Predicting Medicare Fraud Using Machine Learning Techniques

kmjacinto2145

Executive Summary

This report details the techniques used to prepare and model Medicare Fraud data from 2009, with the goal of identifying 350 Medicare providers that are most likely to be fraudulent from a new sample of 974 providers. This is a continuation of a previous report submitted in early July.

Before machine learning techniques could be applied, the dataset, which consisted of records of Medicare claims, had to be prepared. The dataset went through data cleaning and feature engineering, before ultimately being combined with a dataset on providers and aggregated to give a dataset that contained characteristics of each provider such as the average patient age and the number of claims per provider.

As part of the modelling process, some techniques were used to improve the accuracy of the machine learning models, such as principal component analysis, shrinkage, and k-fold cross validation. During the modelling process itself, several statistical learning techniques were applied, including logistic regression, K-Nearest Neighbours, random forests, and support vector machines, with the results of most of these methods located in the Appendix. Ultimately, a principal component random forest model was chosen for its near-perfect accuracy.

This report also outlines the typical characteristics of fraudulent providers. For instance, it was found that fraudulent providers had more claims and longer claim process times than non-fraudulent providers.

Data Preparation

To determine what parts of the dataset could be cleaned and manipulated, descriptive statistics such as the mean and count of each variable were put together, with a sample of the results provided in the Appendix. In addition, the dataset was checked for duplicate and outlier values.

This initial inspection of the data found that while there were no duplicate values present, there were many missing values within the date, physician, and code variables.

Likely reason for missing values	**Variables**
Human error	*ClaimStartDt, ClaimEndDt, AttendingPhysician*
Data not required for outpatients	*AdmissionDt, DischargeDt, DiagnosisGroupCode, ClmProcedureCode*
Not all slots needed	*OperatingPhysician, OtherPhysician, ClmDiagnosisCode, ClmProcedureCode*

Before the missing values were removed, several new variables were developed as part of the feature engineering process. These are:

- **ClaimProcessTime** in days, which was calculated by subtracting the claim start date from the claim end date.
- **Age**: calculated by subtracting each beneficiary's date of birth from their date of death (if the beneficiary had died) or the 1st of January, 2010 and converting the units from days to years.
- **ClmDiagnosisCount**: calculated by counting the number of diagnosis codes for each claim.

- **ClmProcedureCount**: calculated by counting the number of procedure codes for each claim.
- **Count**: the number of claims for each provider.
- **Dead**: indicates whether the patient died. The provider-level variable Dead gives the proportion of patients who died for each provider.

Several techniques were used to treat the missing values prevalent in the data. Some columns that featured missing values, such as *DiagnosisGroupCode*, were removed. In fact, all variables featuring codes were removed, particularly because it was difficult to use these variables in the final model. Other missing values, such as in *ClaimProcessTime* as mentioned beforehand, underwent multivariate imputation, which involves inferring the value of each missing data point by using the rest of the dataset.

Modelling Approach

Rebalancing the Data

The aggregated dataset is somewhat imbalanced, with only 440 of the 4,436 providers confirmed as fraudulent. An imbalanced dataset can be detrimental to modelling most classification models are sensitive to imbalances in the classes - in this case, a predictive model is likely to (incorrectly) classify fraudulent providers as non-fraudulent. To solve this issue, the dataset was upsampled; in other words, new samples of fraudulent providers were developed so that the number of fraudulent and non-fraudulent providers was even.

This problem could also have been solved by "downsampling" the non-fraudulent providers, i.e. reduce the number of non-fraudulent records, however, this is in itself problematic as downsampling will result in a small sample size.

Dimensionality Reduction

The final aggregated dataset consists of 31 explanatory variables. As mentioned in the previous report, having a large feature space is problematic as there may be issues of multicollinearity, i.e. strong correlation between variables, and sparseness of the data, which can inflate the amount of storage space and processing time required.

To solve this issue, we employ Principal Component Analysis (PCA), a dimensionality reduction technique that projects data of a higher-dimensional space into that of a lower dimension. This is achieved by creating new, uncorrelated variables that maximise the variance of each dimension. In this case, we compress the 31 variables of the original aggregated dataset into 10 new predictors, with 10 predictors being chosen as it struck a balance between compressing the dataset and still capturing as many variables as possible.

Data Scaling

For some machine learning models such as K-Nearest Neighbours and Support Vector Machines, the explanatory variables need to be "scaled", in other words, the data centred around 0 and scaled with respect to the standard deviation:

$$x_{standardised} = \frac{x - \mu}{\sigma}$$

The reason why the data must be scaled for both K-Nearest Neighbours and Support Vector Machines is that these two models use the distance between data points as part of

model training. This unfairly emphasises variables that have a relatively large range such as *InscClaimAmtReimbursed*, which varies from 0 to nearly 60,000.

Cross-Validation

When comparing the performance of predictive models, we aim to minimise the test error of the models. To do this, cross-validation techniques were applied. For this problem, a combination of k-fold cross validation (where k = 10) and the validation set approach (with test size = 30%) was used.

The validation set approach, which involves training the model on part of the training data and testing the model on the other part, was used as it allowed certain performance metrics such as confusion matrices and AUC scores to be calculated. K-fold cross-validation works by randomly dividing the observations into k groups of equal size, and using each of the k folds as a validation set before averaging the results. Ultimately, the k-fold cross-validation method was prioritised as it removed the randomness associated with the validation set approach, and reduced the bias of the model.

Hyperparameter Selection and Tuning

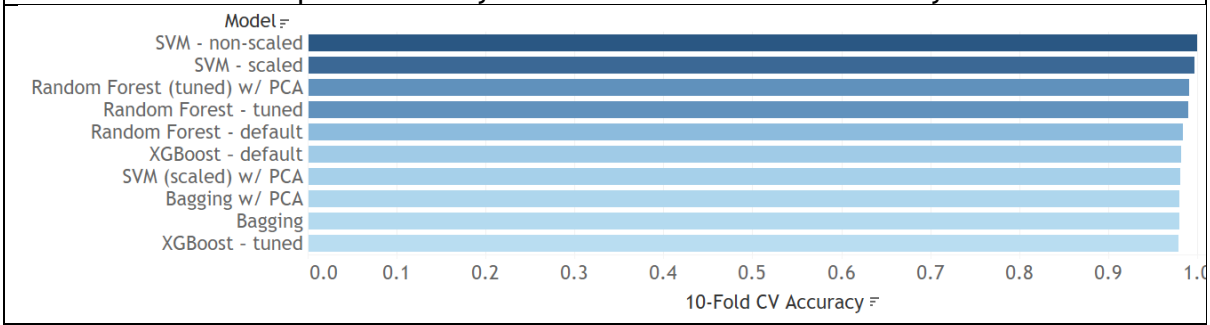
The hyperparameters, i.e. the parameters used to control the learning processes for each machine learning model, have a profound impact on the performance of the models. As an extreme example, a Support Vector Machine model trained using the default hyperparameters could predict that all data points will belong to the same class.

To alleviate this issue, some models underwent hyperparameter tuning, i.e. experimenting with different sets of hyperparameters to find the most optimal combination based on a metric specified for each model. This was achieved through performing a grid search, which involves specifying a set of potential values for each hyperparameter and then testing each combination of hyperparameters. For example, the Support Vector Machine model used in this problem was tested using 5 values of hyperparameter *C* and 5 values of hyperparameter *gamma*, which resulted in 25 different SVM models being tested.

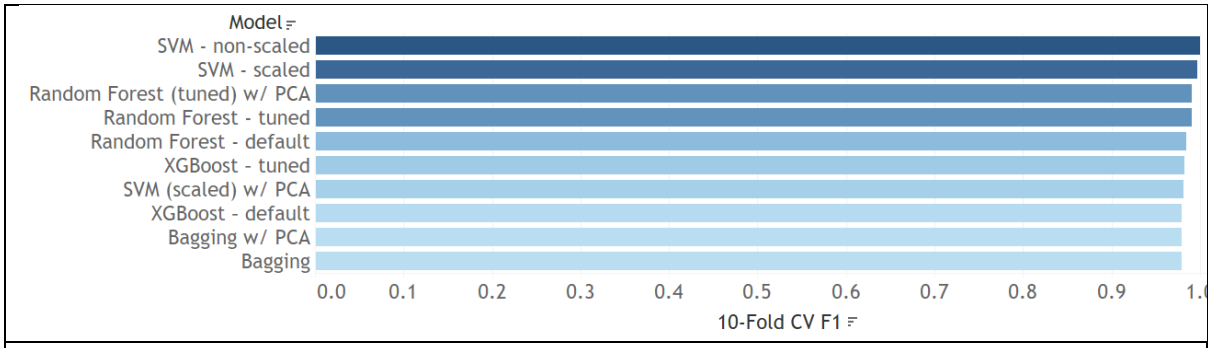
Summary of Models

The modelling stage involved experimenting with over 25 different machine learning models, ranging from logistic regression to neural networks. The models were judged on a combination of their 10-fold cross-validation F1 score, their 10-fold cross-validation accuracy score, their validation set F1 score, and their AUC score. More details about these metrics can be found in the Appendix.

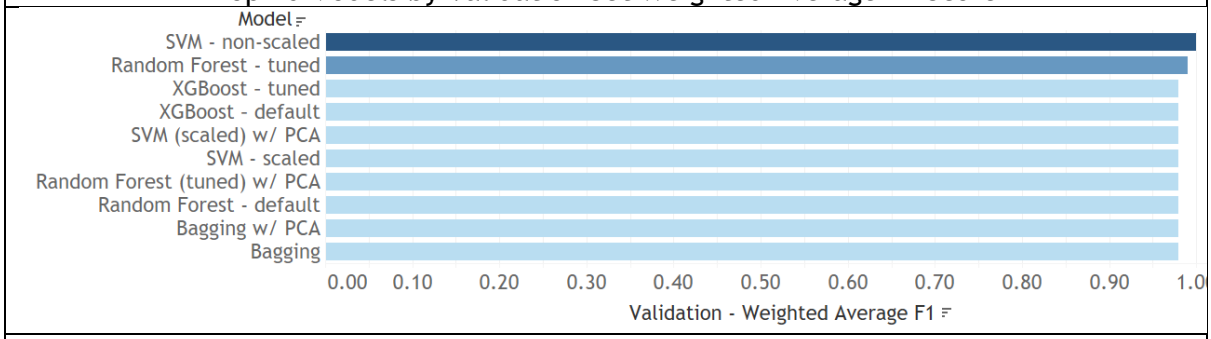
Top 10 Models by 10-Fold Cross-Validation Accuracy



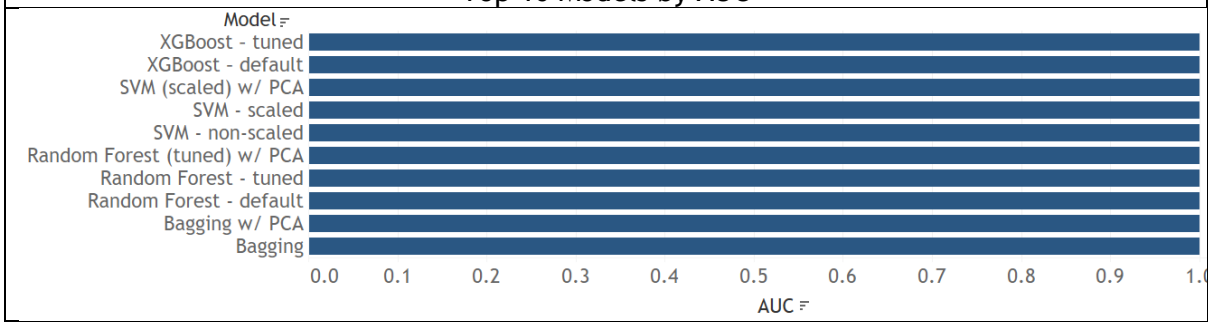
Top 10 Models by 10-Fold Cross-Validation F1 Score



Top 10 Models by Validation Set Weighted Average F1 Score



Top 10 Models by AUC



Because we are using statistical learning techniques to predict new fraudulent providers rather than identify the factors that make a provider fraudulent, we want to **maximise predictive accuracy at the expense of interpretability**. The four charts depict the top 10 machine learning models based on the four aforementioned metrics. At first glance, the models featured in these charts all originate from one of four types:

- Support Vector Machines (SVM)
- Random Forests
- Xtreme Gradient Boosting, i.e. XGBoost
- Bagging

Although all four models perform similarly, there are some key differences in terms of ease of use:

- Although both the scaled and non-scaled support vector machine models had near-perfect accuracy, these models suffer from requiring a **grid search** to tune the hyperparameters, a process that increases the runtime of these models.
- The random forest models in this problem did undergo hyperparameter tuning, however, only one of its parameters, *bootstrap*, needed to be experimented with. In fact, the random forest model can be **fitted very accurately without hyperparameter tuning** if necessary: the default random forest model is equal first in terms of AUC and in the top 5 models according to the other three metrics.

- When training SVM models, the **exogenous variables must be scaled** to avoid overemphasising variables with larger ranges, and hence the non-scaled SVM model is most likely invalid. The dataset does not need to be scaled when using a random forest model.

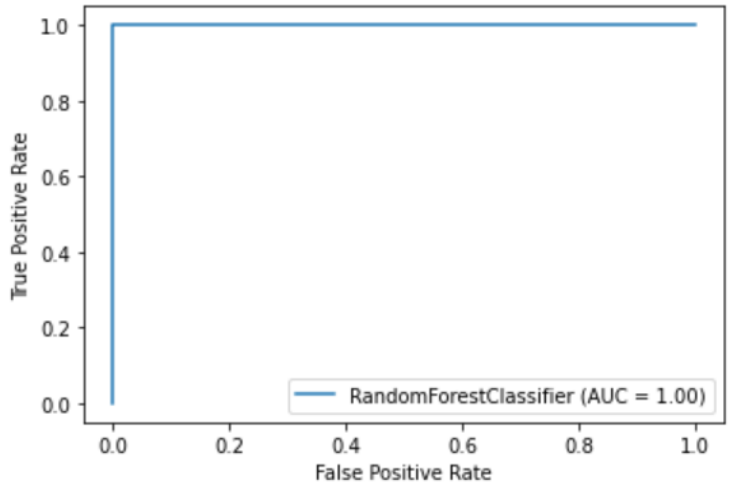
In addition:

- The random forest PCA model outperforms the bagging models on all four metrics. It also outperforms the XGBoost models in the k-fold cross-validation metrics.
- Due to the large feature space, it was pivotal that **principal component analysis** was used in the training data. Hence, non-PCA models were not prioritised compared to PCA models, even if they performed better than their equivalent PCA models.

Ultimately, the **hyperparameter-tuned Random Forest model with PCA** was chosen as the most suitable model for the final forecasts due to its combination of accuracy, ease of use, and dimensionality reduction. As shown in the image below, the random forest model does not mis-classify a single fraudulent provider as non-fraudulent, and has a perfect AUC value of approximately 1.

```
[[1190   30]
 [   0 1202]]
              precision    recall  f1-score   support

           0       1.00      0.98      0.99      1220
           1       0.98      1.00      0.99      1202

    accuracy                           0.99      2422
   macro avg       0.99      0.99      0.99      2422
weighted avg       0.99      0.99      0.99      2422
```

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay object at 0x0000020D187B2248>
```



The hyperparameters used in this model are:

- n_estimators = 200
- max_depth = None
- min_samples_leaf = 1
- min_samples_split = 2
- bootstrap = False

These hyperparameters were selected as a result of a grid search.

The random forest model works by:

1) Creating a **bootstrapped dataset**, i.e. creating a new dataset by randomly selecting samples from the original dataset
2) Developing a **decision tree** using the bootstrapped dataset, selecting only a random subset of columns at each step.
3) Step 2 is then repeated multiple times, resulting in **numerous trees**. In Python, the number of trees that are grown is set to 100 by default. In the
4) The **bootstrapped dataset is then** tested on all the trees that were created. The predicted value given the dataset is based on which class was classified more often. For example, if there were 100 trees, and a provider was classified as fraudulent 51 times, they would be ultimately classified as fraudulent. This process is known as **bagging**.
5) Entries that did not appear in the bootstrapped dataset (also known as **"out-of-bag"** samples) are tested on all the trees that did not feature that sample.
6) In our model, *bootstrap* is set to False, which results in the entire training set being used as the bootstrap set.

Typical Characteristics of Fraudulent Providers

- Fraudulent providers tend to make **significantly more claims than non-fraudulent providers**. As shown in Figure A2.1, on average, fraudulent providers made more than 380 claims a year, whereas non-fraudulent providers made approximately 70.
- Fraudulent providers have **longer claim process times**. Figure A2.2 shows that fraudulent providers spend a median of 2.4 days processing a claim, as opposed to the 1.48 days spent by non-fraudulent providers
- As mentioned in the July report, **claims by fraudulent providers receive significantly larger reimbursements**. Figure A2.3 shows that the average claim reimbursement amount for those with chronic health conditions is 30-60% higher than those who do not have chronic health conditions. Consequently, fraudulent providers are more likely to (maliciously) claim for individuals with chronic health conditions, as shown in Figure A2.4, to take advantage of the higher payout.

Limitations of Analysis

- **Inconsistent use of multivariate imputation**: due to missing values in *ClaimStartDt* and *ClaimEndDt* in the training dataset, there were some null values in *ClaimProcessTime* that were imputed. However, in the evaluation dataset, there were no null values for *ClaimProcessTime*.
- **Arbitrary selection of k value for k-fold cross-validation:** the use of 10-fold cross-validation is arbitrary despite being widely used in machine learning. This could be alleviated by using Leave-One-Out Cross Validation (LOOCV) but this is computationally expensive and high in variance.

Appendix

Alternative Models

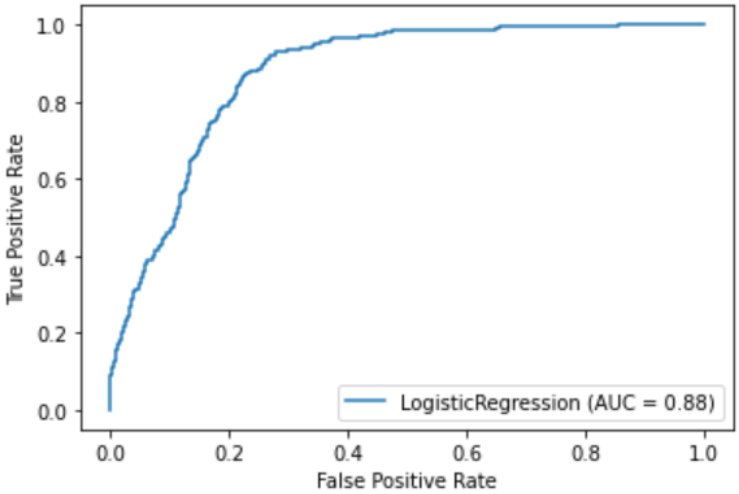
This table summarises the results of all the machine learning models used in this investigation. More detailed results, including the confusion matices and ROC curves for selected models, are also provided.

	10-Fold CV F1 score	10-Fold CV Accuracy	Validation set - Weighted Average F1 score	AUC
Logistic regression	0.77543	0.784688	0.79	0.88
Logistic - best subset	0.772553	0.782085	0.78	0.88
Ridge regression	0.712986	0.744795	0.72	0.85
Lasso regression	0.775385	0.785183	0.78	0.88
Linear Discriminant Analysis	0.712718	0.744671	0.72	0.85
Quadratic Discriminant Analysis	0.756627	0.720516	0.73	0.85
KNN - non-scaled	0.966407	0.965187	0.96	0.96
KNN - scaled	0.950703	0.94809	-	-
Decision Tree	0.970326	0.969401	0.97	0.97
Random Forest - default	0.98453	0.984267	0.98	1
Random Forest - tuned	0.990558	0.990461	0.99	1
SVM - scaled	0.997284	0.997275	-	-
SVM - non-scaled	1	1	1	1
Perceptron	0.792946	0.786302	0.81	0.84
XGBoost – default	0.979766	0.981913	-	-
XGBoost – tuned	0.982268	0.979064	0.98	1
Bagging	0.979397	0.979683	0.98	1
Gradient Bootsted Models	0.932556	0.929014	0.93	0.97
Logistic regression w/ PCA	0.611708	0.658821	0.64	0.73
Ridge regression w/ PCA	0.595997	0.652254	0.64	0.73
Lasso regression w/ PCA	0.61125	0.658821	0.64	0.73
Linear Discriminant Analysis w/ PCA	0.595997	0.652254	0.64	0.73
Quadratic Discriminant Analysis w/ PCA	0.731333	0.667987	0.67	0.82
KNN (scaled) w/ PCA	0.960225	0.958497	0.95	0.95
Decision Tree w/ PCA	0.965567	0.964321	0.96	0.96
Random Forest (tuned) w/ PCA	0.990683	0.990586	0.98	1
SVM (scaled) w/ PCA	0.981546	0.98117	0.98	1
Perceptron w/ PCA	0.859431	0.848486	0.82	0.89
XGBoost (tuned) w/ PCA	0.977135	0.976586	0.97	1
Bagging w/ PCA	0.979504	0.980303	0.98	1
GBM w/ PCA	0.885344	0.879582	0.88	0.94

Logistic regression

```
[[1017  203]
 [ 315  887]]
              precision    recall  f1-score   support

           0       0.76      0.83      0.80      1220
           1       0.81      0.74      0.77      1202

    accuracy                           0.79      2422
   macro avg       0.79      0.79      0.79      2422
weighted avg       0.79      0.79      0.79      2422
```

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay object at 0x0000020D69C873C8>
```

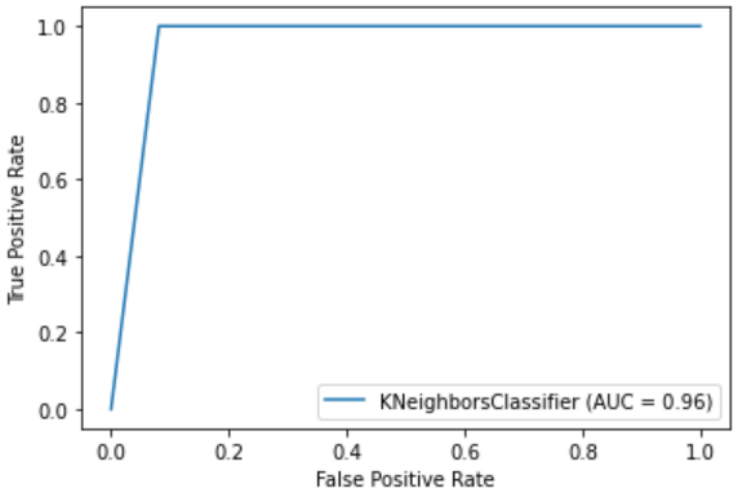


Logistic regression, due to its inflexibility, is poor at predicting fraudulent providers, but does have very strong interpretability as we can see which predictors are statistically significant.

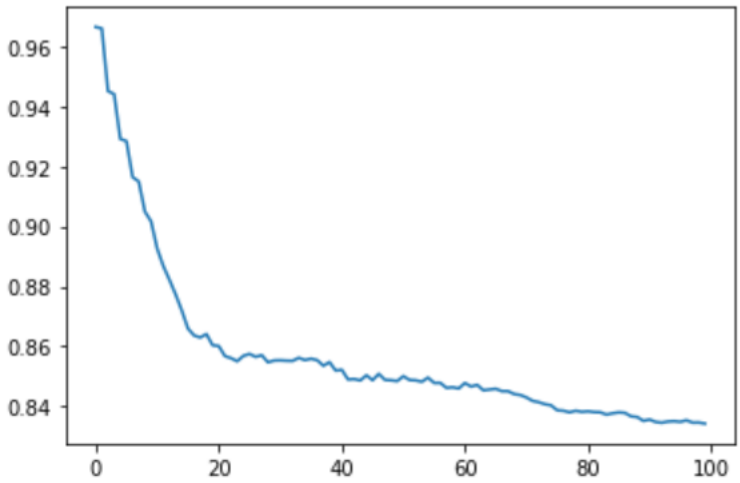
KNN (with scaled exogenous variables)

```
[[1121   99]
 [   0 1202]]
              precision    recall  f1-score   support

           0       1.00      0.92      0.96      1220
           1       0.92      1.00      0.96      1202

    accuracy                           0.96      2422
   macro avg       0.96      0.96      0.96      2422
weighted avg       0.96      0.96      0.96      2422
```

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay object at 0x0000020D694F1188>
```



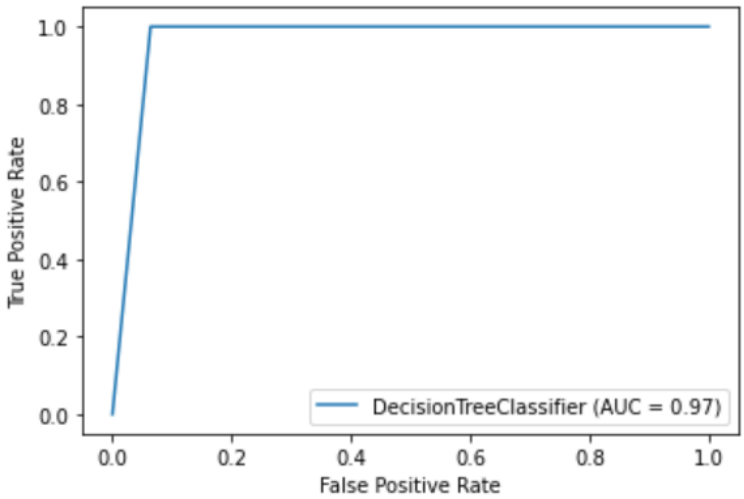
The KNN model is set to a k value of 1. This value was chosen as a result of a grid search: at k = 1, the value of f1 is maximised:



Decision Tree

```
[[1142   78]
 [   0 1202]]
              precision    recall  f1-score   support

           0       1.00      0.94      0.97      1220
           1       0.94      1.00      0.97      1202

    accuracy                           0.97      2422
   macro avg       0.97      0.97      0.97      2422
weighted avg       0.97      0.97      0.97      2422
```

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay object at 0x0000020D18D2BA08>
```

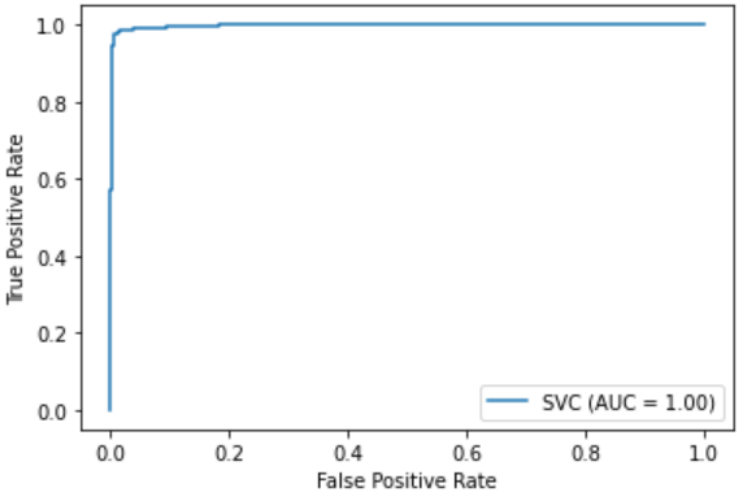


Although the decision tree models are not as accurate as a random forest model, they may have improved in accuracy had the trees been pruned.

Support Vector Machines (with scaled exogenous variables)

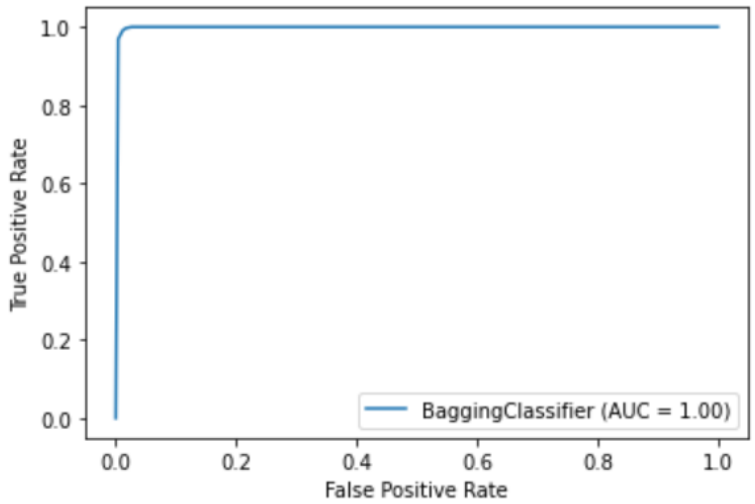
```
[[1212    8]
 [  35 1167]]
              precision    recall  f1-score   support

           0       0.97      0.99      0.98      1220
           1       0.99      0.97      0.98      1202

    accuracy                           0.98      2422
   macro avg       0.98      0.98      0.98      2422
weighted avg       0.98      0.98      0.98      2422
```

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay object at 0x0000020D6BB0DE88>
```



Bagging

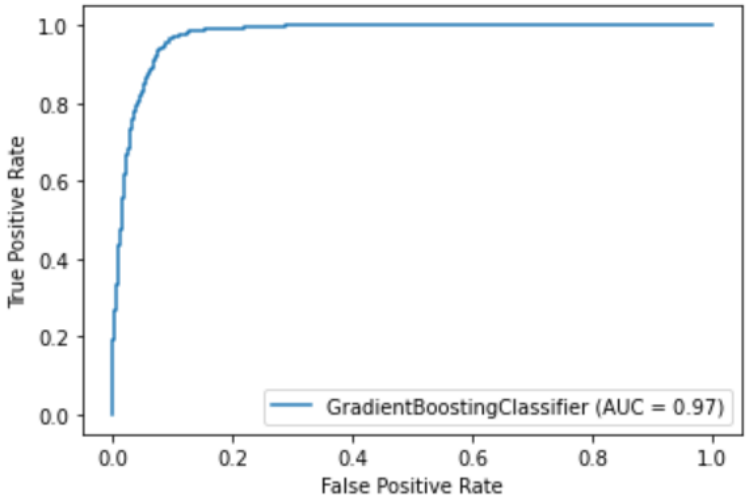
```
[[1173   47]
 [   0 1202]]
              precision    recall  f1-score   support

           0       1.00      0.96      0.98      1220
           1       0.96      1.00      0.98      1202

    accuracy                           0.98      2422
   macro avg       0.98      0.98      0.98      2422
weighted avg       0.98      0.98      0.98      2422

<sklearn.metrics._plot.roc_curve.RocCurveDisplay object at 0x0000020D18BFAA88>
```



Gradient Boosted Models

```
[[1083  137]
 [  30 1172]]
              precision    recall  f1-score   support

           0       0.97      0.89      0.93      1220
           1       0.90      0.98      0.93      1202

    accuracy                           0.93      2422
   macro avg       0.93      0.93      0.93      2422
weighted avg       0.93      0.93      0.93      2422
```

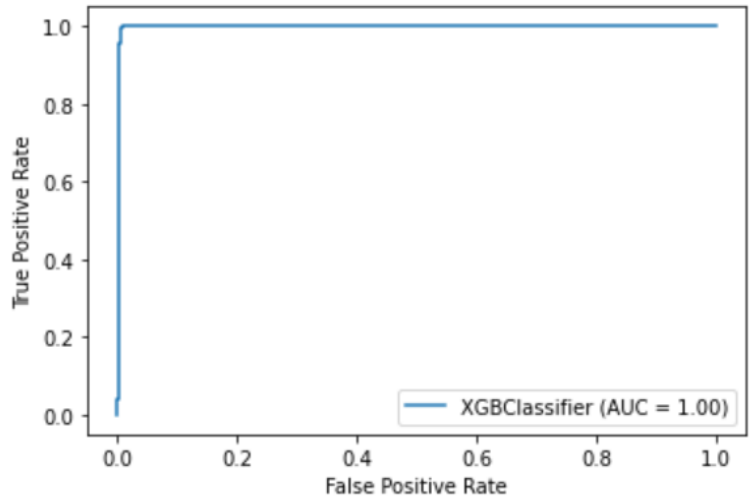
```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay object at 0x0000020D6BEEAF08>
```



XGBoost

```
[[1178   42]
 [   0 1202]]
              precision    recall  f1-score   support

           0       1.00      0.97      0.98      1220
           1       0.97      1.00      0.98      1202

    accuracy                           0.98      2422
   macro avg       0.98      0.98      0.98      2422
weighted avg       0.98      0.98      0.98      2422
```

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay object at 0x0000020D6BC197C8>
```



Additional Charts

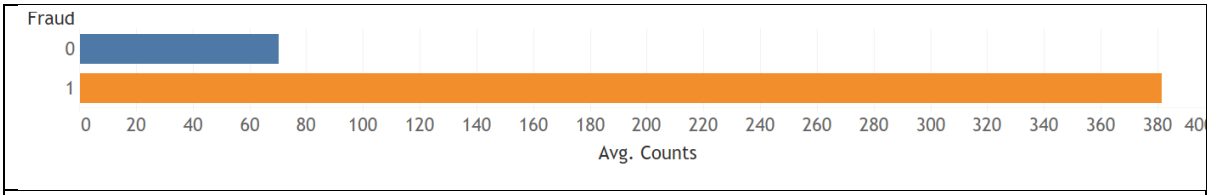


Figure A2.1: Average number of claims by fraud classification

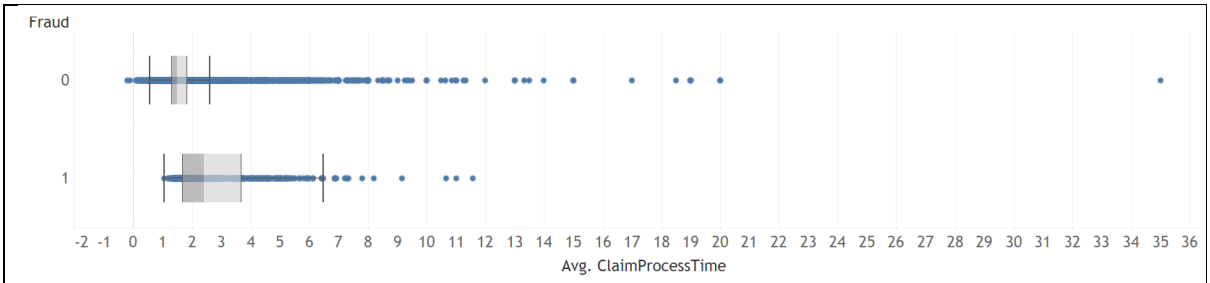


Figure A2.2: Average claim processing time by fraud classification

Figure A2.3: Average reimbursement by chronic condition

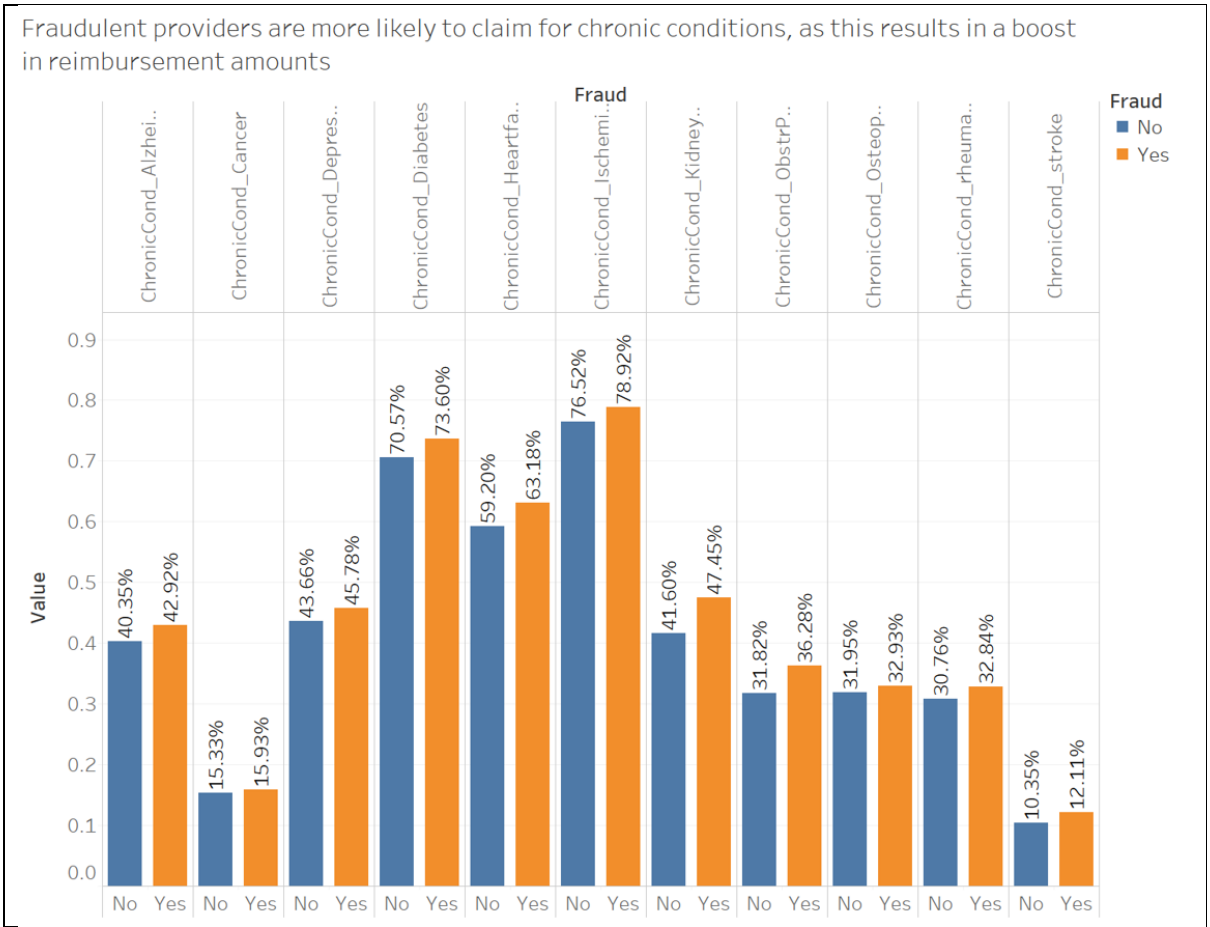


Figure A2.4: % of claims for beneficiaries with chronic health conditions, by fraud classification

	InscClaimAmtReimbursed	DeductibleAmtPaid	ClmProcedureCode_1	ClmProcedureCode_2	ClmProcedureCode_3	ClmProcedureCode_
count	436254.000000	435580.000000	17239.000000	4006.000000	700.000000	95.0000
mean	964.640760	74.838562	5904.486571	4099.813030	4171.528571	4218.7052
std	3738.459109	267.948116	3041.036732	2040.320242	2350.632691	1831.9079
min	0.000000	0.000000	11.000000	42.000000	42.000000	42.0000
25%	40.000000	0.000000	3893.000000	2724.000000	2724.000000	2762.0000
50%	80.000000	0.000000	5381.000000	4019.000000	4019.000000	4019.0000
75%	300.000000	0.000000	8669.000000	4439.000000	5168.000000	4889.5000
max	125000.000000	1068.000000	9999.000000	9999.000000	9982.000000	9986.0000

Figure A2.5: A sample from the descriptive statistics

Technical glossary

- The **accuracy** of a machine learning model is the number of correct predictions as a percentage of the total predictions made.
- The **f1 score** of a machine learning model is the harmonic mean of the precision and recall scores.
    - Precision is equal to the total number of true positive predictions divided by the total positive predictions
    - Recall, also known as the true positive rate, is equal to the total number of true positive predictions divided by the total actual true values.

- The **ROC curve** is a graphical plot of the true positive rate against the false positive rate.
    - The ideal ROC curve should be as close to the top-right corner as possible.
    - **AUC** is the area underneath the ROC curve. AUC can lie from 0 to 1.