**Project A – Association Mining**

**1 - Pre-processing**

The D1 dataset was loaded with pandas and inspected with the info() method. Column names were converted to lower case for convenience. Pre-processing. was limited to conversion of the Date variable to a pandas datetime object, and the Sales\_ID, and Customer\_ID to string types

A screenshot of a computer code

Description automatically generated

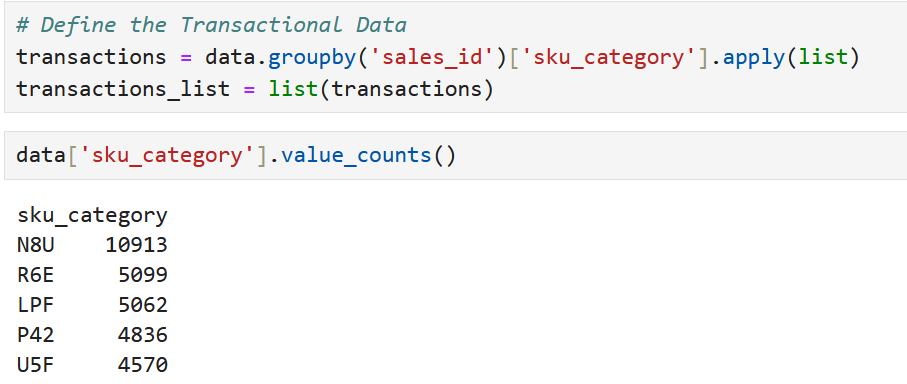
The processed dataset contains the processed variables and no missing data.

A screenshot of a computer program

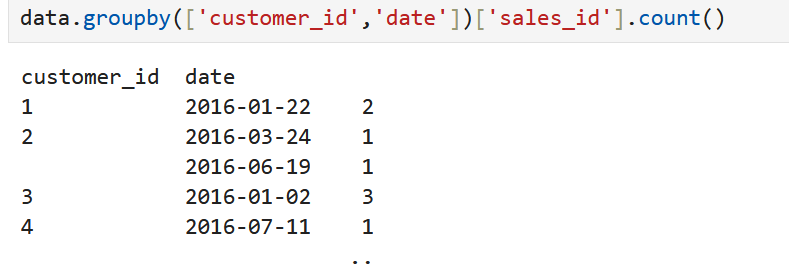
Description automatically generated

**2 - Association mining**

Association mining was carried out to determine which categories of products are bought together in a transaction rather than individually. First, the dataset was transformed to a transactional dataset using the sales\_id and sku\_category variables.



sales\_id was used because some customers had multiple sales on different dates. For example, customer 2 had two purchases, one in March and one in June 2016.



The transactions table containing all sku\_categories purchased for each sales\_id was populated and this was used to generate association rules. The apyori 's apriori function was used to generate the result set with an initial min\_support value of 0.005. This was based on the calculation of the inverse of the number of sku\_categories.

A close-up of a computer screen

Description automatically generated

The min\_confidence was initially set as 0.5 as a pragmatic approach to filter out weak associations without discarding potentially important or useful rules. The results data were converted to pandas dataframe and inspected. The initial support and confidence parameters returned a total of 12 rules. Setting the min\_confidence to a higher level was deemed unnecessary as it would have reduced the number of rules returned. We finally settled on a min\_confidence level of 0.4 as it returned a manageable set of 30 rules, and included a rule with a left value of 9.92 that was not present in the initial set because the confidence level was 0.41.

The code to generate the rules, convert to a pandas dataframe, and the resulting top 5 rules ordered by descending lift values are as follows:

A screenshot of a computer

Description automatically generated

The left-hand side and the right\_side of the association rules represents the antecedent, or the "if", and the consequent, or the "then" parts of the rules respectively.

The support is the proportion of transactions that contain both the left\_side and the right\_side. Or more simply, how frequently the rules occur in the dataset. A higher level of support would indicate that the rule is more common.

The confidence is the proportion of transactions that contain the left\_side where the right\_side also occurs. It tells us how often the rule has been found to be true. A high level of confidence reflects a higher level if reliability of the rule. Higher confidence means that the rule holds true more frequently when the antecedent occurs.

The lift shows whether the occurrence of the left\_side increases the likelihood of the right\_side. Where left is greater than 1 the left\_side and right\_side are positively correlated and that the scale to which the increase in the right\_side would be in the presence of the left\_side rule.

For our dataset, the top 5 rules show that moderately low levels of support, in the range 0.06% for {N8U,OXH} → {IEV} to 1% for {IEV,OXH} → {LPF}.

The min\_confidence was set at 0.4 in our apriori algorithm therefore the support will be at least 40%. The range in our top 5 results is 41% for {N8U,LPF} → {OXH} to 63% for {N8U,OXH} → {LPF}. These two examples are noteworthy as they contain the same items in different orders. In rule 28, OXH is likely to be bought when N8U and LPF are bought together, but it’s not as certain (41% confidence). In rule 29, LPF is much more likely to be bought when N8U and OXH are purchased together (63% confidence), making LPF a more consistently associated item in this context.

All the lift values in for our top 5 rules are 8.6 and above. This means that the presence of the right\_rule is at least 8.6 more likely in the presence of the left\_rule, rather than if the item in the right\_rule was independent. For example, {N8U,LPF} → {OXH} shows that the occurrence of N8U and LPF together increases the likelihood of an additional purchase of OXH is nearly 10 times greater.

**3 – Five common categories purchased with 01F**

To report the five most common product categories that customers bought with the product category ‘01F’ we modified our existing code filter the results such that the left\_side or the right\_side contain the sku\_category 01F. We lowered the min\_confidence parameter to 0.25 to include more rulesets. The values were ordered by support for this set to discover the rulesets with the highest proportions of rulesets where 01F occurs.

A screenshot of a computer

Description automatically generated

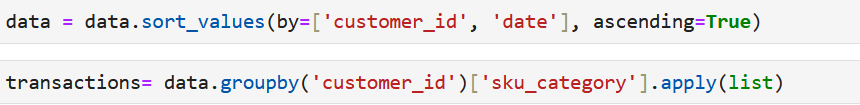
The first two rule sets are the same, but in different orders. The top 5 most common product categories purchased alongside the product category 01F are therefore IEV, LPF, OXH, FU5, and N8U.

**4 - Sequence analysis**

Sequence analysis can be carried out on this data set. The main feature of sequence analysis is that the algorithm aims to identify patterns in the order of transactions over time. In our case, the dataset was used to understand the sequence in which analyse the purchase different SKU categories over time using the date variable allowing us to identify the most frequent sequences of SKU categories that customers typically bought in a particular order.

Steps

1. The data were prepared for sequence analysis by ordering the dataset by customer\_id, then date, both in ascending order to group all transactions of each customer together, ensuring that the sequence analysis was conducted on a per-customer basis.
2. We ran the Philippe Fournier Viger SPMF java library provided in the week 8 tutorial.
3. Use the code provided in the week 8 tutorial to process the java library text outputs as association rules for the sequences into a pandas dataframe
4. The parameter of the association rules were modified for this process. Given the large number of possible rules the min\_support was set to 0.01 and min\_confidence was set to 0.5 to limit the result to a meaningful and manageable set.



A screenshot of a computer

Description automatically generated

The results of the sequence analysis allowed us to determine the likelihood that subsequent product categories would be purchased following purchase of initial product categories. For example, in our results the we showed that if a customer purchased [NTA] then the likelihood of a subsequent purchase of [IEV] is 62%, however this rule is based on 1.4% of all customer sequences meaning it is based on a small proportion of the dataset.

**5 - Relevance to decision makers**

There are several scenarios where these analyses could be actionable or insightful for decision makers. For example, the associating mining algorithm provided information about which product categories are purchased together in single transactions. Decision makers could use this information to drive sales campaigns such as product bundling or offering discounts on grouped sales. Similarly, this information could be used to inform store layouts where categories frequently purchased together are co-located in stores.

Similarly, sequence analysis could target individual customers with personalised product recommendations or targeted promotions based on potential for subsequent purchases. Another potential use case would be to forecast demand for future products based on volume of products sold that have a higher likelihood of follow-up purchases.

**Project B – Clustering diabetes data**

**1 – Pre-processing**

The diabetes dataset was loaded, and the five features of interest were selected.

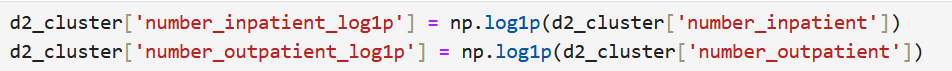


K-Means clustering is sensitive to outliers because the algorithm relies on minimising the Euclidean distances between data points and centroids, which means that outliers (data points that are far from the general distribution of the data) can disproportionately affect the position of the centroids.

The data were assessed for outliers by looking at the distributions of the variables to determine if transformations are required.

The num\_lab\_procedures and num\_medications were approximately normally distributed, so no transformation was required for these variables.

The number\_outpatient and number\_inpatient were highly skewed with many zero values, therefore were transformed using the numpy log1p approach. This transformation wass used because these variables are heavily zero-inflated and the log shifts the data by 1.



The time\_in\_hospital was only slightly skewed so transformation was not strictly necessary and therefore not used.

Similarly, the data were scaled using the StandardScaler as features with larger ranges in the data can dominate the Eucladean distance calculation leading to biased clustering results

The transformed data were assigned to a new dataframe for scaling and the scaled data were loaded back to a dataframe with the transformed column names.

A screenshot of a computer

Description automatically generated

**2 – Clustering model**

1. We chose a k-means clustering algorithm because:

* K-means is a good clustering approach for numeric data, and in our case the ease of transformation and standardisation provided an appropriate dataset.
* The data represent similar features of the diabetic patient healthcare experience therefore the K-means algorithm should attempt to group points closer together in the feature space thereby representing similar discrete groups of patients.

1. The attributes chosen are the previously transformed and scaled number of lab procedures, number of outpatient visits, number of inpatient visits, number of medications, and the time spent in the hospital.
2. The optimal number of clusters was identified using the elbow method and validated using the silhouette method as follows:

A graph with lines and dots

Description automatically generated

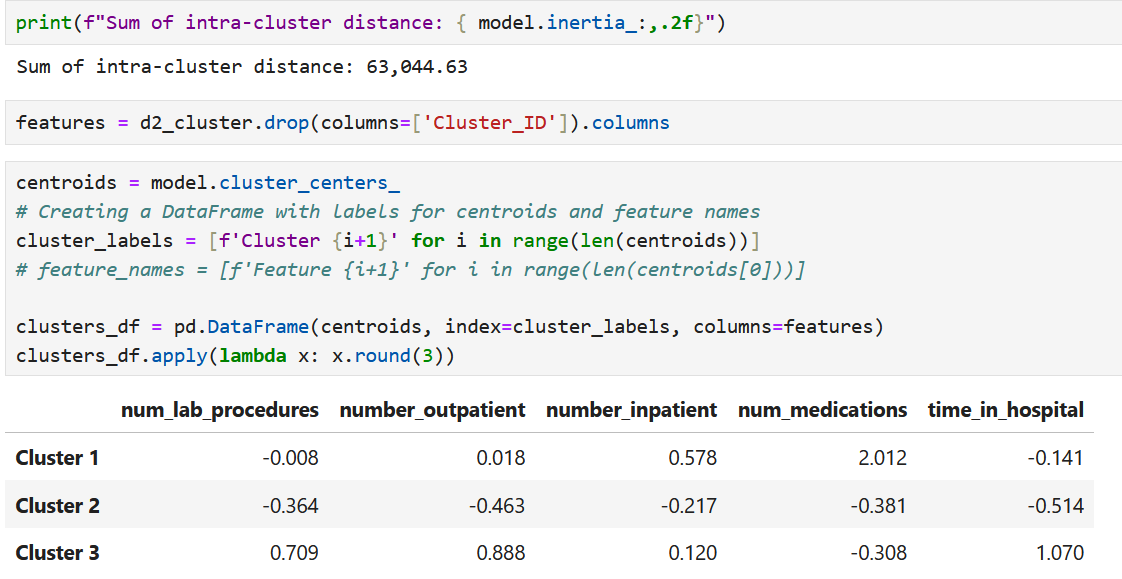
The elbow method shows that the rate of decrease of inertia slows down at a k value of 3 or 4. The optimal number of clusters was tested using the Silhouette score.

A screenshot of a computer

Description automatically generated

Based on these silhouette scores, k=3 gives the highest score indicating that 3 clusters provide the best separation between clusters in the data.

1. The K-means model from scikit-learn was instantiated and used to fit the values of the features selected from the dataset. The centroids of the fitted data were returned from the model.cluster\_centers\_ numpy ndarray. The output of this process is:



The centroid values represent the average feature values for the data points in each respective cluster. Given that our data were standardised, the positive and negative values of a centroid reflect how that cluster’s feature values compare to the overall dataset’s normalised feature values. The clusters can be interpreted as follows:

**Cluster 1**: The num\_lab\_procedures (-0.008) is close to zero indicating it does not have strong influence in this cluster, whereas the number\_inpatient (0.578) and num\_medications (2.012) are higher reflecting that this cluster tends to have higher number patients with inpatient visits and medications than the others.

**Cluster 2**: All the values in cluster 2 are reasonably strongly negative suggesting that this cluster represents patients with have fewer medications, number\_inpatient, number\_outpatient and num\_lab\_procedures that the patients in the other clusters, and the time in hospital (-0.514) suggests the patients have shorter hospital stays.

**Cluster 3**: This cluster shows relatively high values for num\_lab\_procedures, num\_medications, and time\_in\_hospital, suggesting patients in this cluster are likely to have more lab\_procedures, take more medications, and spend longer in the hospital compared to the other clusters.

1. Yes, normalisation was carried out using the standard scaler. Given that k-means clustering uses the Eucladian distance between data points, the variables with higher ranges of values biased the results. In this case, we also conducted the k-means clustering with non-normalised values of the variables and the results were biased toward num\_lab\_procedues range (1 to 118) and num\_medications (range 1 to 72) compared to the other variables which were zero-inflated.The clustering with scaled values produced a better clustering solution.

**3 – Visualisation**

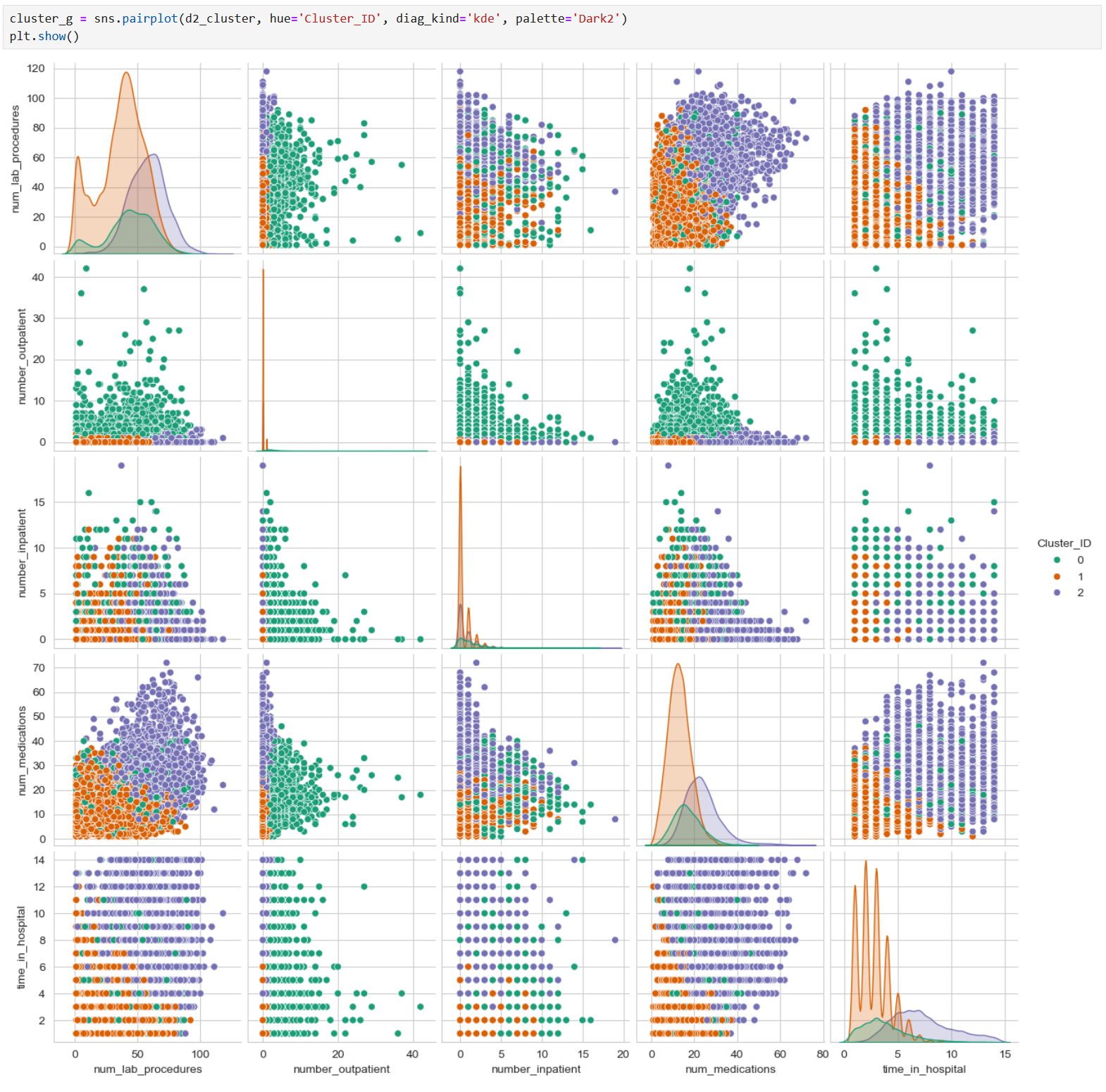
Cluster IDs were assigned to the original dataset and sns.pairplot was used to visualise the clusters with the diagonals represented by kernel density estimation (KDE). These are shown in the figure below. The results of the visualisation were interpreted in the context of the diagonal plots (KDE) and the off-diagonal plots (scatterplots).

The diagonal plots show that our **Cluster 1 (labelled cluster 0)** tends to have more concentrated peaks with lower values for num\_lab\_procedures, num\_medications, and time\_in\_hospital. Number\_inpatient and number\_outpatient are more difficult to interpret give the zero-inflated data and concentrations of the KDE plots.

**Cluster 2 (labelled cluster 1)** shows slightly broader distributions for the num\_medications feature , suggesting a wider range of values. The KDE plots for time\_in\_hospital and num\_medications are flatter, showing more variation.

**Cluster 3 (labelled cluster 2)** shows a distribution of num\_lab\_procedures and num\_medications features that are more spread out, with some patients undergoing significantly more lab procedures and taking more medications compared to the other clusters.

The results of the pairplots are consistent with the findings from the analysis and interpretation of the centroids of the clusters.



**4 – Inclusion of the Age variable**

The age variable is categorical and therefore not suited to cluster detection using a K-means clustering. For this reason we chose a K-modes algorithm which can be used where categorial data are present. K-modes uses the mode, or most frequent category, of the categorial variable to represent cluster centroids when calculating the Euclidean distance.

The results of the K-modes clustering are interesting. The inclusion of the age variable did not show differentiation for that variable in the centroid, however it did change the averages of the data of the other features in the clusters 1 and 2. The impact of age in the model has shifted the labels such that the characters of patients formerly in cluster 2 were shifted to cluster 3, and vice-versa for the former cluster to 3 to the new cluster 2. The characteristics of cluster 1 remained similar.A screenshot of a computer

Description automatically generated

Also of note, there was no difference between the modal age in each cluster. The modal age was in age group 8 for each cluster.

A computer screen shot of a computer code

Description automatically generated

**5 – Relevance for decision makers**

The K-means clustering approach is an unsupervised descriptive data-mining technique that can help uncover patterns, segment populations, and help generate insights to inform further analysis. In this respect it provides a hypothesis generating approach to support further analysis by supervised machine learning data models.

There are also potential opportunities where the findings could lead to operational benefits. For example, the analysis identified distinct groups within the data based on prior medical history and in-hospital procedures.

From our initial K-means clusters:

**Cluster 1** represents high-resource patients needing more attention (e.g., frequent hospital visits and longer stays).

**Cluster 2** represents low-resource patients, enabling more streamlined care or even preventive programs potentially driven through primary care or hospital-in-the-home.

**Cluster 3** represents patients with more demand for procedural and medication interventions. These could represent newly diagnosed diabetic patients given lower previous medical exposure, and longer time in hospital. At a more granular level, decision makers could target these patients for more tailored clinical care and allied health support such as pharmacist-led medication reviews.

**Project (c): Building and Evaluating Predictive models**

**Part 1 – Decision Trees**

**1 – Pre-processing**

In this project we are building a predictive model with the aim to classify outcomes to a target variable. The following pre-processing steps were undertaken and are collated in a separate pre-processing file called data\_prepprocess.py:

* The data were checked for missing values and two variables A1Cresult and max\_gluc\_serum were 99% and 82% missing respectively. These were dropped.
* Recoding was undertaken to limit the number of dummy variables generated from encoding the categorical variables.
* Medical speciality has 51 levels of which 72% were invalid. These levels were reduced to three levels surgical, general practice, internal medicine, emergency, and other (which included invalid).
* The admission\_type\_id, discharge\_disposition\_id, and admission\_source\_id were consolidated into fewer levels.
* One of the levels of the discharge\_disposition types was ‘deceased’. Rows of data where deceased was a level were removed because our target variable was readmitted to hospital and readmission of deceased patients does not make logical sense. It would have likely introduced noise in the data. 320 deceased patient records were dropped.
* The drug names, and age columns were recoded into numerical variables.
* Dummy variables were created for race, medical\_specialty, admission\_type, discharge\_disposition, and admission\_source.

For the test train split two factors were considered for the test/train split, size of dataset and balance of the target variable. Given the data size was 20,000 records and the readmitted variable id well balanced (47% readmitted, 53% not readmitted), a standard 80:20 split was used for training:test data.

**2 – Decision tree model with default settings**

The decision tree model was built with the processed data, using default settings as follows:

A screenshot of a computer

Description automatically generated

1. **Accuracy of training and test data sets**

The reported training and test accuracy were:

A screen shot of a computer code

Description automatically generated

1. **Number of nodes and leaves**

A computer screen shot of a code

Description automatically generated

1. **The variable for the first split**

A screenshot of a computer code

Description automatically generated

The first split was made on number\_inpatient. This step is a critical step in the prediction modelling because it is made at the root node, and it is based on the feature that provides the highest purity of the split (the highest information gain) across the dataset. This first split divides the data into two groups that are as uniform as possible concerning the target variable.

1. **The first 5 features by importance were retuned using the feature\_importance\_ attribute from the model**

A screenshot of a computer

Description automatically generated

These were num\_lab\_procedures, num\_medications, time\_in\_hosptal, number\_inpatient and age.

1. **Parameters were used to build the tree**

A screenshot of a computer program

Description automatically generated

The parameters used to build the tree were reported using the model get\_params() method. The default settings were:

* **ccp\_alpha**: this is a pruning parameter that is used to control model complexity. The default setting is ‘off’ with no pruning.
* **class\_weight:** controls the weight allocated to each class. The default setting is none meaning all classes have an equal weight of 1.
* **criterion**: specifies the method used to make splits at each node based on the distribution of the target variable.
* **max\_depth**: is used to limit the depth of the decision tree. The default setting of ‘none’ means that the model will continue to run until all leaves are pure.
* **max\_features:** specifies the number of features to consider when looking for the best split and is used to limit the number of features at each split.
* **max\_leaf\_nodes:** grows the decision tree with a maximimum number of nodes using best first, meaning ordered by relative reduction in impurity per node.
* **max\_impurity\_descrease:** each node is split in a way that decreases the impurity to at least this value.
* **min\_samples\_leaf:** The minimum number of samples required to be at a leaf node.
* **min\_samples\_split**: specifies the number of samples that must be used to split an internal node.
* **min\_weight\_fraction\_leaf:** The minimum weighted fraction of the sum of the total weights (of all the input samples) required to be at a leaf node. Samples have equal weight when sample\_weight is not provided.
* **monotonic\_cst:** controls the monotonicity constrains to either increasing, none, or decreasing. For example, setting this to increasing would mean that as feature values increases, the predicted target value should not decrease.
* **random\_state:** controls the randomness of the estimator such that random\_state fixed to an integer results in deterministic behaviour (used for reproducibility). Our was set to 1.
* **splitter**: specifies the split strategy, the options are best or random.

Source: <https://scikit-learn.org/dev/modules/generated/sklearn.tree.DecisionTreeClassifier.html>

**3 –Decision tree model tuned with GridSearchCV**

The GridSeachCV was used to automate hyperparameter tuning by searching for the optimal combination of hyperparameters that allow the algorithm to generalise well to unseen data. We performed this regularisation step using different combination of the criterion, max\_depth, and min\_samples\_leaf hyperparameters to attempt to reduce the complexity of the model.

A screenshot of a computer

Description automatically generated

Criterion parameters of gini and entropy were tested. These parameters control model purity and information gain respectively. Purity is used to maximise separation of classes and optimise allocation of classes into single nodes. Information gain measures how much information is gathered that relates to the target class.

The max\_depth hyperparameter control the number of levels a decision tree can have from the root to the farthest leaf. Shallow trees (low max\_depth) tend to be less complex and less likely to overfit, however if trees are too shallow the tendency is to underfit.

The min\_samples\_leaf parameter specifies the number of samples required for each node. This ensures that leaves with too few samples are not generated and limits the potential for outliers to generate leaves.

We plotted the mean\_train\_scores and mean\_test\_scores (accuracy) for each iteration of max\_depth (averaged across 10 cross-validation splits) as follows:

A screenshot of a computer code

Description automatically generated

The resulting graph show that the optimal max\_depth is 4.

A graph with a line and a red line

Description automatically generated

This is consistent with the best\_params output from the cross-validated model.

A screenshot of a computer

Description automatically generated

We used these best\_params\_ attributes to specify a best\_ model:

A screenshot of a computer code

Description automatically generated

1. **Classification accuracy of training and test datasets in the revised model**

A screenshot of a computer code

Description automatically generated

The training accuracy of 63.2% indicates that the model fits well to the training data. While not perfect, it indicates that the model is learning from the training data without memorising it and overfitting.

The test accuracy of 61.8% indicates how well the revised model has generalised to unseen test data. The test accuracy is slightly lower than the training accuracy (63.2% vs. 61.8%) suggesting that the model has a good balance between fitting the training data and generalising to the test data.

1. **Tree size (leaves and nodes)**

A screenshot of a computer code

Description automatically generated

The revised model has 31 nodes and 16 leaves and is therefore much smaller and compact compared to the default model. This is because the grid search cross validation has regularised the model to limit tree depth and the minimum samples per lead to reduce the model complexity. This compact size indicates that the model has focused on the most relevant splits without trying to capture all the details of the training data.

This is in comparison to the default model which had 9017 nodes and 4509 leaves and was extremely large and complex. The larger tree size meant that the default model grew until it could no longer split, thereby introducing noise and small variations in the training data.

1. **Variable for first split**

A screen shot of a computer code

Description automatically generated

As with the default model, the revised model first split was on number\_inpatent. This indicates that the number\_inpatient has an important role in separating the data and predicting the target variable.

1. **Five most important variables**

1. number\_inpatient

2. num\_medications

3. number\_emergency

4. age

5. number\_outpatient

In the tuned model the number\_inpatient is the most authoritative variable is predicting the target variable. This is markedly different to the untuned model. There are commonalities in terms of the strongest predictive features such as, in addition to number\_inpatient, mum\_medications, and age, however, the importance of the features in the tuned model is much higher than the default. Number\_emergency and number\_outpatient were not detected in the default model.

The importance of features in the revised model most likely can be attributed to the reduced complexity of the tree (fewer splits). The algorithm therefore prioritised a smaller set of strongest features. As a result, features like number\_inpatient might play a more central role in defining the early splits, which simplifies the model to reduce overfitting under regularised conditions.

1. **Overfitting assessment**

The revised model produced similar training and test scores (63.2% and 61.8% respectively). This is a good indicator of a model that has found a reasonable trade-off between bias and variance, avoiding both overfitting and underfitting. The model is learning useful patterns that generalise to new data.

The smaller size of the tree in the revised model (31 nodes and 16 leaves) suggested that the tree was pruned and regularised, preventing it from growing in complexity and overfitting the training data.

**4 – ROC curves and model differences**

The following table summarises the key differences between the models:

|  |  |  |
| --- | --- | --- |
| Area of difference | Default model | Tuned model (best) |
| Complexity | Number of nodes: 9017  Number of leaves: 4509  Large and complex | Number of nodes: 31  Number of leaves: 16  Smaller and constrained |
| Accuracy | Training accuracy: 100%  Test accuracy: 56.5%  Perfect training accuracy indicating overfitting. Test accuracy is much lower than training accuracy, meaning the model does not perform well on unseen data. | Training Accuracy: 63.2%  Test Accuracy: 61.8%  Scores are closer to each other, indicating a better balance between fitting the training data and generalising to unseen data. |
| First split feature | number\_inpatient | number\_inpatient |
| Feature importance | 1. num\_lab\_procedures 2. num\_medications 3. time\_in\_hospital 4. number\_inpatient 5. age   Overfitting likely identified many features with high importance | 1. number\_inpatient 2. num\_medications 3. number\_emergency 4. age 5. number\_outpatient   Similar features however regularisation likely ensured a more meaningful contribution of features to the splits |

The Receiver Operating Characteristic (ROC) curve is a graphical representation used to evaluate the performance the model. It plots the trade-off between the True Positive Rate (TPR) and the False Positive Rate (FPR) for different classification thresholds. We created ROC curves for test data because the test set gives an unbiased estimate of the model's true performance. The training data performance would be of limited value in this case as the model has already memorised the data and has learned specific patterns from it.

The following important characteristics are calculated to produce the curves for each model:

* **y\_proba:** the predicted probabilities that a model assigns to each sample belonging to the positive class
* **fpr (false positive rate):** The proportion of negative instances that are incorrectly classified as positive.
* **tpr (true positive rate):** The proportion of positive instances that are correctly classified as positive
* **thresholds:** values between 0 and 1 that determine how the predicted probabilities are converted into the class predictions (0 or 1).
* **roc\_curve:** a function which computes the fpr and tpr at different thresholds by comparing the actual target values to predicted probabilities
* **roc\_auc\_score**

A screenshot of a computer program

Description automatically generated

A graph of a line

Description automatically generated with medium confidence

The general characteristics of the predicted readmissions were visualised in bar plots. The identify the readmissions, best\_model was instantiated to produce an array of data with labelled readmissions. This was converted to a dataframe, and filtered to only include readmissions. Only the top 5 important features were used.

A screenshot of a computer

Description automatically generated

The resulting dataframe was used to group and describe the features of readmitted patients.

A group of different colored bars

Description automatically generated

The characteristics of readmitted patients can be summarised as follows:

* **Previous inpatient encounters in the year prior to admission:** Predominantly (over 80%) of patients had up to 2 prior inpatient encounters.
* **Number of medications during admission:** The majority of these patients has between 11 and 20 medications administered during their admission.
* **Number of emergency encounters in the year prior to admission:** Similar to inpatient encounters, around 90% had few prior encounters (up to 2)
* **Age group of readmitted patients**: The dominant age groups were the three deciles between 60 and 80 years of age (approx. 70%), representing an older age group.
* **Number of outpatient encounters in the year prior to admission**: as for inpatient and emergency encounters, the majority of prior encounters was up to 2 (approximately 80%).

**Part 2 - Predictive modelling using Regression**

**1 – Pre-processing**

The same initial pre-processing steps that were used in data\_preprocess.py were employed for regression modelling. This included:

* Dropped the variables A1Cresult and max\_gluc\_serum as they were 99% and 82% missing respectively.
* Recoding was undertaken to limit the number of dummy variables generated from encoding the categorical variables. Medical\_speciality, admission\_type\_id, discharge\_disposition\_id, and admission\_source\_id were all recoded as per the decision tree pre-processing.
* Rows of data where deceased was a level in discharge\_disposition\_id were removed because our target variable was readmitted to hospital and readmission of deceased patients does not make logical sense.
* The drug names, and age columns were recoded into numerical variables.
* Dummy variables were created for race, medical\_specialty, admission\_type, discharge\_disposition, and admission\_source.

For the regression modelling, variables were split into numerical and categorical types:

A screenshot of a computer

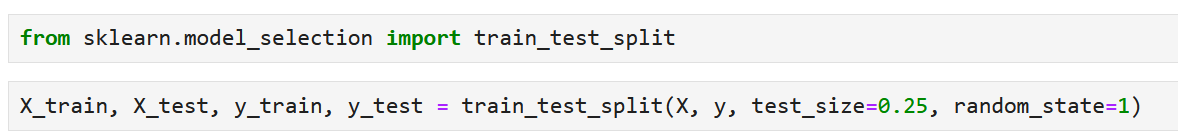
Description automatically generated

Categorical variables were one-hot encoded and age was label encoded. Numerical variables were scaled with standard scaler.

A screenshot of a computer code

Description automatically generated

For the train/test split the same 75/25% split for training/test was used because the data set is reasonably large (20,000 records) and the target variable is balanced. Again, the 25% test set gives more data to evaluate the model.



**2 – Build a regression model and tune with GridSearchCV**

Build the default model

A screenshot of a computer program

Description automatically generated

Perform cross validation with GridSearchCV

**A screenshot of a computer

Description automatically generated**

1. A logistic regression model was used because it is a simple and efficient algorithm for binary classification problems for categorial outcomes such as ‘readmitted’. It performs well on linearly separable data and provides interpretable results through its coefficients. This model choce balances model simplicity and performance.
2. The regression function used is the logit function. This models the relationship between the data features and the probability of an outcome in the class 0 or 1. The logit function predicts the probability that an observed feature belongs to a the 0 or 1 class using the logo-odds transformation.
3. Yes, standardisation was used for the logistic regression model to improve convergence of the gradient decent. Differences in scales can lead to failure of the model to converge. Also, since logistic regression describes the effect of unitary change in feature variables on the log-odds of the target, the results are easier to interpret with standardised features.
4. Variables used

We used all the variables in the model that we

1. The top 5 most important variables:

A screenshot of a computer program

Description automatically generated