Statistical analysis

There are two python files...Network_locum.py and nl_xgb.py

In Network_locum.py, the 3 data sets (sessions, ccgs and practices) are read as pandas dataframes, followed by data selection and merging.

nl_xgb.py is the machine learning code for predictions. I have focused on probability of a job being completed but it can be modified to predict probability of a job not being completed

-To calculate the fill rate, I used

```
data['fill_rate'] = (data['n_completed_jobs']/data['n_posted_jobs'])*100 #calculate fill rate
```

To compute n_completed_jobs and n_posted_jobs, I did the following

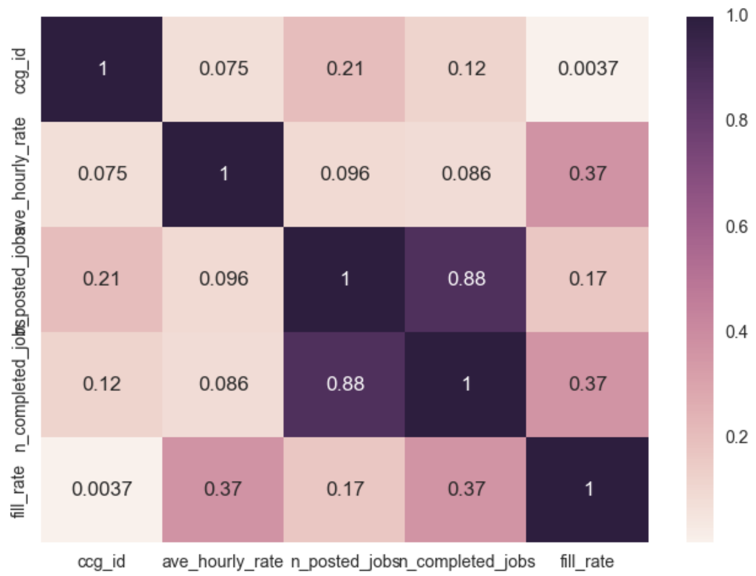
- n_completed_jobs is the count of posted jobs (posted_datetime) for each ccg. I used ccg rather than practice_id because it was quicker to do due to the expected quick turn around

- n_posted_jobs is the number of jobs per ccg_id for which status is completed. It makes more sense to focus on 'completed' status as I presume that is a better focus for Network Locum.

Question 1: Does a higher fill rate result in a higher hourly rate?

This question seeks to know whether there is a strong positive relationship between the two variables. The correlation heatmap below shows that the value of the correlation is positive 0.37. It is therefore only weakly correlated.

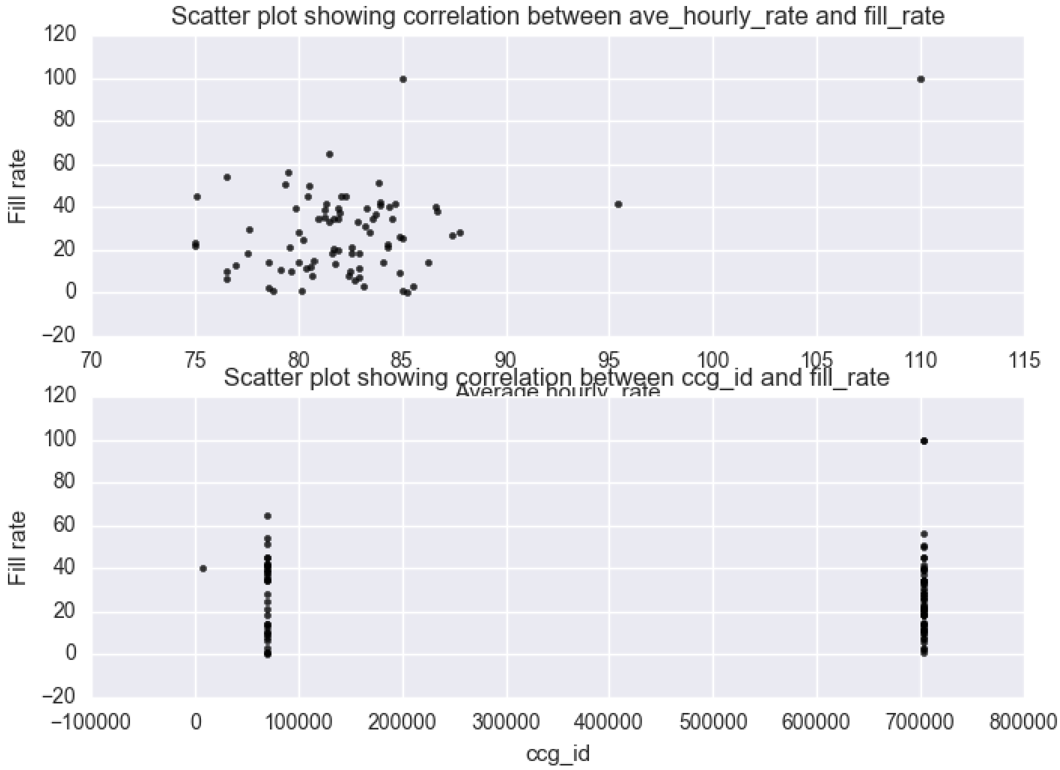
Figure 1:



The correlation can also be seen using a scatter plot of the two variables as shown below in figure 2. The first subplot shows the strength of the correlation.

Further analysis using Pearson's r coefficient confirms this conclusion

Figure 2: **corr_plots.png**



Question 2: What is the variability of this relationship across CCGs?

data['fill_rate'].describe()

count 82.000000

mean 27.693015

std 19.163384

min 0.139925

25% 13.034542

50% 26.083009

75% 39.485553

max 100.000000

The standard deviation is a measure that is used to quantify the amount of variation or dispersion of a set of data values. The standard deviation and the mean for the fill rate are 19.163384 and 27.693015 respectively. This shows the fill rate is very dispersed and not clustered around the fill rate mean.

I would show the frequency distribution using a normal distribution curve if I had more time.

Machine learning model for prediction

For the prediction task, I used extreme gradient boosting (xgb) algorithm. Xgb is a boosting technique that uses an ensemble of weak prediction models. It builds the model in a stage-wise fashion and it generalizes them by allowing optimization of a loss function.

I split the data produced for the model into train-validate-test sets (details in the code). Below is a snippet of the optimization performed by the algorithm and the eval mlogloss using the validation set.

Each dataset was shuffled before analysis to introduce some random sampling into the model building.

[969] train-mlogloss:0.093844 eval-mlogloss:0.154051

[970] train-mlogloss:0.093815 eval-mlogloss:0.154056

[971] train-mlogloss:0.093787 eval-mlogloss:0.154074

[972] train-mlogloss:0.093740 eval-mlogloss:0.154056

[973] train-mlogloss:0.093710 eval-mlogloss:0.154065

[974] train-mlogloss:0.093663 eval-mlogloss:0.154059

[975] train-mlogloss:0.093604 eval-mlogloss:0.154034

Stopping. Best iteration:[875] train-mlogloss:0.096909 eval-mlogloss:0.153851

The model was then used to predict the status of the held out test set. The classification performance was analysed using the classification report function from sklearn

		precision	recall	f1-score	support
Class 0	0.0	0.98	0.93	0.95	183
Class 1	1.0	0.94	0.98	0.96	204
avg / total		0.96	0.96	0.96	387

The precision and recall for both classes are pretty good. The algorithm identifies true positive for class 0 98% of the time and 94% of the time for class 1. Recall is a measure of how many of each class was correctly identified compared to the total available.

To measure how good the predictions are, I used Matthew's correlation coefficient. It takes into account true and false positives and negatives and is generally regarded as a balanced measure which can be used even if the classes are of very different sizes. So I have used it because it is very robust when the amount of data belonging to the classes is different

The mcc for these predictions : 0.91266025711

Isaac Alabi