Data Mining the Diabetes Mellitus Disease

CMP-7023B Data Mining

Student ID: 100249635

ABSTRACT

This project demonstrates the use of supervised and unsupervised learning algorithms on ICU patient data, attempting to classify patients as being positive or negative for *diabetes_mellitus*. The project encompasses the KDD process, putting data mining and machine learning principles into practice for the stages of data identification, cleaning, preprocessing and model training. The trained supervised model was able to detect *diabetes_mellitus* with 71% accuracy, providing insight into the determinants of the condition and which features are more indivative of a positive result.

1 INTRODUCTION

This project aims to use KDD and Data Mining methods to classify the Diabetes Mellitus status of ICU patients using medical testing data generated within 24 hours of their admission. Patients admitted to ICUs do not always have accurate or verified medical records, making identification and diagnosis of chronic illness vital to determining patient care strategies. Machine learning, alongside KDD techniques, will be used to train models to recognise and predict Diabetes Mellitus from the limited data available, with the aim of delivering an acceptable level of accuracy using both supervised and unsupervised methods.

2 DATA AND RESOURCES

The data used in this project consists of a largely incomplete dataset of medical test results from a range of ICU patients. The incomplete nature of this data requires a cleaning and pre-processing stage to facilitate the use of supervised machine learning algorithms. Accompanying the dataset is a data dictionary containing detailed descriptions of the features within the main dataset and their corresponding data types.

Tasks for the project will be completed in the Python environment using the Anaconda distribution (version 3.8.8). Pre-processing tools, supervised, and unsupervised models used in the project are sourced from the Scikit Learn [1] library.

3 METHODOLOGY

The project will be undertaken in four stages:

- Data Identification and Summarising
- Cleaning and Pre-processing
- Supervised Model Selection and Training
- Unsupervised Modelling

Each stage relates to the corresponding Python file found in the appendix of this report.

3.1 Data Identification and Summarising

The dataset in use contains 88 features (including the target features *diabetes_mellitus*) and 79159 records (see Appendix A). Feature types include continuous features, binary features, and categorical features, alongside multiple data types (see Appendix B). Upon inspection, some features contain large amounts of missing data or null values.

The features are divided into numeric and categorical partitions and investigated further (see Appendix C). Feature 'bmi' was transferred to numeric as it is a continuous variable. The majority of the features in the dataset are continuous numeric (weight, age, glucose level), with other features being binary (patient has disease or does not) or categorical (ethnicity).

Once partitioned, descriptive statistics are calculated on both sets of features (see Appendix D and F).[1] Descriptive statistics include the number of missing values and percentage of missing values. For numeric features, a separate table of features with more than 60% of values missing is included (see Appendix E). These features will be removed during the cleaning process. Also included in descriptive statistics is the number of unique values in each feature, as this will assist in identifying constants and quasi-constants.

3.2 Cleaning and Pre-processing

The dataset will undergo cleaning and pre-processing before being subject to supervised learning algorithms. An effective cleaning and pre-processing stage is considered imperative to gain optimal results from machine learning applications, resulting in being able to make accurate inferences from the results [2].

The first task in the cleaning and pre-processing stage is to remove features that either contain all null values or are not relevant to training the supervised models. Features *readmission_status, encounter_id, hospital_id, icu_type* have been removed for this purpose respectively. These features do not provide any meaningful insight into the patient's medical condition, only information pertaining to the ICU they are in. Features previously identified as having 60% or greater missing values are also removed at this stage, and lists of numeric and categorical features updated.

CMP-7023B Data Mining, ,
2022.

[1] Tables should be in landscape format but Latex code not working for unknown reasons. Tables attached as CSV files in .zip folder. Apologies.

To check for imbalance in the remaining numeric features, Kernel Denstity Estimate (KDE) plots are generated to visualise the features' probability distributions. The resulting plots confirm that all remaining numeric features have unimodal distributions, and that both positive and negative classifications appear to to be distributed around a similar mean [3]. Figure 1 shows four KDE plots of numeric features generated with Python *seaborn* library.

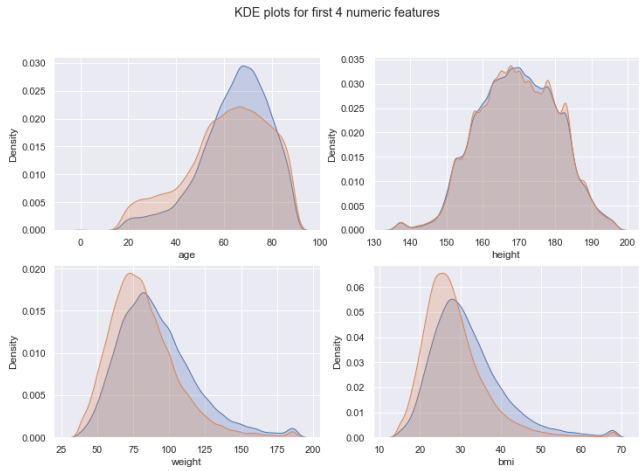

Figure 1: KDE plots for first four numeric features

To check for imbalance in categorical features, countplots have been generated to visualise the frequency of each variable in each category. Countplots can assist in identifying any features that contain highly imbalanced data, such as a binary feature containing >99% '1' values. Features with such imbalance will not contribute meaningfully to the model and will be removed. Figure 2 shows four countplots of categorical features generated with Python *seaborn* library.

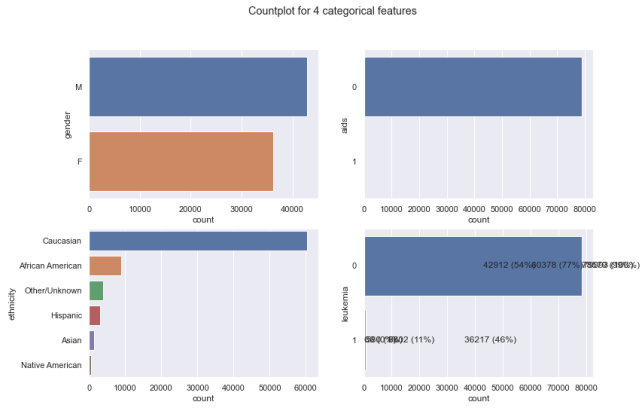

Figure 2: Countplots for four categorical features

The countplots have identified some features that may be constant or quasi-constant. No fully constant (100%) features were identified, however three features were identified when tuned to 99% sensitivity. Features *aids, leukemia* and *lymphoma* contain <1% positive (1) values, so will also be removed from the model using the *fastML* library's *get_constant_features* utility. Justification for removing these features is that patients with these conditions are so uncommon within the data that they may not provide any informative benefit when training a supervised model.

Features that are not useful due to large amounts of missing data have been removed, however remaining features are not complete. Imputation can be used to generate the missing feature values based on its other values to enable models to be trained with greater statistical efficacy and reduce the occurrence of biased estimators [4]. Many approaches to imputation exist, this project will use a simple approach of imputing missing values with a mean of the whole feature's values. Missing values are imputed using *SimpleImputer* from *sklearn*, using mean values for numeric features and most frequent values for categorical features.

Following imputation, the dataset is checked for features that are highly correlated, as these variables will not have any explanatory power and are effectively redundant [3]. To determine correlation, all remaining features are correlated into a heat-map matrix using *pandas*. Figure 3 shows

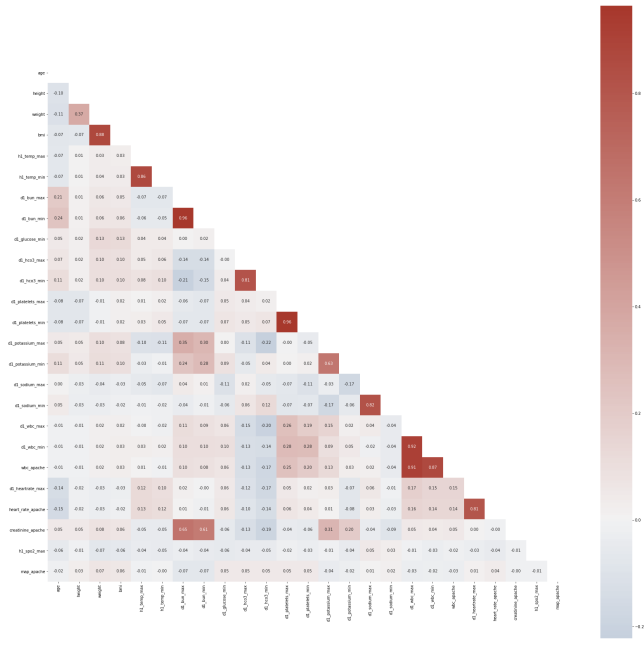

Figure 3: Correlation heat-map matrix of features

the correlation heat-map matrix. Most features appear to

have insignificant degrees of correlation (<90%) and none are perfectly correlated. To ensure that no features are quasi-redundant due to high levels of correlation, any features correlating at >90% will be removed. Upon unstacking and sorting the matrix by degree of correlation, features *d1_bun_max*, *d1_platelets_max*, and *d1_wbc_max* are highly correlated. The three features all relate to properties in the patient's blood, being maximum blood urea, maximum platelet count, and maximum white blood cell count in their first 24 hours in the ICU.

Following the removal of features not conducive to model training and imputation of missing values, the next task is outlier detection. Outliers, being data points that are largely anomalous from the population of the data, may be formed via unexpected circumstances and therefore contain high levels of explanatory power when modelling [5]. The outlier detection method chosen for this project was *IsolationForest* from *sklearn*. This method was chosen for its efficacy in handling high volume datasets and for its efficiency and low memory requirements, making it an ideal algorithm for use on lower performance systems [6]. *IsolationForest* was deployed on the dataset with the maximum samples parameter set to the total data records and the contamination set at 0.01, which yielded 792 outliers. These outliers were not removed from the data as they may contain some explanatory power. Figure 4 shows the results of the *IsolationForest* outlier detection.

Figure 4: Scatter plot showing outliers generated by *IsolationForest*

The final task in cleaning and pre-processing is encoding. Categorical features in the dataset must be encoded to be interpretable by supervised learning models. Firstly, the target feature *diabetes_mellitus* is removed from the rest of the dataset as this does not require encoding. The encoder selected to encode categorical features is *OneHotEncoder* from *sklearn*. Binary features become encoded as paired opposing arrays (where one array is '0', the other '1'). Categorical features, such as ethnicity, are encoded in binary arrays for each category.

Feature name	Mutual info
d1_glucose_min	0.03433563
bmi	0.022279312
d1_bun_min	0.019708835
creatinine_apache	0.0168251
weight	0.015124069
age	0.011774696
cirrhosis_0	0.010265501
elective_surgery_0	0.008386778
gender_M	0.007776967
solid_tumor_with_metastasis_0	0.007470873

Table 1: k-10 best features as generated by *SelectKBest*

3.3 Supervised Model Selection and Training

Once cleaning and pre-processing of the dataset is complete, the data must be split into partitions for training and testing. The data in this project was split 80/20 between train and test, and includes a random state to ensure that results are reproducible.

Following the train/test split, features to use to train the supervised models are selected. This project will use *SelectKBest* from *sklearn* to select the most informative features based on mutual information calculated from dependencies between the features. Two training sets will be generated, one with the k-10 best features only and another with all of the top features nominated by *SelectKBest*. Table 1 shows the k-10 best features, ranked by mutual information scores. The results appear to be broadly consistent with the determinants of *diabetes_mellitus*, being a lack of insulin affecting the body's ability to regulate glucose in the bloodstream and the differing effects of weight, height, and age on the body's ability to regulate glucose [7].

Before selecting supervised learning models, the target variable *diabetes_mellitus* needs to be resampled. Presently, the value counts on the target training data are heavily imbalanced towards '0' (negative *diabetes_mellitus* by 40850 to 22477. Using *resample* from *sklearn*, the target variable '1' is up-sampled to match the current '0' value count of 40850. Figure 5 shows the results of up-sampleing the target variable. Up-sampling is applied to both the training sets including all features selected and the k-10 best features selected.

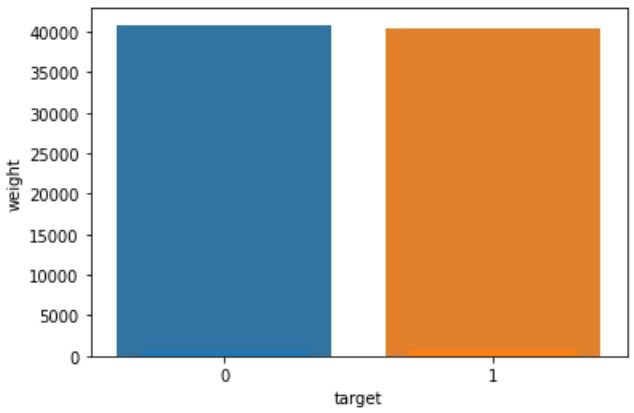

Figure 5: Countplot demonstrating effect of up-sampling target variable

Three supervised learning models are selected for the training data: *RandomForestClassifier, SVC*, and *Gaussian Naive Bayes*. Primary justification for selecting these models is their suitability towards binary classification problems, however each model contains distinct benefits in its own right. *RandomForestClassifier* is considered to be highly competent for classification using numeric values, which the dataset is fully comprised of [8]. *SVC* performs well when modelling feature-rich datasets [9]. Finally, *Gaussian Naive Bayes* has a high success rate in classifying other chronic illnesses, such as heart disease [10].

A Pipeline with a transformation and classification stage will be used to deploy the models on the training data. Pipelines are adequate for this task because it combines the transformation and classification operations in a separate environment for each model, ensuring that the models are trained in isolation and do not influence each other. Seperate Pipelines will also be generated for the models being trained with the k-10 best features, separated from the models training using k-all features.

The transformer selected for the unsupervised models is *StandardScaler* from *sklearn*. *StandardScaler* converts the data within the feature to a Gaussian distribution, with standardisation being imperative for unsupervised models to function as unexpected results may occur if data is alternately distributed.

Both sets of training data are fitted to the pipelines and subsequently scored with accuracy, balanced accuracy, and F1 scores. Table 2 shows the best performing classifier for accuracy in both sets of training data is *RandomForestClassifier*. *SVC* outperforms *Gaussian Naive Bayes* for k-all classifiers, but the inverse is true for k-10 features. Finally, it

		k-all Features			k-10 Fe
	Classifier	Accuracy	Balanced Acc	F1	Accura
0	RandomForestClassifier	0.71	0.64	0.50	0.68
1	SVC	0.68	0.66	0.58	0.66
2	GaussianNB	0.65	0.61	0.50	0.67

Table 2: Accuracy scores for k-all and k-10 classifiers training data

appears models are more efficacious when using training data containing k-all features than k-10 features. Evaluation and discussion of these results is located in the 'Evaluation' section.

Further to accuracy scores, precision and recall and ROC plots have been made for the three models. Figure 6 shows the plots.

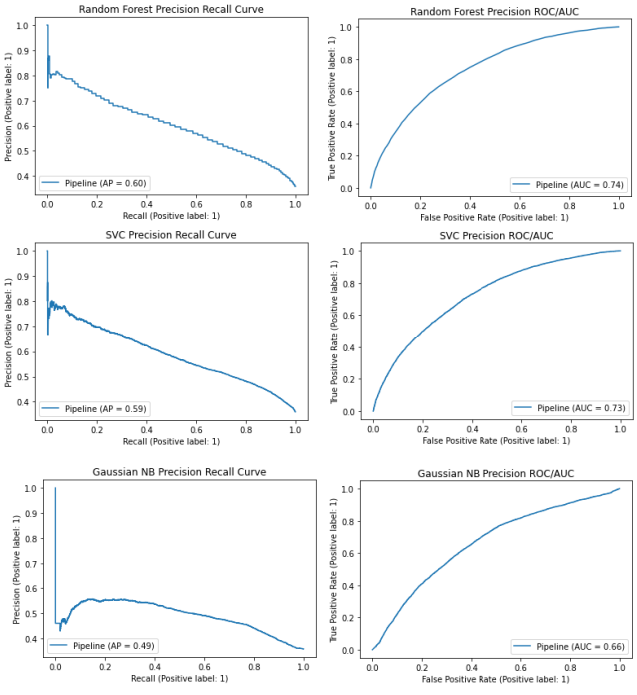

Figure 6: Precision/recall and ROC plots for models

RandomForestClassifier appears to be the best performing model, so this model will undergo hyperparameter tuning to attempt to increase the performance further. Hyperparameter tuning makes use of cross validation from *GridSearchCV* in *sklearn*, in which the best combination of model parameters will be determined after iterating through all possible

combinations. For the *RandomForestClassifier*, hyperparameters to tune are 'criterion' and 'n_estimators', with optimal estimators considered to be in the range of 20-100 for a dataset of this size [11]. Precision and recall and ROC plots show that hyperparameter tuning has made a minute difference to the model's performance. Figure 7 shows the precision and recall and ROC plots for the tuned model.

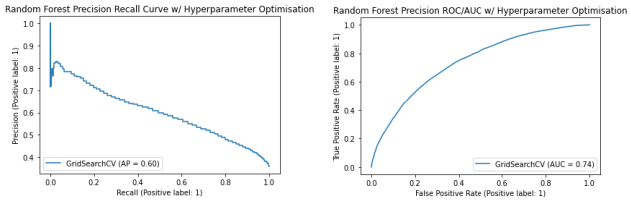

Figure 7: Precision/recall and ROC plots for tuned model

3.4 Unsupervised Modelling

Finally, the dataset will be subject to unsupervised learning algorithms. Clustering using *KMeans* from *sklearn* will be used to classify the dataset between '1' - has *diabetes_mellitus* and '0' - does not have *diabetes_mellitus*. *KMeans* clustering is a popular method for unsupervised modelling, as it is a simple and effective algorithm [12]. Before undertaking clustering, the up-sampled data will be reduced to 30% of its original size to simplify the process and make clusters easier to identify.

Firstly, the optimal number of clusters to feed into the algorithm is determined by the 'Elbow method' and silhouette score. The combination of these two methods is an accurate and discerning approach to discovering clusters [13]. Figure 8 shows the Elbow plot and silhouette score for optimum number of clusters, which is 2.

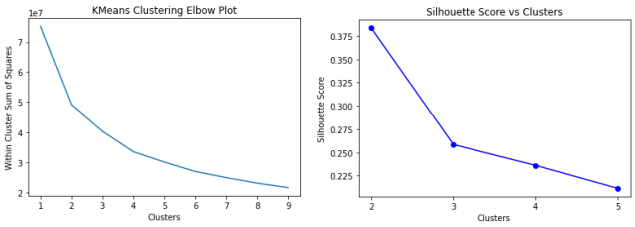

Figure 8: Elbow plot and silhouette score for KMeans clusters

When fitted to the test data, Figure 3.4 *KMeans* algorithm forms two distinct clusters when features 'd1_glucose_min' and 'weight' are used. This is perhaps indicative of a threshold in the minimum amount of glucose in the blood over the first 24hours in the ICU, which appears to reduce slightly as weight increases.

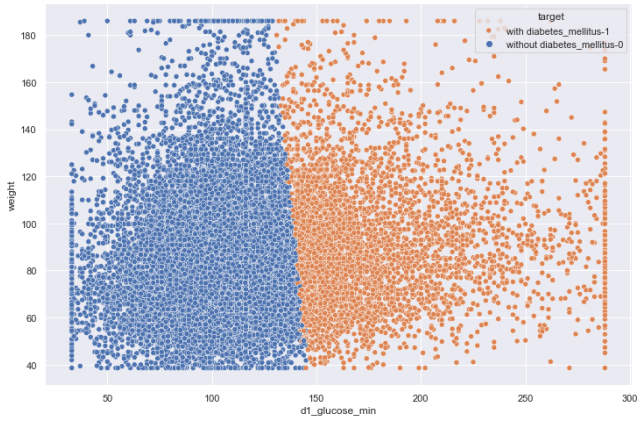

Figure 9: *KMeans* clusters for features 'd1_glucose_min' and 'weight'

Principal Component Analysis is a potent tool that can be used to verify the accuracy of the *KMeans* clusters by reducing the dimensionality of the data, drastically decreasing the size and complexity of the dataset whilst retaining the interdependencies required to have an explanatory effect [14]. The test data is first normalised before being reduced to Principal Components in a two column array. Figure 3.4 shows the resulting clusters, with one dense cluster of '0' - without diabetes_mellitus and no discernible cluster for '1' - with diabetes_mellitus.

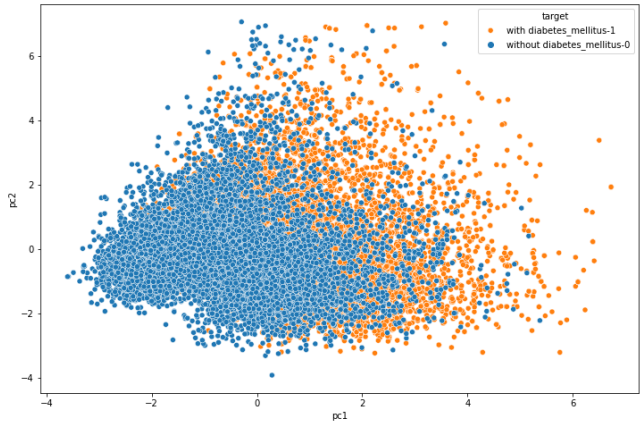

Figure 10: *KMeans* clusters following dimensionality reduction by Principal Component Analysis

Rand and mutual information scores were undertaken on the *KMeans* clustering using Principal Component Analysis,

however both yielded results of 1.0. This possibly indicates an error as the *KMeans* predictions for the target variable exactly match the actual target variable, implying that the unsupervised model is 100% accurate which is disproved by the visualisation of the clusters.

4 EVALUATION AND CONCLUSION

The supervised learning algorithms yielded adequate scores for classifying *diabetes_mellitus* with an accuracy score of 0.71 for *RandomForestClassifier*. This metric is based off the count of predictions that turn out to be false, so the trained model is currently returning erroneous results at 29%. This score could be improved by a more intuitive method of imputation, using KMeans imputation or deep learning methods to impute missing values, rather than using mean values which can be far removed from the actual probability distribution of the feature [4].

The precision and recall and ROC plots for *RandomForestClassifier* did not seem to improve by a discernible amount following hyperparameter tuning. This may be caused by not tuning enough of the hyperparameters during this stage, as only two ('criterion' and 'n_estimators') were selected. Logically, the more hyperparameters that are tuned should result in a more sensitive trained model, resulting in more accurate classification on test data.

The unsupervised learning algorithm *KMeans* yielded an erroneous Rand and mutual information score of 1.0 following Principal Component Analysis. It is possible that an error in the code caused the testing target variable to equal the predictions from the training data.

In conclusion, the supervised learning algorithms performed well on the test dataset when using k-all features, identifying *diabetes_mellitus* correctly 71% accurately. This has practical implications for ICU staff, as a trained supervised model can assist in the diagnostic stage of an ICU patient intake, particularly where patient medical records may be lacking. Additionally, it can inform both medical professionals and their patients about the indicators for *diabetes_mellitus*, considering the increasing incidence of late onset among the population [15].

REFERENCES

[1] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[2] Jayaram Hariharakrishnan, S. Mohanavalli, Srividya, and K. B. Sundhara Kumar. Survey of pre-processing techniques for mining big data. pages 1–5, 2017.

[3] I. H. Witten, Eibe Frank, Mark A. Hall, and Christopher J. Pal. *Data mining : practical machine learning tools and techniques /*. Morgan Kaufmann, 2017.

[4] Son Phung, Ashnil Kumar, and Jinman Kim. A deep learning technique for imputing missing healthcare data. pages 6513–6516, 2019.

[5] Jingke Xi. Outlier detection algorithms in data mining. 1:94–97, 2008.

[6] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. pages 413–422, 2008.

[7] Amit Sapra and Priyanka Bhandari.

[8] James Schnebly and Shamik Sengupta. Random forest twitter bot classifier. pages 0506–0512, 2019.

[9] Thae Ma Ma, Kunihito YAMAMORI, and Aye Thida. A comparative approach to naïve bayes classifier and support vector machine for email spam classification. pages 324–326, 2020.

[10] Charles Bemando, Eka Miranda, and Mediana Aryuni. Machine-learning-based prediction models of coronary heart disease using naïve bayes and random forest algorithms. pages 232–237, 2021.

[11] Liliya Demidova and Mariya Ivkina. Defining the ranges boundaries of the optimal parameters values for the random forest classifier. pages 518–522, 2019.

[12] JinHuaXu and HongLiu. Web user clustering analysis based on kmeans algorithm. 2:V2–6–V2–9, 2010.

[13] H M M T Bandara, D P Samarasinghe, S M A M Manchanayake, L P J Perera, K C Kumaradasa, N Pemadasa, and A P Samarasinghe. Analyzing payment behaviors and introducing an optimal credit limit. pages 68–72, 2019.

[14] Ting-Nien Wu and Chiu-Sheng Su. Application of principal component analysis and clustering to spatial allocation of groundwater contamination. 4:236–240, 2008.

[15] Daniel Lasserson, Robin Fox, and Andrew Farmer. Late onset type 1 diabetes. *BMJ*, 344, 2012.

A APPENDIX

Appendix A - Data shape

	qty
rows	79159
columns	88

Appendix B - Features and data types

Feature	Dtype
encounter_id	int64
hospital_id	int64
gender	object
ethnicity	object
age	float64
elective_surgery	int64
height	float64
weight	float64
bmi	float64
readmission_status	int64
icu_type	object
h1_temp_max	float64
h1_temp_min	float64
d1_albumin_max	float64
d1_albumin_min	float64
d1_bilirubin_max	float64
d1_bilirubin_min	float64
d1_bun_max	float64
d1_bun_min	float64
d1_glucose_min	float64
d1_hco3_max	float64
d1_hco3_min	float64
d1_inr_max	float64
d1_inr_min	float64
d1_lactate_max	float64
d1_lactate_min	float64
d1_platelets_max	float64
d1_platelets_min	float64
d1_potassium_max	float64
d1_potassium_min	float64
d1_sodium_max	float64
d1_sodium_min	float64
d1_wbc_max	float64
d1_wbc_min	float64
h1_albumin_max	float64
h1_albumin_min	float64
h1_bilirubin_max	float64
h1_bilirubin_min	float64
h1_bun_max	float64
h1_bun_min	float64
h1_calcium_max	float64
h1_calcium_min	float64
h1_creatinine_max	float64
h1_creatinine_min	float64
h1_glucose_max	float64
h1_glucose_min	float64
h1_hco3_max	float64
h1_hco3_min	float64
h1_hematocrit_max	float64
h1_hematocrit_min	float64
h1_inr_max	float64
h1_inr_min	float64
h1_lactate_max	float64
h1_lactate_min	float64
h1_sodium_max	float64
h1_sodium_min	float64
d1_arterial_po2_max	float64
d1_arterial_po2_min	float64
d1_pao2fio2ratio_max	float64
d1_pao2fio2ratio_min	float64
h1_arterial_pco2_max	float64
h1_arterial_pco2_min	float64
h1_arterial_ph_max	float64
h1_arterial_ph_min	float64
h1_arterial_po2_max	float64
h1_arterial_po2_min	float64
h1_pao2fio2ratio_max	float64
h1_pao2fio2ratio_min	float64
wbc_apache	float64
intubated_apache	int64
d1_heartrate_max	float64
heart_rate_apache	float64
gcs_motor_apache	float64
gcs_eyes_apache	float64
creatinine_apache	float64
bilirubin_apache	float64
h1_spo2_max	float64
paco2_apache	float64
map_apache	float64
aids	int64
cirrhosis	int64
hepatic_failure	int64
immunosuppression	int64
leukemia	int64
lymphoma	int64
solid_tumor_with_metastasis	int64
ventilated_apache	int64
diabetes_mellitus	int64

Appendix C - Numeric and Categoric features

Numeric Feature	Categoric Feature
age	encounter_id
height	hospital_id
weight	elective_surgery
bilirubin_apache	ethnicity
creatinine_apache	gender
heart_rate_apache	icu_type
map_apache	readmission_status
paco2_apache	gcs_eyes_apache
wbc_apache	gcs_motor_apache
d1_heartrate_max	intubated_apache
h1_spo2_max	ventilated_apache
h1_temp_max	aids
h1_temp_min	cirrhosis
d1_albumin_max	hepatic_failure
d1_albumin_min	immunosuppression
d1_bilirubin_max	leukemia
d1_bilirubin_min	lymphoma
d1_bun_max	solid_tumor_with_metastasis
d1_bun_min	diabetes_mellitus
d1_glucose_min	
d1_hco3_max	
d1_hco3_min	
d1_inr_max	
d1_inr_min	
d1_lactate_max	
d1_lactate_min	
d1_platelets_max	
d1_platelets_min	
d1_potassium_max	
d1_potassium_min	
d1_sodium_max	
d1_sodium_min	
d1_wbc_max	
d1_wbc_min	
h1_albumin_max	
h1_albumin_min	
h1_bilirubin_max	
h1_bilirubin_min	

h1_bun_max	
h1_bun_min	
h1_calcium_max	
h1_calcium_min	
h1_creatinine_max	
h1_creatinine_min	
h1_glucose_max	
h1_glucose_min	
h1_hco3_max	
h1_hco3_min	
h1_hematocrit_max	
h1_hematocrit_min	
h1_inr_max	
h1_inr_min	
h1_lactate_max	
h1_lactate_min	
h1_sodium_max	
h1_sodium_min	
d1_arterial_po2_max	
d1_arterial_po2_min	
d1_pao2fio2ratio_max	
d1_pao2fio2ratio_min	
h1_arterial_pco2_max	
h1_arterial_pco2_min	
h1_arterial_ph_max	
h1_arterial_ph_min	
h1_arterial_po2_max	
h1_arterial_po2_min	
h1_pao2fio2ratio_max	
h1_pao2fio2ratio_min	
bmi	

Appendix D - Numeric Descriptive Statistics

	age	height	weight	bilirubin_apache	creatinine_apache	heart_rate_apache	map_apache	paco2_apache	wbc_apac
count	76317.0	77978.0	77086.0	29109.0	64575.0	79003.0	78931.0	18540.0	61408.0
mean	62.452	169.565	85.115	1.17	1.547	99.831	87.278	42.267	12.228
std	16.447	10.835	25.466	2.276	1.604	30.622	42.059	12.379	6.928
min	0.0	137.2	38.6	0.1	0.3	30.0	40.0	18.0	0.9
25%	53.0	162.5	67.7	0.4	0.73	87.0	53.0	34.7	7.58
50%	65.0	170.0	81.6	0.7	1.0	104.0	66.0	40.1	10.5
75%	75.0	177.8	98.4	1.1	1.61	120.0	124.0	47.0	15.3
max	89.0	195.59	186.0	60.2	11.18	178.0	200.0	95.0	45.8
median	65.0	170.0	81.6	0.7	1.0	104.0	66.0	40.1	10.5
missing	2842.0	1181.0	2073.0	50050.0	14584.0	156.0	228.0	60619.0	17751.0
missing_pct	0.036	0.015	0.026	0.632	0.184	0.002	0.003	0.766	0.224
unique_values	75.0	394.0	3070.0	484.0	1097.0	148.0	170.0	682.0	2779.0
unique_pct	0.001	0.005	0.039	0.006	0.014	0.002	0.002	0.009	0.035
range	89.0	58.39	147.4	60.1	10.88	148.0	160.0	77.0	44.9

Appendix E - Numeric Features with 60% missing values

	bilirubin_apache	paco2_apache	d1_inr_max	d1_inr_min	d1_lactate_max	d1_lactate_min	h1_albumin_max	h1_albumin_
count	29109.0	18540.0	29693.0	29693.0	21350.0	21350.0	6888.0	6888.0
mean	1.17	42.267	1.583	1.47	2.907	2.075	3.001	3.0
std	2.276	12.379	0.944	0.739	3.035	2.063	0.733	0.733
min	0.1	18.0	0.9	0.9	0.4	0.4	1.1	1.1
25%	0.4	34.7	1.1	1.1	1.2	1.0	2.5	2.5
50%	0.7	40.1	1.3	1.2	1.9	1.5	3.0	3.0
75%	1.1	47.0	1.6	1.5	3.3	2.268	3.5	3.5
max	60.2	95.0	7.756	6.127	19.8	15.1	4.7	4.7
median	0.7	40.1	1.3	1.2	1.9	1.5	3.0	3.0
missing	50050.0	60619.0	49466.0	49466.0	57809.0	57809.0	72271.0	72271.0
missing_pct	0.632	0.766	0.625	0.625	0.73	0.73	0.913	0.913
unique_values	484.0	682.0	466.0	381.0	661.0	476.0	38.0	37.0
unique_pct	0.006	0.009	0.006	0.005	0.008	0.006	0.0	0.0
range	60.1	77.0	6.856	5.227	19.4	14.7	3.6	3.6

Appendix F - Categorical Features Descriptive Statistics

	encounter_id	hospital_id	elective_surgery	readmission_status	gcs_eyes_apache	gcs_moto
count	79159.0	79159.0	79159.0	79159.0	77935.0	77935.0
mean	212863.7267398527	106.18862037165704	0.18940360540178627	0.0	3.497337524860461	5.4969525
std	38113.81659134482	63.7765270037911	0.39183136576822464	0.0	0.9306487991270125	1.2571198
min	147001.0	1.0	0.0	0.0	1.0	1.0
25%	179772.0	49.0	0.0	0.0	3.0	6.0
50%	212864.0	112.0	0.0	0.0	4.0	6.0
75%	245812.0	165.0	0.0	0.0	4.0	6.0
max	278997.0	204.0	1.0	0.0	4.0	6.0
missing	0.0	0.0	0.0	0.0	1224.0	1224.0
missing_pct	0.0	0.0	0.0	0.0	0.01546255005747925	0.0154625
unique_values	79156.0	203.0	2.0	1.0	4.0	6.0
unique_pct	0.9999621015929964	0.002564458873912	2.5265604669083744e-05	1.263280233454187 2e-05	5.053120933816749e-05	7.5796814