1 How do the data look like?

I started with introductory analysis, presented in `QuickGlance.ipynb`. The data set doesn't need any tidying, but - it's unbalanced. Fraudulent cases are not even 0.2% of all transactions. To find features distinguishing Fraudulent and non-fraudulent cases I made two data frames of equal size (so randomly under-sampled) non-fraudulent data frame. Plotting the V-features of both distributions one can conclude that:

- features are quite symmetrical, there is no need to apply any transform
- features V13, V15, V24, V25 look similar for both kinds of data. Removing them may increase accuracy of predictive models - losing not so much information we can decrease the dimensionality of the problem. Both approaches were tested, removing these features leads to *slightly* better results (0.01 in recall using binary classifiers and about 1-3$ of mean absolute error in regression)

I decided to drop Time feature as it should not be crucial in the real situation (data were collected for two days only, moreover the distributions look similarly). What may be worth noticing, several features have higher correlation for fraudulent data than for genuine.

2 Two-class predictive model

2.1 Metric

I decided to build supervised model classifying transactions as fraudulent or non-fraudulent basing on V-features of under-sampled data only. The main metric I used was recall - it's better to classify a genuine transaction as fraudulent than to do otherwise. Moreover in such unbalanced data set the most popular metric - accuracy - can't be used. Any predictive model would prefer to say that transaction is non-fraudulent in every case and get about 100% accuracy. However, I found the best hyper-parameters for the predictive model using the under-sampled, balanced set. Then accuracy was not so meaningless and was used, with F1 score as additional metrics.

2.2 Used models

I used three models: SVM (good to use in high-dimensional spaces, able to handle non-linear regression as well), a decision tree (fast, but can over-fit) and logistic regression (simple, fast and less sensitive to over-fitting than decision tree). Using 4-fold cross-validation the hyper-parameters of all of theme were tuned (file `BinaryClassifiers.ipynb`). Eventually, I decided to use SVM with cubic kernel because of the highest recall value.

3 Regression to predict amount

To do regression I employed XGBoost algorithm. It's fast (so I could tune many hyper-parameters) and can be extremely accurate, as show Kaggle competitions. The hyper-parameters optimization (`Regression.ipynb`) was also done with 4-fold cross-validation. The model was fed with fraudulent data only - it was working under the assumption that the transaction was fraudulent. Metrics I looked at were commonly used in regression problems mean absolute error and mean squared error. It was due to wide range of fraud amounts - from 0 to 1200. I also tried another approach - applying $x \to \log(1 + x)$ function to amount to obtain more symmetric distribution. As experiment has shown, the latter approach is less accurate and was no longer used in model construction. The one classified as optimal gave mean absolute error at about 64 and mean squared error about 17546. Mean of the distribution is about 120 and median about 9, so this doesn't seem to be an outstanding result.

4 Full model

Full model was implemented as a class (`Model.ipynb`) and uses two algorithms described above - tuned SVM classifier and XGBoost regressor. On balanced set it reaches recall 0.99 and mean absolute error 87.5 (this is greater than the value above - the regressor is given information from binary classifier that prefers to classify genuine transactions as fraudulent). Using 1:100 unbalanced data set (`Model-BigSample.ipynb`) one still gets recall at 0.96 and mean absolute error at about 91.