Fraud detection project

Working with imbalance classification

Lu Luo Oct 2020

Task and data

Data

- The data consists of genuine and fraudulent transactions over 2 days.
- Highly imbalanced: only 0.172% of the transactions are fraudulent.
- There are 28 anonymous features in the dataset, all standardised with a mean of 0 and standard deviation of 1.
- Additional features include Time (seconds since the first transaction) and Amount (transaction amount).

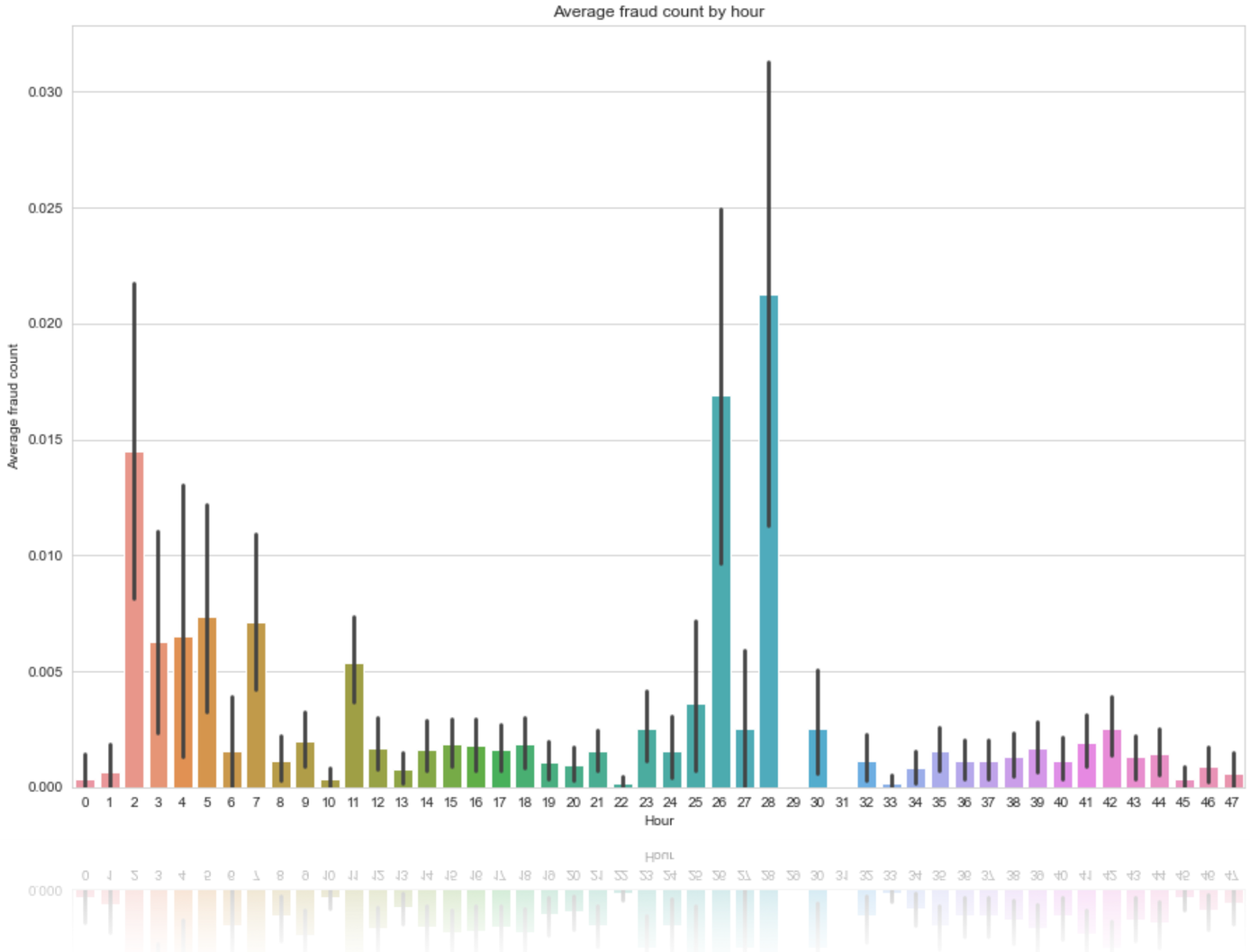
Task and challenges

- Build a ML model to identify fraudulent transactions from normal transactions
- The imbalanced data is challenging in terms of model training. Traditional machine learning techniques do not like extremely imbalanced data. This could potentially be resolved using over sampling and under sampling.
- Scoring metric for training and evaluating needs to be carefully considered. (Predicting all data points as normal will give you a 99.82% accuracy)

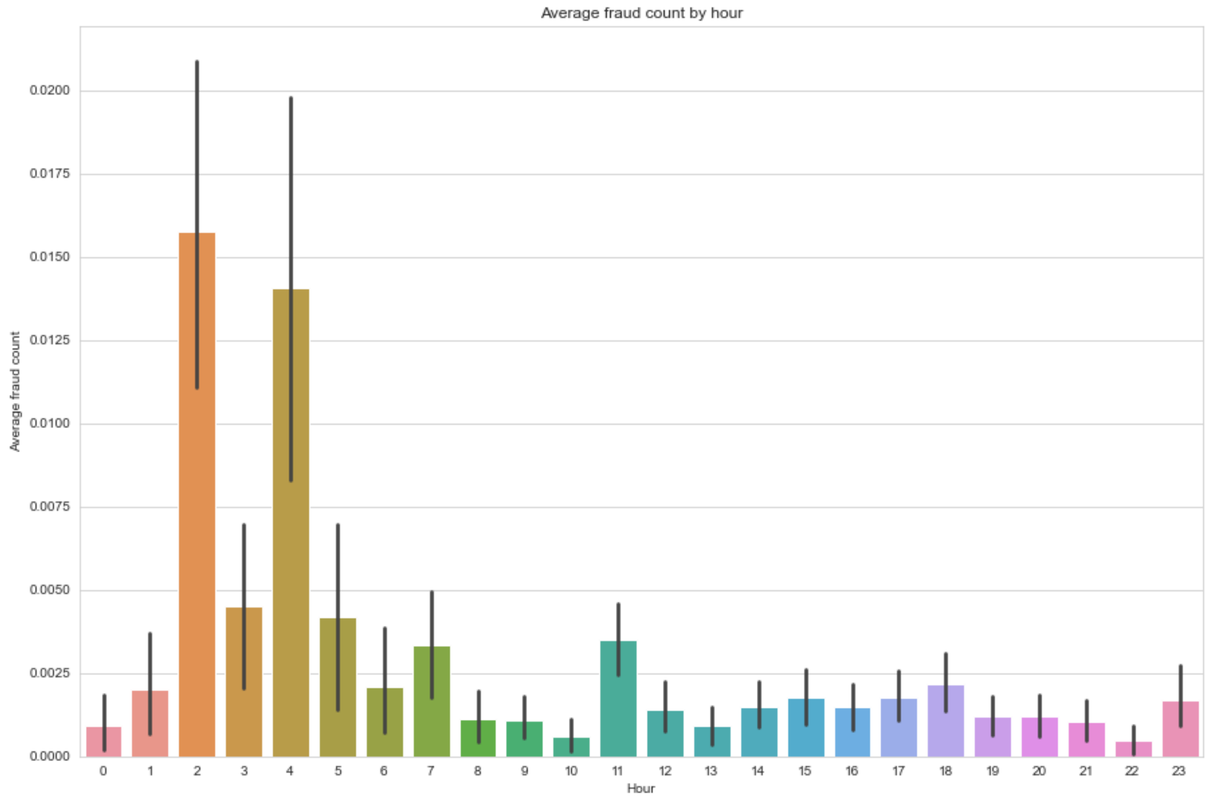
First look

Time

- There seems to be a peak around the first few hours since transaction 0

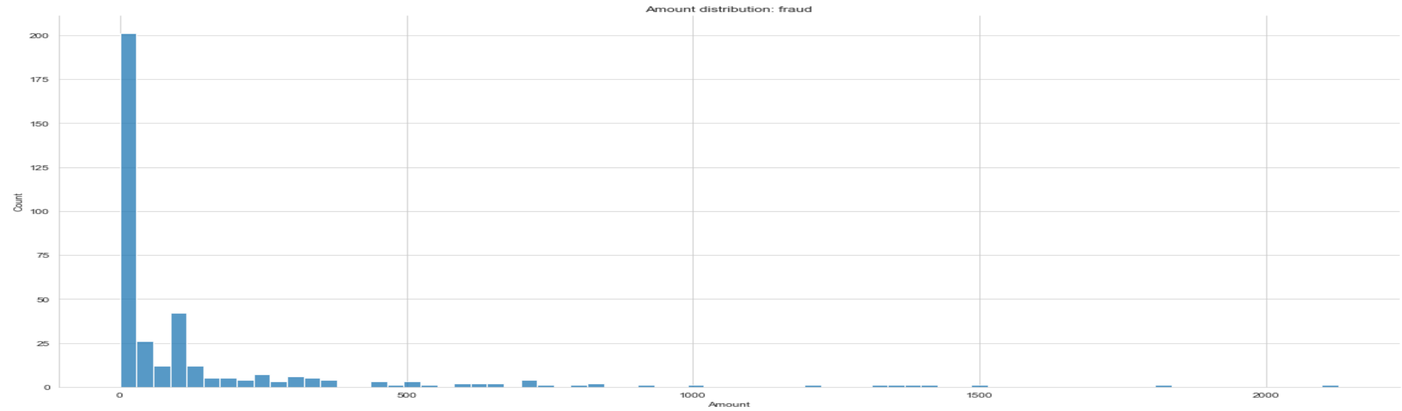


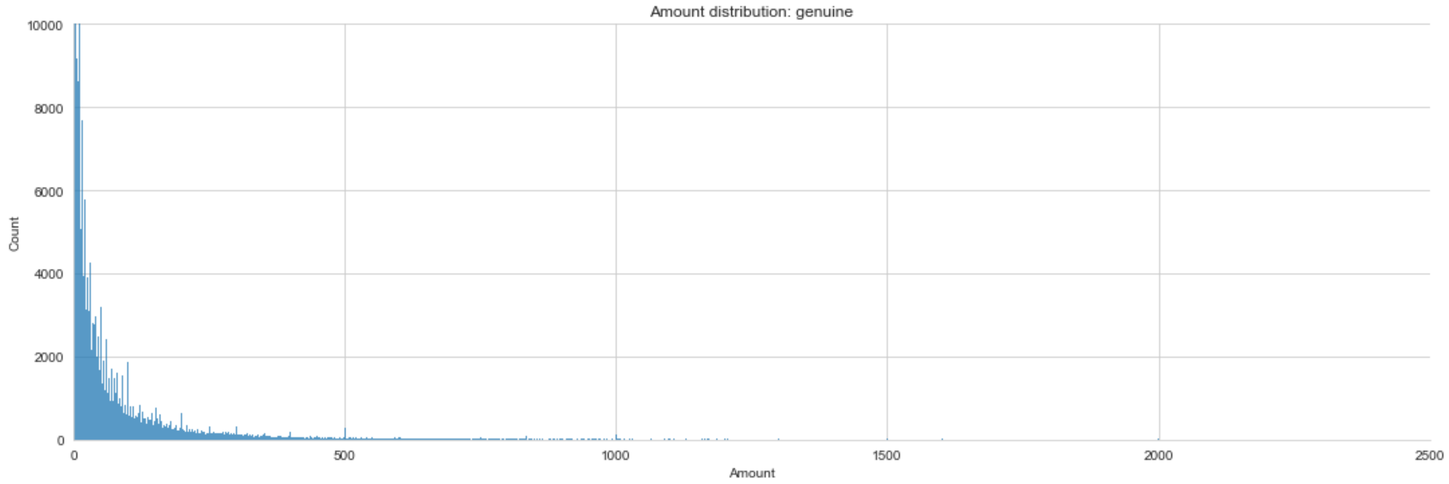
Time

- I transformed the time seconds into number of hours since hour 0.
- 

Amount

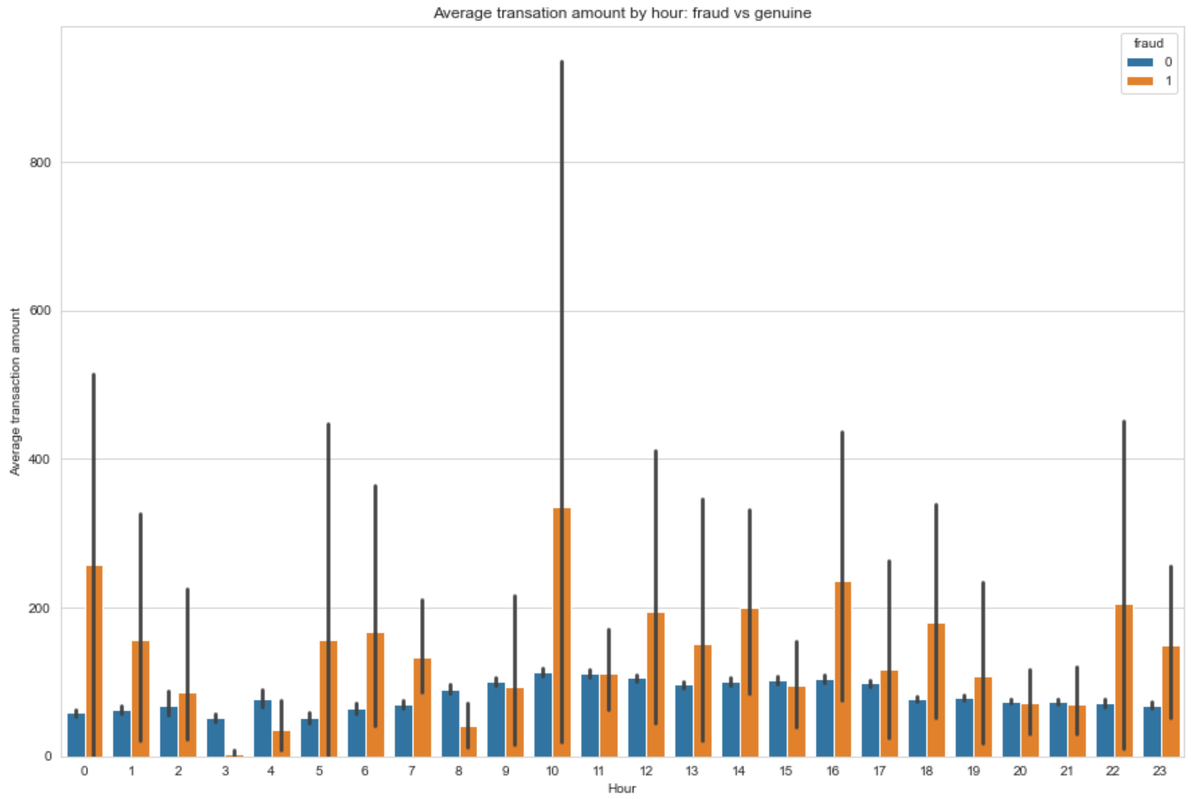
- The distributions are similar: a vast majority low value transactions (fraudulent have lower median and shorter tail). However there is a cluster around £100 for fraudulent transactions.





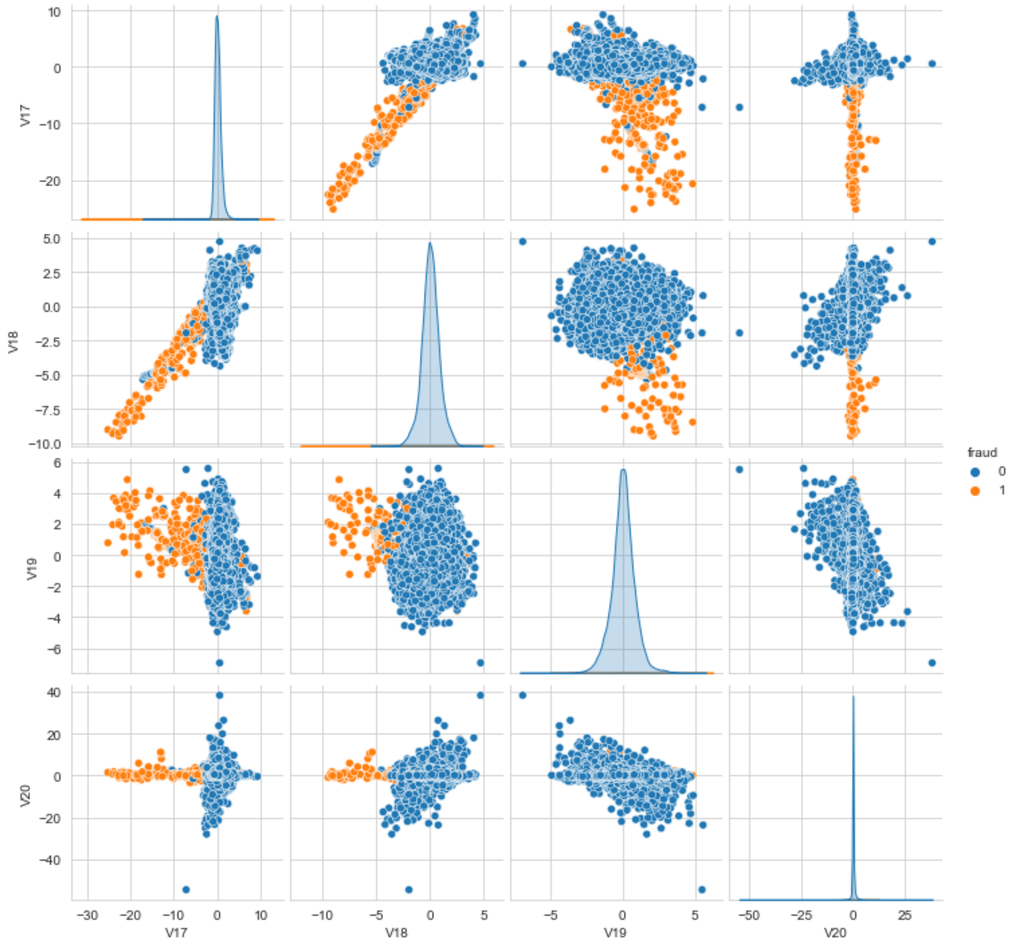
Amount

- The mean is higher for fraudulent transactions, with higher variance too.

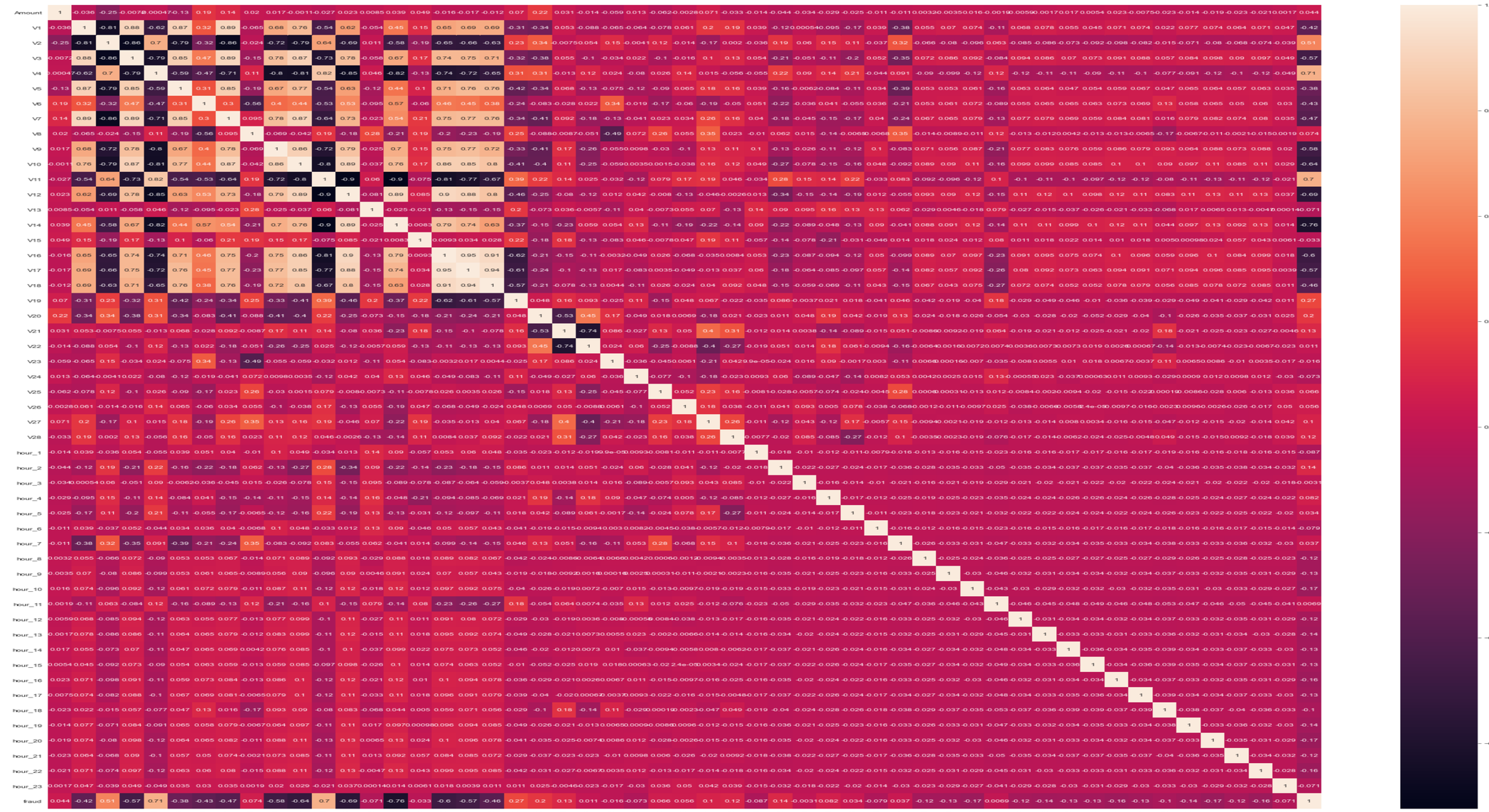


V1-V28

- V1-V28 give us some really good information. For some features (combination of features) the separation is very good.



Correlation heatmap



Quasi-separation?

- When running logistic regression I came across this warning:

```
Possibly complete quasi-separation: A fraction 0.55 of observations can be
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.
```

- I still do not understand how to deal with this problem. On one side, it is a great thing, as it shows some features can produce perfect predictions, which is exactly what we want.
- I am not sure whether this because of the data imbalance.
- Looking at the heat map, there is suggestions of certain features being very predictive.

Preprocessing steps

- I conclude the first look with the following preprocessing steps:
  - Normalise amount
  - Transform time into hours from hour 0
  - We have later found out that hours/time is not predictive in fraud. Perhaps there happens to be peaks just after hour 0 in the 2 days sampled. That same pattern didn’t show in the test set.
  - Note outliers: because this is a prediction task, not an analysis task, I decided not to delete the outliers because I want to build a model that can deal with outliers, particularly if the outliers are correlated with fraud cases.

Model building

Model choices

- Because this is a classification problem, I short listed these models to try:
    - Basic logistic regression
    - KNN
    - Random forest
    - SVC

Sampling choices

- For resolving issues that could arise from imbalance, I tried combinations of these over and under sampling methods:
    - SMOTE: Synthetic Minority Oversampling Technique
    - BorderlineSMOTE: SMOTE that samples more around the class borderline
    - Randomised under sampling
- I chose NOT to resample the validation set, because I want to make sure the validation set is as close to the test set as possible.
- With the limited time I got to play around with resampling, they actually produced worse results, compared to no resampling, particularly under sampling completely collapsed precision score on validation.

Metrics

- Because I was working with severely imbalanced data, I choose the metrics that only deal with precision (FP) and recall (FN), fraud being positive.
- The area under precision and recall curve is used.
- Also I devised a metric that takes into account the cost of FP (client annoyed at the hassle, losing business), and FN (fraudulent transaction not detected, losing money)

Random search CV

- For KNN, random forest, SVM and XGBoost, I used Sklearn’s RandomSearchCV to cross validate and tune hyper parameters.
- XGBoost produced the best result, 1.00 train, 0.84 test.
- Questions on how to narrow the train and validation loss gap, and how to improve random search.

What’s next?

What's next

- The customised cost based metric is quite interesting, and it would be good to work out an interpretation of the score in money.
- Experiment on grid search and see if it improves the train/validation gap
- Experiment on early stopping
- Experiment on NN for fraud detection
- Dive deep into the meaning of quasi-separation; why is it; and how to deal with it.