Assessment Task 3: Data Mining in Action

Table of Contents

[**Description of the Data Mining Problem** 2](#_Toc54044075)

[**Data Preprocessing and Transformations** 2](#_Toc54044076)

[**Data Cleaning** 2](#_Toc54044077)

[**Data Transformations** 3](#_Toc54044078)

[**Methodology used to solve the Problem** 4](#_Toc54044079)

[**Classification Techniques used** 6](#_Toc54044080)

[**Summary of the Results and Parameter Settings** 6](#_Toc54044081)

[**The best classifier built** 10](#_Toc54044082)

**Description of the Data Mining Problem**

As a data scientist at a consultant company I have be tasked with a data mining problem given involves determining whether a property will qualify to be sold on the market or not (binary classification problem).

A training data set is given for us to fit our classifier to that data as well as an unknown dataset (the same but does not include the qualified attribute) which is what we will be basing qualified prediction off.

The tools I am be using to solve this problem is Excel to view csv files in a spreadsheet format and do more basic analysis (using the built-in charts and functions). In combination with Python and a variety of established statistical and machine learning libraries to do the bulk of the supervised and unsupervised learning. The main libraries used include:

* Pandas: To read, manipulate and store structured data in the form of dataframes or csv (comma separated values) files. To also provides basic statistical and data visualisation functionality.
* Sklearn: Used for data normalisation and there classifier functions. It is a large library and had implementations for all types of classifiers I wanted to use for predictions.
* Matplotlib: A vast and powerful library for data visualisation. The majority of pandas visualisations actually use this library.

**Data Preprocessing and Transformations**

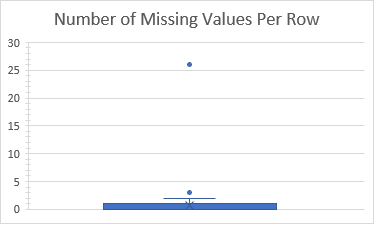
**Data Cleaning**

The original dataset contained 36 attributes including the qualified attribute and had a size of approximately 17 megabytes. My first task was to check the proportion of missing data and depending on the proportion of missing data from each attribute and row.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | BATHRM | HF\_BATHRM | HEAT | HEAT\_D | AC | NUM\_UNITS | ROOMS | BEDRM | AYB | YR\_RMDL | EYB |
| Number Missing | 20 | 21 | 20 | 20 | 20 | 20 | 32 | 24 | 10 | 40473 | 0 |
| Proportion | 0.02666418 | 0.02799739 | 0.02666418 | 0.02666418 | 0.02666418 | 0.02666418 | 0.04266268 | 0.03199701 | 0.01333209 | 53.9589638 | 0 |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | STORIES | SALEDATE | PRICE | SALE\_NUM | GBA | BLDG\_NUM | STYLE | STRUCT | GRADE | CNDTN | EXTWALL |
| Number Missing | 52 | 0 | 13506 | 0 | 0 | 0 | 20 | 20 | 20 | 20 | 20 |
| Proportion | 0.06932686 | 0 | 18.0063194 | 0 | 0 | 0 | 0.02666418 | 0.02666418 | 0.02666418 | 0.02666418 | 0.02666418 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | ROOF | INTWALL | KITCHENS | FIREPLACES | USECODE | LANDAREA | GIS\_LAST\_MOD\_DTTM | QUALIFIED |
| Number Missing | 20 | 20 | 21 | 21 | 0 | 0 | 0 | 0 |
| Proportion | 0.02666418 | 0.02666418 | 0.02799739 | 0.02799739 | 0 | 0 | 0 | 0 |

Since the AC, STYLE, STRUCT, GRADE, CNDTN, EXTWALL, ROOF and INTWALL attributes are categorical there was no global value representative of the entire same (e.g. mean or median) that I could use. Considering that 75% of the property instances only have one attribute value missing this means there is a high chance that each missing attribute value (of the aforementioned attributes) is the only attribute missing in that row/instance. Therefore, the worst-case scenario is that by removing each row with one of the missing attribute values mentioned, that we are removing 140 rows which equates to 0.018% of the sample size. Therefore, this should not affect our model as it is a relatively insignificant reduction in size.

For the remainder of the attributes that have missing values I have replaced those missing values with the median of the attribute except for PRICE and attributes that are a year or date. In relation to price this is because 18.01% of the sample are missing values for price and 27.65% of the values are 0. My hypothesis is that this is a result of the data entry personnel not having a price because it hasn’t sold recently (could be for other reasons as well) and so they either did not enter a price or entered 0. The attributes that are a year or date have this problem as well but with differing statistics. Therefore, I will be initially replacing all empty attribute values for these attributes to 0.

**Data Transformations**

Through observation of the training dataset in Excel I determined which attributes were non numeric so I could do some sort of conversion to a numeric value.

The AC holds a Boolean value but it is non-numeric (true was represented by Y and false was presented by N). However, this was a simple fix as all I had to do was replace the current row with a 1 if it contained Y and a 0 if it contained N.

This is the excel function I used to make this change: AC=IF(F2="Y",1,0)

The description attributes (seven of them and can be identified if they have an \_D at the end of the attribute name) was String data. Upon reviewal of the attribute description table provided by the company I am consulting for, I realised that it is likely that each code has a singular unique description (i.e. STYLE code of 7.0 means that the description is 3 story). To check this I created an function that converted the dataframe into a list I could then loop through and count the number of instances of any unique value in the description attributes (e.g. HEAT\_D) for each code attribute (e.g. HEAT).

This function returns each unique value per code. For example, this the result it produced for the STYLE and STYLE\_D:

STYLE

7.0 1 {'3 Story': 6710}

10.0 1 {'4 Story': 254}

4.0 1 {'2 Story': 56967}

6.0 1 {'2.5 Story Fin': 4942}

9.0 1 {'3.5 Story Fin': 90}

1.0 1 {'1 Story': 3068}

3.0 1 {'1.5 Story Fin': 1872}

5.0 1 {'2.5 Story Unfin': 509}

0.0 1 {'Default': 46}

8.0 1 {'3.5 Story Unfin': 4}

2.0 1 {'1.5 Story Unfin': 81}

12.0 1 {'4.5 Story Fin': 9}

14.0 1 {'Split Level': 225}

13.0 1 {'Bi-Level': 13}

15.0 1 {'Split Foyer': 193}

99.0 1 {'Vacant': 2}

94.0 1 {'Outbuildings': 1}

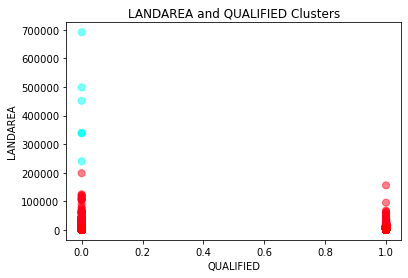
11.0 1 {'4.5 Story Unfin': 1}

{'year': [2018], 'month': [7], 'day': [22]}

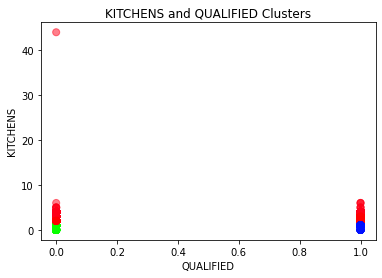
The extended version can be seen in my cleaning Jupyter Notebook and proves that my hypothesis was correct.

**Methodology used to solve the Problem**

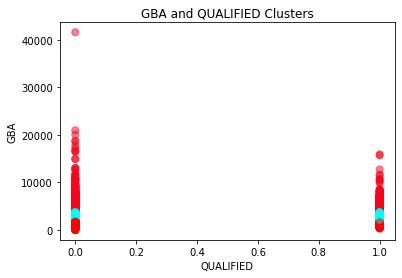
Before creating and fitting classifiers, I decided to do some clustering to determine if I could gain any insights from it and then potentially use them to tweak my classifiers. I clustered each attribute with the qualified attribute, but the only somewhat significant results are produced are shown below:



Shows that there are clusters under 150000 ft^2 landarea in both unqualified and qualified properties. However, there is a dispersed cluster above 200000 ft^2 only present in the unqualified properties. Therefore, a classifier should hopefully determine this as a metric (or decision in a decision tree) to determine whether a property qualifies.



Clustering kitchens and qualified attributes show that was no relationship or correlation between the number of kitchens and whether a property qualifies. This underscores that fact that some attributes don’t have a direct correlation but a classifier could potentially still use this to pick up a pattern.

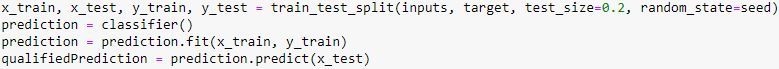


Clustering Gross Building Area and qualified attributes highlights that a cluster of values between 15000 and 21000 ft^2 is only present in unqualified properties. Therefore, a classifier should hopefully determine this as a metric (or decision in a decision tree) to determine whether a property qualifies.

After clustering I had to create four different datasets for training and testing purposes using random partitioning. Essentially this means, despite the proportion of training and test data being the same (80% of the training dataset was for fitting/training the classifier and 20% for testing), the rows selected to be in either partition (training or testing) were random.

* x\_train and y\_train: this would be the parameters given to the classifier to fit the data to the qualified attribute (how it will determine if a property qualifies or not). The x\_train dataset contains all the attributes except the qualified attribute and the y\_train dataset only contains the qualified attribute.





* x\_test: this is the data the classifier will use to predict the unknown (at least before it is checked against y\_test) qualified attribute.

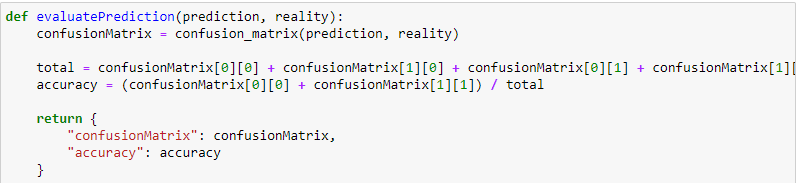


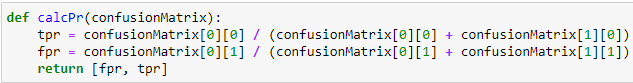
* y\_test: once the predicted qualified attribute is computed by the classifier’s predict function, it is checked against y\_test which is what the qualified attribute was in reality.



Once the classifier had been fitted and made a prediction I used several different metrics to evaluate it’s performance.

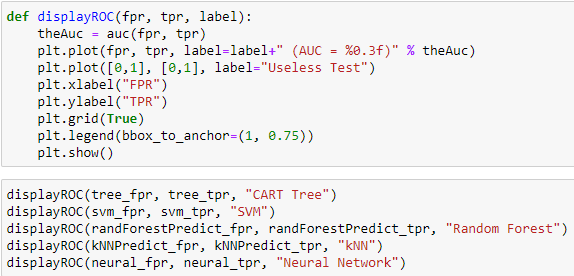
The evaluate prediction function produces a confusion matrix and accuracy rate. Once this is returned I could also calculate the True Positive Rate and False Positive rate using the confusion matrix.

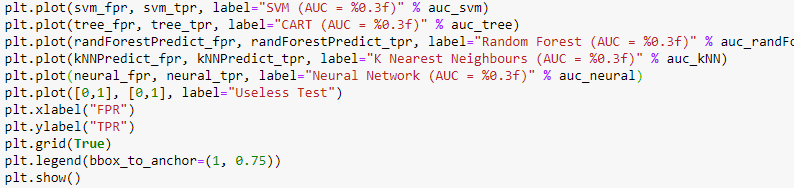




The final metric used in evaluating the performance of the classifier is displaying the ROC curve and the Area Under the Curve. This will show us the probability of the classifier making an incorrect classification and for what values it is more likely to do so.

Code to produce ROC curves and calculate AUC:





**Classification Techniques used**

**Summary of the Results and Parameter Settings**

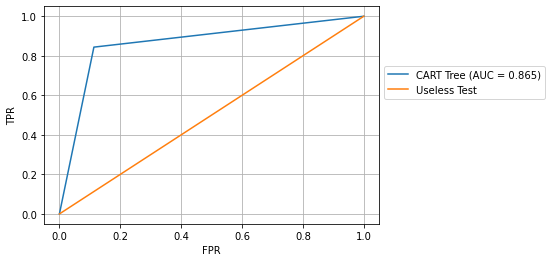
CART Decision Tree:

|  |  |  |  |
| --- | --- | --- | --- |
| TN | FN | FP | TP |
| 7663 | 979 | 993 | 5363 |

Accuracy: 86.85%

TPR: 88.67%

FPR: 15.62%



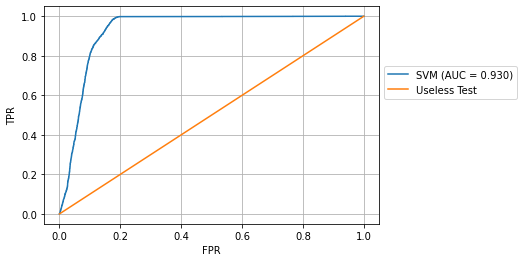
Support Vector Machine:

|  |  |  |  |
| --- | --- | --- | --- |
| TN | FN | FP | TP |
| 7052 | 1590 | 59 | 6297 |

Accuracy: 89.01%

TPR: 81.60%

FPR: 0.93%



Random Forest:

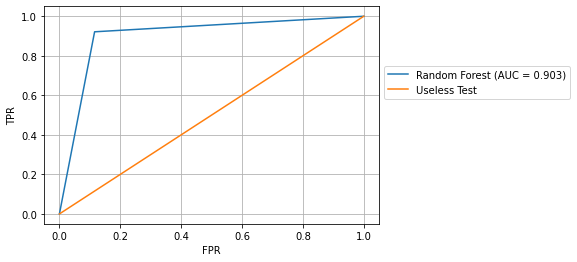
The number of estimators chosen for the Random Forest was 10 (arbitrary value).

|  |  |  |  |
| --- | --- | --- | --- |
| TN | FN | FP | TP |
| 7643 | 999 | 500 | 5856 |

Accuracy: 90.01%

TPR: 88.44%

FPR: 7.87%



K Nearest Neighbours Classification:

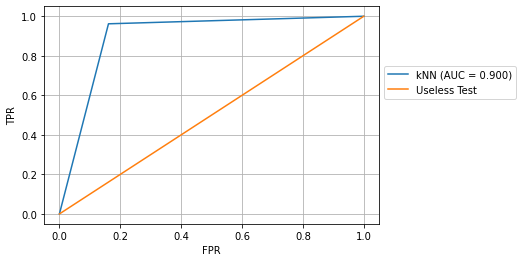
K number of neighbours was chosen by computing the accuracy of the prediction for the k [1, 10] and selecting k which gave the maximum accuracy.

|  |  |  |  |
| --- | --- | --- | --- |
| TN | FN | FP | TP |
| 7249 | 1393 | 243 | 6113 |

Accuracy: 89.09%

TPR: 83.88%

FPR: 3.82%



Neural Network:

The number of hidden layers for the Neural Network was chosen by using a rule-of-thumb method that states the number of hidden neurons should be two thirds the size of the input layer (thirty attributes).

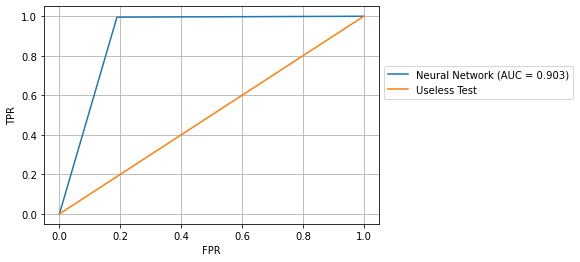
Source: https://stackoverflow.com/questions/52485608/how-to-choose-the-number-of-hidden-layers-and-nodes/52498226

|  |  |  |  |
| --- | --- | --- | --- |
| TN | FN | FP | TP |
| 7008 | 1634 | 29 | 6327 |

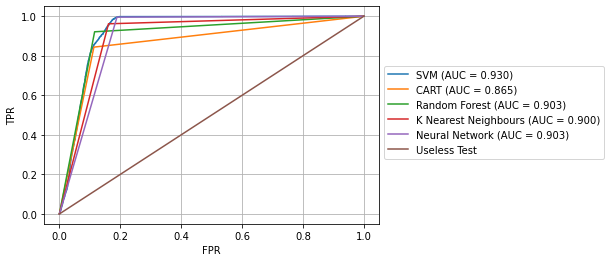
Accuracy: 88.91%

TPR: 81.09%

FPR: 0.46%



ROC curves of each classifier combined:



**The best classifier built**

As shown in the previous section there are a variety of different ways to determine the performance of a classifier.

The classifier with the highest accuracy rate was the Random Forest Classifier with an accuracy rate of 90.01%. The highest AUC was produced by the Support Vector Machine which had an AUC equal to 0.93. And the Support Vector Machine as well as the Neural Network produced the lowest number of false positives (under 0.1% of the test dataset).

Given that in the description of the data mining problem it was determined that the sole purpose of this exercise was to produce the best predictions of property qualification overall, the best classifier should be the one that performed best overall. Hence, the Random Forest classifier is what I would deem the best classifier for this scenario.

**The Jupyter Notebooks and Datasets I produced**

https://drive.google.com/drive/folders/1CTnuPk-HFjMLuHfHXHlOWCLRrjRllyk-?usp=sharing