0.61      0.68      0.64      8184
        True       0.57      0.50      0.54      7036

    accuracy                           0.60     15220
   macro avg       0.59      0.59      0.59     15220
weighted avg       0.59      0.60      0.59     15220

MLPClassifier(max_iter=700, random_state=10)
```

c. Did the training process converge and result in the best model?

No, the model is not performing well on the test data and has a substantial difference in performance between the training and test sets. The model needs further adjustments, such as regularization techniques, different model architectures, or hyperparameter tuning to improve its generalization to the test data.

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\neural_network\_multilayer_p
erceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(700) reached and the optimization hasn't converged yet.
  warnings.warn(
```

3. Refine this network by tuning it with GridSearchCV. Report the trained model.

a. Explain the parameters used in building this model, e.g., network architecture, iterations, activation function, etc.

1) 'hidden_layer_sizes': [(3,), (5,), (7,), (9,)]
2) 'alpha': [0.01,0.001, 0.0001, 0.00001]
3) max_iter: 700 and other parameters are same as previous.

b. What is the classification accuracy on training and test datasets?

The train accuracy is around 0.643 and the test accuracy is approximately 0.633.

```
Train accuracy: 0.6426560036044154
Test accuracy: 0.633508541392904
              precision    recall  f1-score   support

       False       0.64      0.74      0.69      8184
        True       0.63      0.51      0.56      7036

    accuracy                           0.63     15220
   macro avg       0.63      0.62      0.62     15220
weighted avg       0.63      0.63      0.63     15220

{'alpha': 0.001, 'hidden_layer_sizes': (5,)}
```

c. Did the training process converge and result in the best model?

Yes, it looks that the model seems to have converged to a reasonably good solution according to the learning curve. However, the train accuracy and test accuracy are not extremely high, indicating that there may still be room for improvement.

d. Do you see any sign of over-fitting?

No, we cannot find any sign of over-fitting. The train accuracy (0.643) is very close to the test accuracy (0.634), indicating that the model is generalizing well to unseen data.

4. Let us see if feature selection helps in improving the model. Build another Neural Network model with a reduced feature set. Perform dimensionality reduction by selecting variables with a decision tree (use the best decision tree model that you have built in the previous modelling task). Tune the model with GridSearchCV to find the best parameter setting. Answer the following:

a. Did feature selection favour the outcome? Any change in network architecture? What inputs are being used as the network input?

1) The model with selected features by the decision tree did not manage to improve model performance. We simply dropped it and focus on the model with RFE.
2) **Feature Selection and Reduced Input Features**: The feature selection process with RFE has significantly improved the outcome by reducing the number of input features from 111 in the previous model to just 23 in the current model.
3) **The Change in Network Architecture**: Additionally, there has been a notable change in the network architecture. The previous model employed a hidden layer with a size of 5, whereas the current model utilizes a single hidden layer with 11 neurons.
4) **The network input:** 22 selected features are used for the current model as follows.

	Selected Features
0	age
1	num_lab_procedures
2	num_procedures
3	number_outpatient
4	number_emergency
5	number_inpatient
6	number_diagnoses
7	diabetesMed
8	race_Caucasian
9	race_Hispanic
10	gender_Unknown/Invalid
11	medical_specialty_InternalMedicine
12	medical_specialty_Invalid
13	medical_specialty_ObstetricsandGynecology
14	medical_specialty_Orthopedics-Reconstructive
15	medical_specialty_Pathology
16	medical_specialty_Resident
17	medical_specialty_Surgery-Cardiovascular/Thoracic
18	medical_specialty_Surgery-Maxillofacial
19	medical_specialty_Surgery-Pediatric
20	max_glu_serum_>300
21	metformin_Steady
22	insulin_Steady

b. What is the classification accuracy on training and test datasets?

The train accuracy in this model is approximately 0.638, showing a decrease compared to the previous model and the test accuracy is 0.635, which is almost similar with the previous result.

```
Train accuracy: 0.638432079297139
Test accuracy: 0.6352825229960578
              precision    recall  f1-score   support

       False       0.64      0.75      0.69      8184
        True       0.63      0.50      0.56      7036

    accuracy                           0.64     15220
   macro avg       0.64      0.63      0.62     15220
weighted avg       0.64      0.64      0.63     15220

{'alpha': 0.01, 'hidden_layer_sizes': (11,)}
```

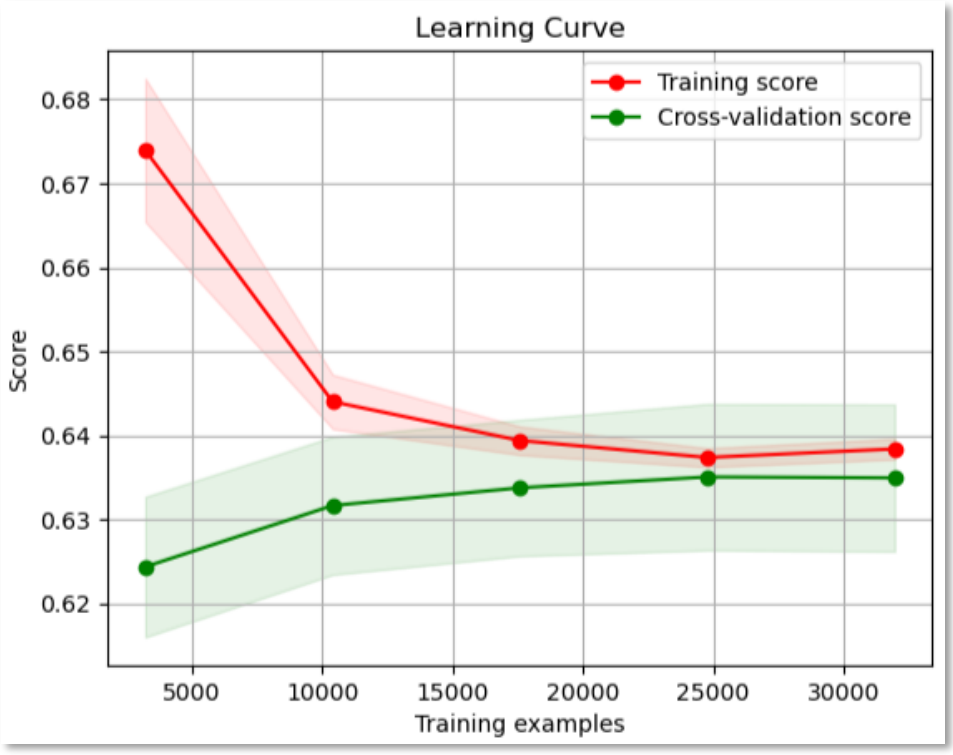
c. How many iterations are now needed to train this network?

```
iterations_needed = rfe_cv.best_estimator_.n_iter_
print("Iterations needed for training:", iterations_needed)
```

```
Iterations needed for training: 79
```

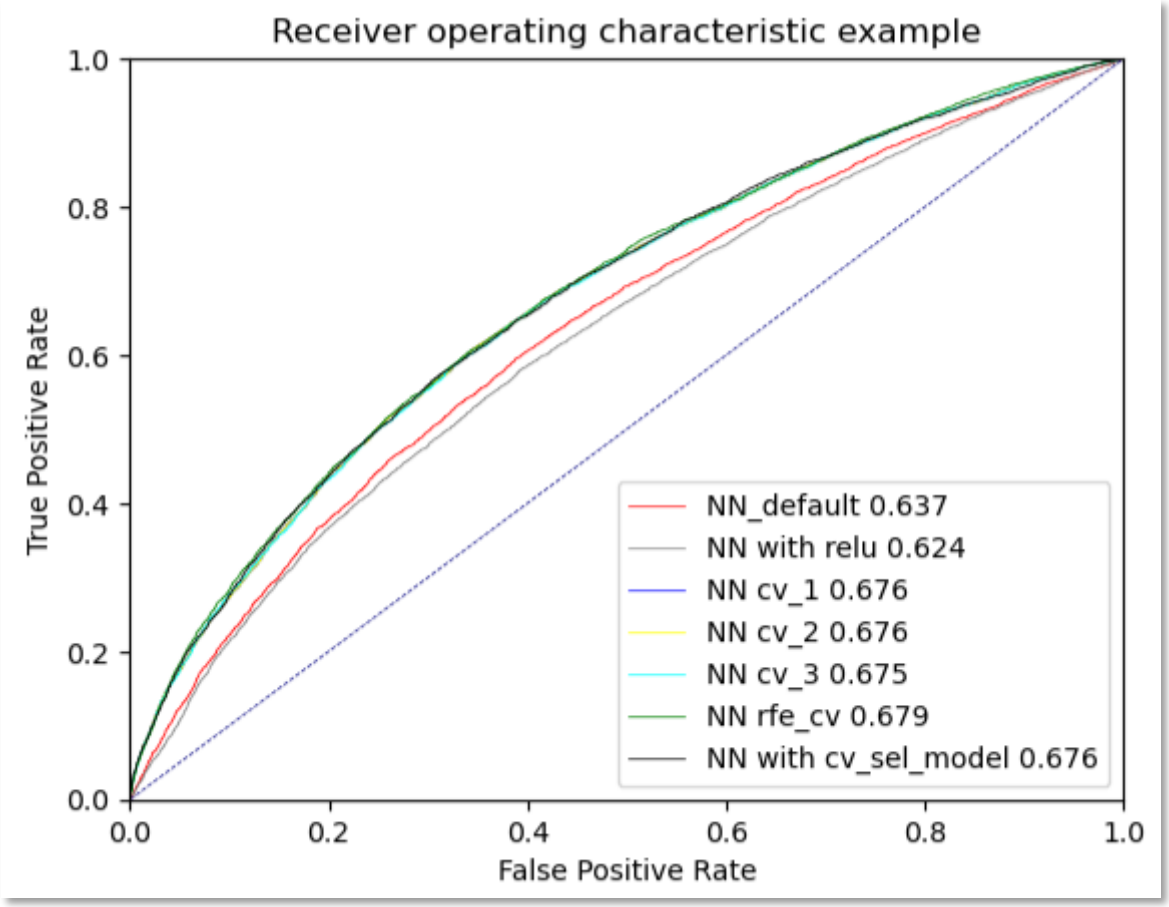
d. Do you see any sign of over-fitting? Did the training process converge and result in the best model?

No, we cannot find any sign of over-fitting. The train accuracy (0.638) is very close to the test accuracy (0.635), indicating that the model is generalizing well to unseen data. In addition, the curves started to flatten out and became parallel, it's an indication that the model has converged.



5. Produce the ROC curve for all different NNs. Now, using the best neural network model, can you provide general characteristics of the patients identified by the model? If it is difficult (or even infeasible) to comprehend, discuss why.

1) We have decided to use the NN with REF model. This model has one of the highest test accuracies, which suggests it performs well on the unseen data. Additionally, it benefits from feature selection, which reduces the number of input variables to 23, potentially improving model interpretability and generalization.



2) With the result from the NN with REF model, we have calculated the importance of each feature by summing the absolute weights across all neurons in the input layer. Based on the calculation, patients who spent longer time in hospital, are old, and have diabetes medication are likely to be readmitted with a high probability.

Patients	who stay longer in hospital who are old who take diabetes medications	a higher possibility of readmission

[Variables ordered by Impotencies]

```
time_in_hospital
age
diabetesMed
num_lab_procedures
number_inpatient
number_emergency
number_outpatient
change
number_diagnoses
num_medications
num_procedures
```

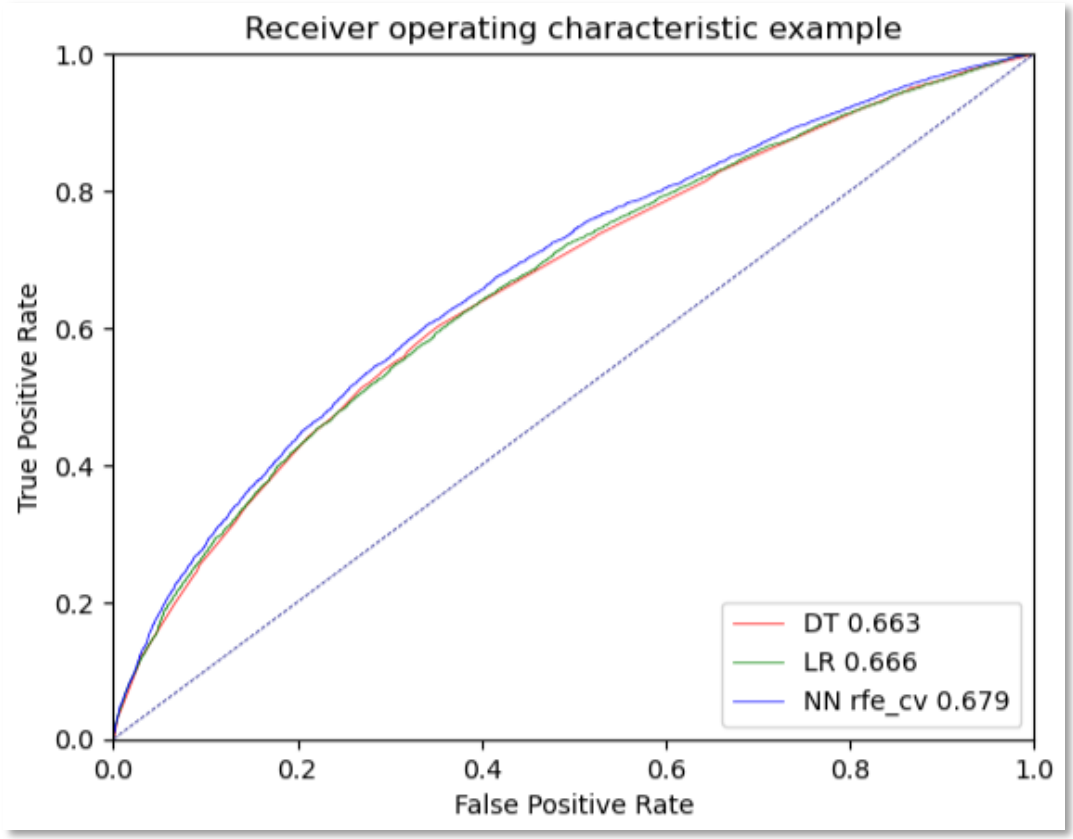
Final remarks: Decision making

1. Finally, based on all models and analysis, is there a model you will use in decision making? Justify your choice. Draw an ROC chart and Accuracy Table to support your findings.

In our case, both Decision Tree and Logistic Regression models have similar performance, with the Decision Tree having a slightly higher test accuracy. The Neural Network with RFE shows the best test accuracy among the models. However, the choice of the best model should consider more than just accuracy.

We would use the logistic regression with default model if we are the decision makers. The reasons to select the LR model are its reasonable accuracy, simplicity, and interpretability. Among the three models, the gap of test accuracy is not significant and can be ignored. The Logistic Regression model with default settings is straightforward to generate and offers ease of interpretation. For example, the important variables analysed by the LR can be simply sorted based on the absolute values of their coefficients whereas we had to calculate summing the absolute weights across all neurons in the input layer in the NN model. Moreover, our case is a binary classification whether a patient will be readmitted (1) or not readmitted (2), which is relatively simple. The Neural Network model would be not necessarily complicating. Finally, we would prioritize a simpler model that can be explained to non-technical stakeholders, such as healthcare professionals or administrators.

Model	Train Accuracy	Test Accuracy
Decision Tree tunned with GridSearchCV	0.634	0.629
Logistic Regression with default setting	0.633	0.628
Neural Network with RFE	0.638	0.635



2. Can you summarise the positives and negatives of each predictive modelling method based on this analysis?

Model	Positive	Negative
Decision Tree	1. Simple to build up 2. Easy to interpret 3. Run fast 4. Easily sort variables by using 'impotencies' feature 5. No need to standardize data 6. Can be tuned easily with GridSearchCV	1. With default setting it can be easily overfitting data 2. Test accuracy is not so high
Logistic Regression	1. Relatively simple to use 2. Easy to interpret 3. Can achieve high test accuracy 4. Suitable for binary classification 5. Easily sort important variables by using 'coef' feature	1. Data standardization is required 2. May be not the best model to analyse complex relationships
Neural Network	1. Suitable for complicating mining 2. Can achieve very high-test accuracy	1. Data standardization is required 2. Takes lots of resources for computing 3. Slow generating results 4. Test accuracy is low 5. Difficult to interpret

Reference List

Alboukadel Kassambara. (2018). Data Novia. *Determining The Optimal Number Of Clusters: 3 Must Know Methods*. https://www.datanovia.com/en/lessons/determining-the-optimal-number-of-clusters-3-must-know-methods/

Sebastian Raschka. (2014). Sebastian Raschka. *About Feature Scaling and Normalization.* https://sebastianraschka.com/Articles/2014_about_feature_scaling.html

Chioka. (2013). Chioka. *Differences between L1 and L2 as Loss Function and Regularization.* http://www.chioka.in/differences-between-l1-and-l2-as-loss-function-and-regularization/

Jason Brownlee. (2021). Machine Learning Mastery. *Gentle Introduction to the Adam Optimization Algorithm for Deep Learning*. https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

Thokagevistk, K., Millier, A., Lenert, L., Sadikhov, S., Moreno, S., & Toumi, M. (2016). Validation of disease states in schizophrenia: comparison of cluster analysis between US and European populations. Journal of market access & health policy, 4, 10.3402/jmahp.v4.30725. https://doi.org/10.3402/jmahp.v4.30725

Scikit Learn. (2023). scikit-learn.org. *Sklearn.tree.ExtraTreeClassifier*. https://scikit-learn.org/stable/modules/generated/sklearn.tree.ExtraTreeClassifier.html