# Case study findings:

## TASK 1

‘task.app.data.csv’ dataset (dataset 0) was left joined with ‘task.cb1.csv’ (dataset 1) on ‘key2’. The resulting dataset was left joined with ‘task.cb2.csv’ (dataset 2) on ‘key1’. I have used left joins, because only the dataset 0 has the response variable. Thus, I have added some extra data to the dataset 0 from the other 2 datasets.

Here are the sizes of the datasets. They are very diferent – dataset 0 is much smaller than dataset 1 and 2.

1) dataset 0

column count: 5

row count: 798

element count: 3990

2) dataset 1

column count: 169

row count: 14571

element count: 2462499

3) dataset 2

column count: 37

row count: 10137

element count: 375069

After the joins the final dataset has the following size:

dataset\_full - before cleaning NAs

column count: 209

row count: 798

element count: 166782

I remove some columns and rows that has too many missing values. For columns there should be at least 20% non-missing values, and for rows – 5% (later I will apply more aggressive filtering). And I get the following dataset:

dataset\_full\_clean - after some columns and rows with many missing values are removed

column count: 60

row count: 772

element count: 46320

After this clean-up, there are still 17 473 Nulls in the dataset (around 1/3 of total number of elements/cells).

*Conclusion about data integrity:*

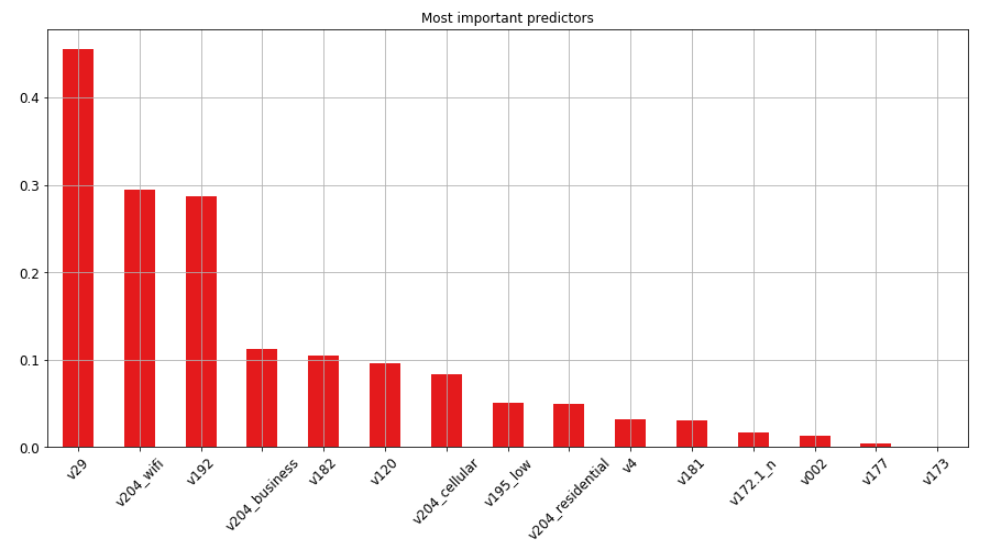
Only small portions of dataset 1 and 2 are used. Even for those parts that are used, there is a significant amount of missing values (NaNs or Nulls). This can really affect the performance of the model. With such data structure it is very important to have a good strategy of dealing with missing data.

## TASK 2

To determine the strongest predictors of target variable I apply 2 approaches:

1. To look at the correlations between each predictor and the target variable. It is a less robust approach, but it is still helpful to do it as an additional check.
2. Response variable is a binary variable (1 or 0). So, I have chosen Logistic Regression to estimate the importance of various factors. It is not the best ML-algorithm to get a high prediction accuracy, but it is very suitable for the task to determine most important factors. It provides coefficients for each predictor measuring their impacts on the target.
   1. To deal with multicollinearity and overfitting I have used regularization (L1-norm) that adds penalty on too high coefficients.
   2. Also, I have performed some dimensionality reduction, by removing factors that have essentially the same correllation with the target.
   3. To improve the performance of the model, I have optimized its parameters (such as regularization strength) with cross-validation.
   4. I have normalized all predictors before using Regression, thus the coefficients are comparable.

The graph below shows the most important factors:



The most important predictors are v29, v192, v204, v195\_Low, v182, v120, v002, v5. These findings are consistent with the correlations between predictors and the target (see Table 1 in Appendix).

Because of similarity/redundancy I have excluded some factors earlier. Most importantly, I have excluded v191 (which is identical to v192). But, I could say that v191 is also an important regressor.

## TASK 3

I have used Random Forest Classifier model to do the predictions. It is a very powerful model both theoretically and empirically. It works well against ovefitting. At the same time, it has only few parameters that are not too sensitive too changes. So, it is relatively easy to calibrate them. Also, since it is based on a decision tree, there is no much need to create additional variables (like polynomials, interections and different powers of existing factors), thus much less time is needed on feature engineering and there is a smaller risk that you can miss something.

I have optimized all the parameters for the model using Cross-Validation and Grid-Search. The optimized model has the following parameters: max\_depth=26 (max tree depth), min\_samples\_split=10 (minimum number of observation in a split node), n\_estimators=50 (number of trees in the forest). The other parameters has their default values, like max\_features=’auto’=’sqrt’, which says that the maximum number of features to consider at split should be a square root of the total number of features.

### Performance of the model

5 fold cross-validation on the train-set gives the following results:

accuracies = [ 0.81451613 0.81451613 0.81451613 0.81300813 0.8442623 ]

mean accuracy = 0.820163762452

If train the model on the whole train set and I get the following accuracy when predicting target on train set - 0.8995. When I train on train-set and predict on test-set, then the accuracy is - 0.8516.

Thus, the model still could be optimized, as it is overfitting a bit. But overall, the results are consistent.

### Comparisson with other models

1. Firstly, it is good to see if the model performes better than simply predicting the majority class for all the predictions (in this case it implies simply predicting that each observation has a target of 0).

1-train[target].mean() = 0.8071

1-test[target].mean() = 0.8516

So, Random Forest model performs slightly better than majority class prediction.

1. Let’s compare with the scores of Logistic Regression from Task 2.

Logistic Regression gives the following result in 5 fold cross-validation on the train-set:

accuracies = [ 0.80645161 0.81451613 0.80645161 0.81300813 0.83606557]

mean accuracy = 0.815298611738

If I train the model on the whole train set, then I get the following accuracy when predicting target on train set - 0.8233. When I train on train-set and predict on test-set, then the accuracy is - 0.8516.

Thus, Random Forest model performs slightly better than Logistic Regression as well.

### Conclusions

Thus, Random Forest model gives a higher accuracy than a majority class prediction model would give. Also, it gives a higher accuracy than the logistic regression in the previous task. However, I do not see a significant improvement, thus a higher focus should be put on feature selection and dealing with missing values. Also, it would be interesting to try some other models like XGboost or some SVM. It is a bit dissapointing the the model does not perform much better than majority class prediction. However, if we would focus not just on accuracy, but on other aspects like precision and recall, than perhaps it would be more useful.

Note:Most of the time was actually spent on data preparation – I have

## Appendix

### Table 1

Correlation between variables and response. We are interested in magnitude, so I have used an absolute value and sorted them from the largest to the smallest.

response 1.000000

v192 0.194025

v191 0.194025

v29 0.172579

v204\_wifi 0.148913

v204\_business 0.105443

v204\_cellular 0.097367

v204\_residential 0.077241

v002 0.072353

v182 0.066963

v120 0.051613

v177 0.048428

v172.1\_n 0.045414

v174 0.044762

v173 0.042317

v195\_low 0.034318

v197 0.034318

v196 0.034318

v172.1\_y 0.032629

v201\_moderate risk 0.029653

v200 0.029653

v181 0.026332

v204\_mobile 0.024923

v178\_certified 0.024196

v180 0.022540

v193\_yes 0.022007

v178\_validdomain 0.021854

v193\_not sure 0.021854

v183 0.021228

v184\_lower fraud risk 0.020346

...

v184\_data entry review 0.012306

v186\_fraud score 601 to 799 0.012306

v193\_no 0.012306

v178\_emailinexistent 0.012306

v203\_nan 0.010840

v203\_no 0.010840

v202\_nan 0.010840

v202\_good 0.010840

v201\_nan 0.010840

v186\_nan 0.010840

v172.1\_nan 0.010840

v184\_nan 0.010840

v199\_moderate 0.010840

v195\_nan 0.010840

v194\_nan 0.010840

v194\_yes 0.010840

v178\_nan 0.010840

v193\_nan 0.010840

v199\_nan 0.010840

v172.1\_p 0.010143

v172.1\_u 0.009589

v4 0.007668

v186\_fraud score 1 to 100 0.007036

v201\_moderate by proxy reputation and country code 0.006696

v204\_nan 0.005322

v204\_wired 0.002650

v184\_moderate fraud risk 0.002095

v186\_fraud score 301 to 600 0.002095

v123 0.001951

v198 NaN