Data Mining and Decision Systems 600092
Assigned Coursework Report
Student ID: 201601312
Date: 07 October 2019

Due Date: 12 December 2019

Methodology

Data Understanding

As the first step of the project, some informal notes were written, based off both the data dictionary and a cursory inspection of the head and tail of the dataset. [Fig 1 and cell 3 of accompanying Jupyter Notebook].

Column	Impression
Random	Numeric, likely of little use as scikit has a shuffle feature. candidate for removing.
ID	unique labels, redundant to the dataframes own id system, unless patient has had multiple sessions. How can we know which data to use if a patient has two different readings? Some may be the same. Will likely use to find duplicates. candidate to keep.
Indication	nominal value, of medical importance. Should be converted to one-hot. candidate to keep.
Diabetes	nominal value, of medical importance. keep.
IHD	nominal value, of medical importance. keep.
Hypertension	nominal value, of medical importance. keep.
Arrhythmia	nominal value, of medical importance. keep.
History	nominal value, of medical importance. keep.
IPSI	Numeric value, should be kept, potentially turned into some ranges if we need something ordinal. keep the same for the time being.
Contra	Numeric value, should be kept, potentially turned into some ranges if we need something ordinal. keep the same for the time being.
Label	Very important for training. Keep.

Fig 1: Table of first impressions

From here it could see that nothing in the data dictionary was obviously inconsistent with the actual data, and had an intuition of what would be useful in modeling could be gained.

Data Preparation

Clean

Since ‘ID’ was an anonymous patient identifier, any duplicate patient record would have duplicate ID. Based on domain knowledge, it was inferred that other possible duplicate entries are most likely to be different patients with the same symptoms. After a cursory search for duplicate IDs [Notebook cell 4] no duplicate entries were found.

The other obvious form of unclean data samples would be null values. 17 were found automatically with the .isna() function [cell 5]. No samples contained multiple null features [cell 6]. To reduce the chance of damaging or artificially reinforcing patterns, each feature that contained null values was inspected more closely to determine the next course of action with the damaged sample. If, for nominal features, a clear mode with a major lead existed, then it would be imputed, otherwise the sample containing it would have to be dropped.

Indication

Whilst trying to visually search for the modal value with a histogram, a further discrepancy in the data was found. Indication contained both ASx and Asx [fig 2]. The data dictionary specified that there should be four events: {a-f, asx, cva, tia}. Due to the larger majority of asx type being ASx (186 vs 32 [cell 8]), it was assumed that Asx was simply a clerical mistake, so all instances of Asx were replaced with ASx.

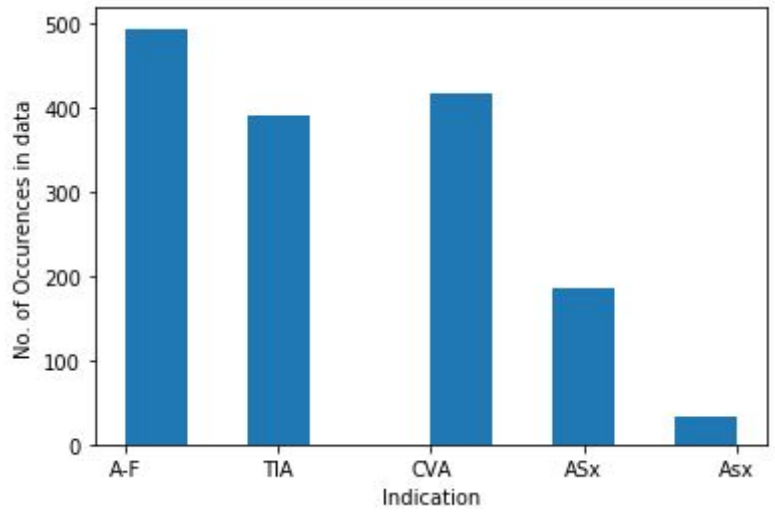

Fig 2: Histogram of ‘Indication’ Feature

Since there was no one category with a large majority, the null values were preserved in a separate dataframe before being removed from the main copy [cell 10].

Diabetes

Majority of samples carried the ‘no’ label for diabetes [Fig 3], so it seemed safe to impute all null values with ‘no’ [cell 12].

Hypertension

Hypertension had a roughly 45/55 split between yes and no values [Fig 3], so it was decided that all null containing samples should be saved to a seperate dataframe, then removed from the working copy.

History

Like Diabetes, majority of samples contained ‘no’ for History [Fig 3], so all nulls were imputed with the mode.

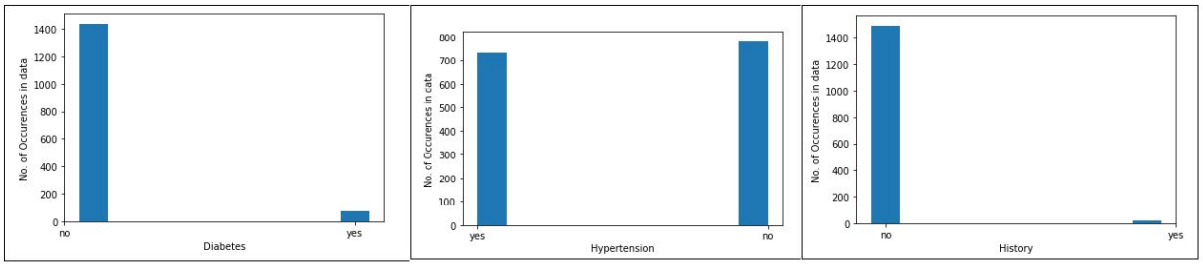

Fig 3: Histograms of Diabetes, Hypertension and History

Label

Since label was the target for classification, any imputation would damage the patterns in the data, so all nulls were copied then removed from the dataframe, with the knowledge that they could be imputed if the need arose through the use of a classification algorithm.

IPSI

IPSI was not categorical, so a combination of a boxplot [fig 4] and the pandas ‘.describe()’ [cell 19] function were used to inspect it.

With a standard deviation of 10.2, and a small interquartile range, it seemed like imputing with the mean would not damage the patterns.

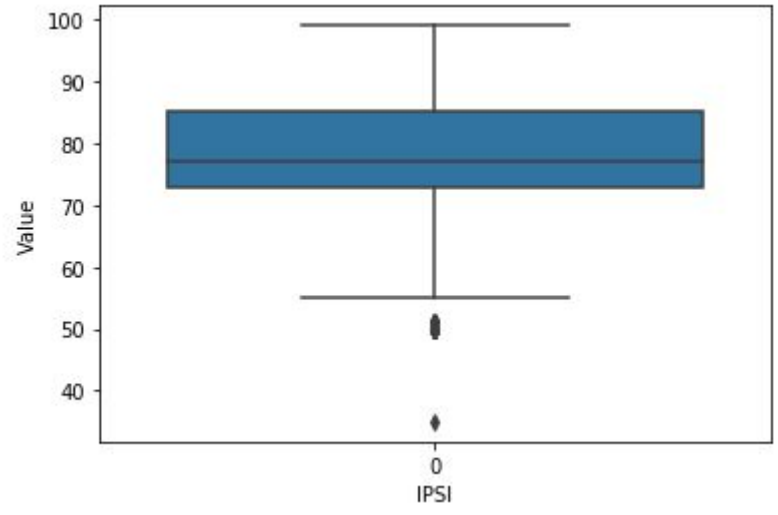

Fig 4: Boxplot of ‘IPSI’

Filter

With all duplicate entries and null values amended, it seemed prudent to begin to filter the data. Since Random did not contain any data relevant to the medical condition of each

patient, and ID was superfluous to the pandas' own indexing system, they were dropped [cells 22 and 23].

The simplest filter technique is Variance, however the pandas implementation of it: '.var()' would not work for the data as-was, since the data was not of a consistent type [cell 24].

Transform

In order to investigate the variance of the dataset, it needed to be transformed. To begin the transformation process, numerics were to be normalised wherever necessary.

IPSI, Contra, and a brief return to cleaning.

As the only integer values, IPSI and Contra were candidates to be normalised. Upon inspection however, an erroneous entry was found in Contra. The minimum was found to be ' ', an empty space [cell 29]. This sample was dropped [cell 31], and in order to find any similar dirty data instances that had flown under the radar, all unique values for every column were inspected[cell 31].

There were two instances of samples with the label, 'unknown' found with this method, and since these were going to be the classification targets, they were quarantined and dropped [cell 32] from the main copy.

Normalisation

Since Contra contained strings, and IPSI floats, but both features should be integers according to the data dictionary, they were both converted to their appropriate type. Although they had a potential range of 101, their actual range was much smaller, so they were min-max normalised to reduce dimensionality, and inconsistency of scale with boolean values elsewhere in the dataset.

Dummies and Booleans

Every example of the value 'yes' and 'Risk' was mapped to 1, and every instance of 'no' or 'No Risk' encoded to 0 [cell 35] across the entire working dataframe [cell 35].
Indication was then discretized via one-hot encoding, creating four new columns : Indication_A-F; Indication_ASx; Indication_CVA; Indication_TIA. Indication was then dropped from the working copy.

Variance

```
Hypertension      0.249867
IHD               0.249493
label             0.222518
Indication_A-F    0.219800
Indication_CVA    0.198424
Indication_TIA    0.191553
Arrhythmia        0.169508
Indication_ASx    0.123073
Contra            0.107669
Diabetes          0.047412
IPSI              0.025201
History           0.014423
dtype: float64
```

All attributes in the dataframe were now of the type 'float64', so there variance was calculable. Sorted in ascending order [Fig 5], it could be seen that no features had a variance of 0, so nothing could be immediately discarded. It was decided that although there was likely room for further reduction of dimensionality, the data was now in a state fit for classification; and depending on results this stage of the methodology could be returned to.

Fig 5. Results of a variance query

Modelling

Train-Test Split

A shuffled train-test split with a ratio of 7/3 was created [cell 39], to be used for all models.

Decision Tree

The decision tree was created, fitted to the training split, then called upon to predict the test data. With an accuracy score of 0.98 [cell 115], a mean squared and absolute error of less than 0.1, this model seemed very successful, however several problems were also apparent.

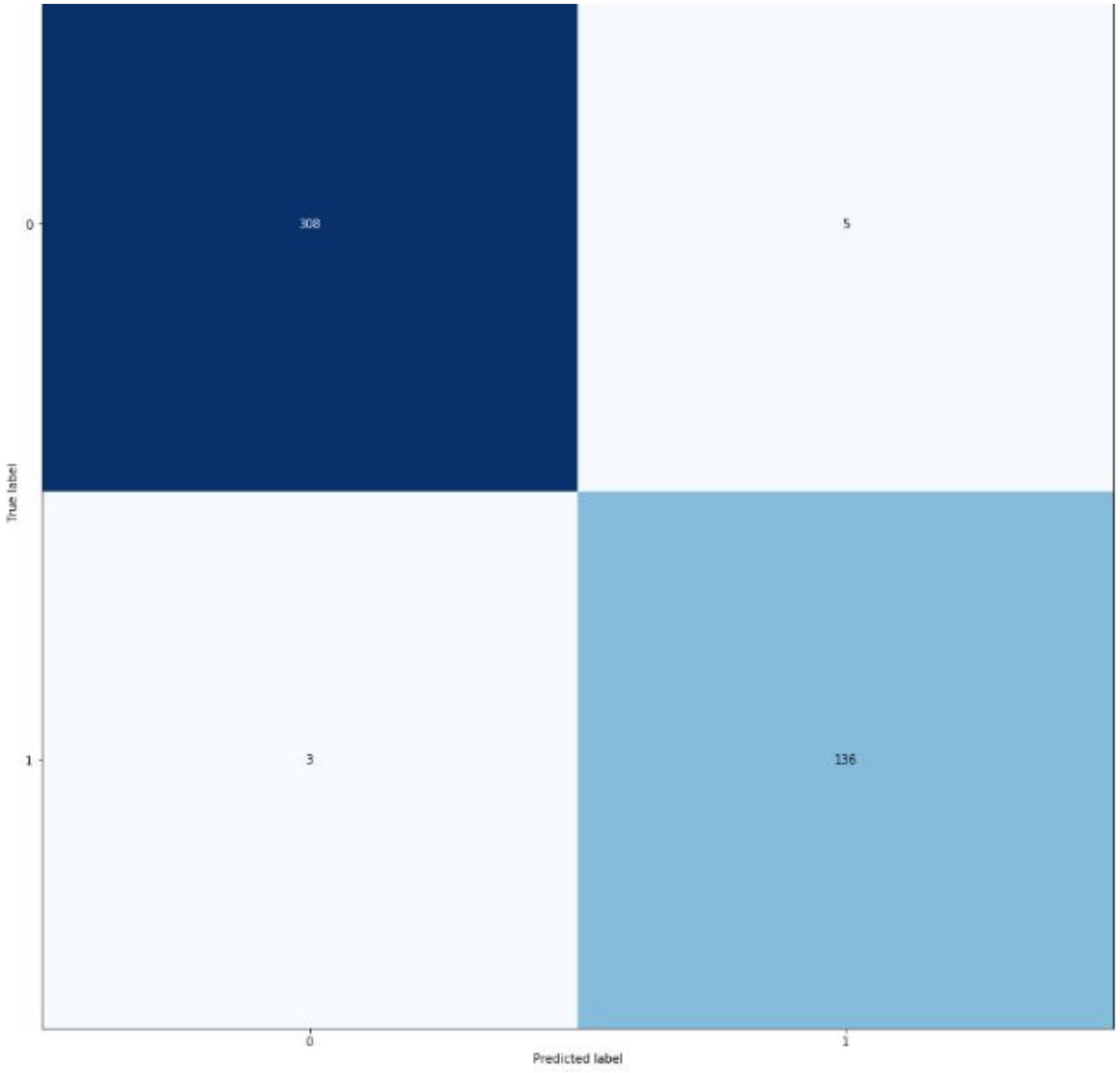

Fig 6. Confusion Matrix of Decision Tree Classifier

Problems with the initial model

Since every instance of a false negative could mean someone at risk not being treated, Sensitivity was deemed the most important metric to be optimised. This model had a sensitivity of 0.97 [cell 48], which was still good, but not perfect.

The large class imbalance between risk and no risk implied that this model might not generalise well, as it simply did not have enough information on at risk patients to make an informed decision. Two possible solutions to this problem were identified: K-fold cross validation and weighting the model by penalising false negatives more strictly. I chose to attempt an implementation of K-fold cross validation first.

K-Fold Cross Validation

In order to find the best combination of *K,* shuffled splits and stratification, every possible combination was tried [cell 46]. The range of *K*'s was 2-15, and each *K* was tried both shuffled and ushuffled, stratified and unstratified. The variable compared was sensitivity, and during the first run of the procedure, a shuffled, stratified split, with $k = 4$ reported a sensitivity of 0.988.

Against the test data however, this model reported the same sensitivity as the unvalidated model with 0.97.

Weighting

From here, weighting the labels so as to exaggerate the importance of the underrepresented labels was attempted.

Weighting an otherwise unmodified decision tree with the ‘balanced feature’ i.e: n_samples / (n_classes * np.bincount(y)) resulted in a sensitivity of 0.97 [cell 52], still equal performance to the default Decision Tree.

Weighting a stratified, shuffled, k-validated (k=10) decision tree initially produced impressive results: When testing on validation data, a sensitivity of 1.0 was reported; a perfect score. Against the test split however, sensitivity was down to 0.99,which was still an excellent result. This indicated that the model was overfitting slightly, however with this sensitivity score the project was ready to continue.

Results

Fig 7. The final version of the modified dataset (head and tail only).

	Diabetes	IHD	Hypertension	Arrhythmia	History	IPSI	Contra	label	Indication_A-F	Indication_ASx	Indication_CVA	Indication_TIA
0	0	0	1	0	0	0.671875	0.111111	0	1	0	0	0
1	0	0	0	0	0	0.546875	0.555556	0	0	0	0	1
2	0	1	1	0	0	0.937500	0.333333	1	1	0	0	0
3	0	0	1	0	0	0.859375	0.833333	1	0	0	0	1
4	0	0	0	0	0	0.546875	0.111111	0	0	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...
1515	0	1	0	0	0	0.640625	0.555556	0	1	0	0	0
1516	0	0	1	1	0	0.859375	0.722222	1	1	0	0	0
1517	0	0	1	0	0	0.625000	0.111111	0	0	0	0	1
1518	0	1	0	0	0	0.546875	0.388889	0	1	0	0	0
1519	0	0	0	0	0	0.390625	0.111111	0	0	0	1	0

1504 rows × 12 columns

Fig 8. Comparison of metrics for all models used

Model	Sensitivity	Specificity	Precision	Accuracy
DecisionTree()	0.97	0.99	0.97	0.98
KFold(k=5, shuffle = True) DecisionTree()	0.97	0.99	0.98	0.98
DecisionTree(class_weight = ‘balanced’)	0.97	0.99	0.97	0.98
StratifiedKFold(k=10, shuffle = True) DecisionTree(class_weight = ‘balanced’)	0.99	0.99	0.98	0.99

Evaluation & Discussion

Data Understanding

I had very little problems during the data understanding stage, as the data dictionary, while not perfect gave me a good enough understanding of the data to begin the other stages of the project.

Data Preparation

The preparation stage was quite difficult, as many issues with the data were not immediately clear, resulting in frequent movement between cleaning, filtering and transformation of the data, but in the end the data was ready for modeling relatively fast.

Given the opportunity, it would have been prudent to spend more time reducing dimensionality, and understanding the relationship between features, but overall the data was prepared in such a way that good classification accuracy was possible.

Modeling

Due to the medical nature of the applications for this dataset, it seems unlikely that patients and doctors would be happy to accept decisions from a system that is not explainable. A black box model would not be appropriate, so a decision tree was chosen as they had enough predictive power to create very good results; and they can be easily traced by navigating the branches to find the reasons for their labeling.
Comparing sensitivity was deemed the most important metric, as a perfect sensitivity would mean all instances of risk identified, while a false positive, while potentially traumatic would not put someone's life at risk. The class imbalance was an issue, but with the tools of class weighting and k-fold validation, it was minimised as much as possible.

Throughout the process, a difference of about 0.02 across all metrics was all that was changed throughout the process, which suggests that while Decision Trees are a good model for this type of problem, it would be very arduous if not impossible to obtain a perfect result from them.

Given the opportunity, a system that provides a confidence score, rather than a binary output would probably be better for a medical advisory system, as the goal is not to completely eliminate human doctors from diagnosis, but simply advise them based on data that would not be feasible for them to sift through by hand.