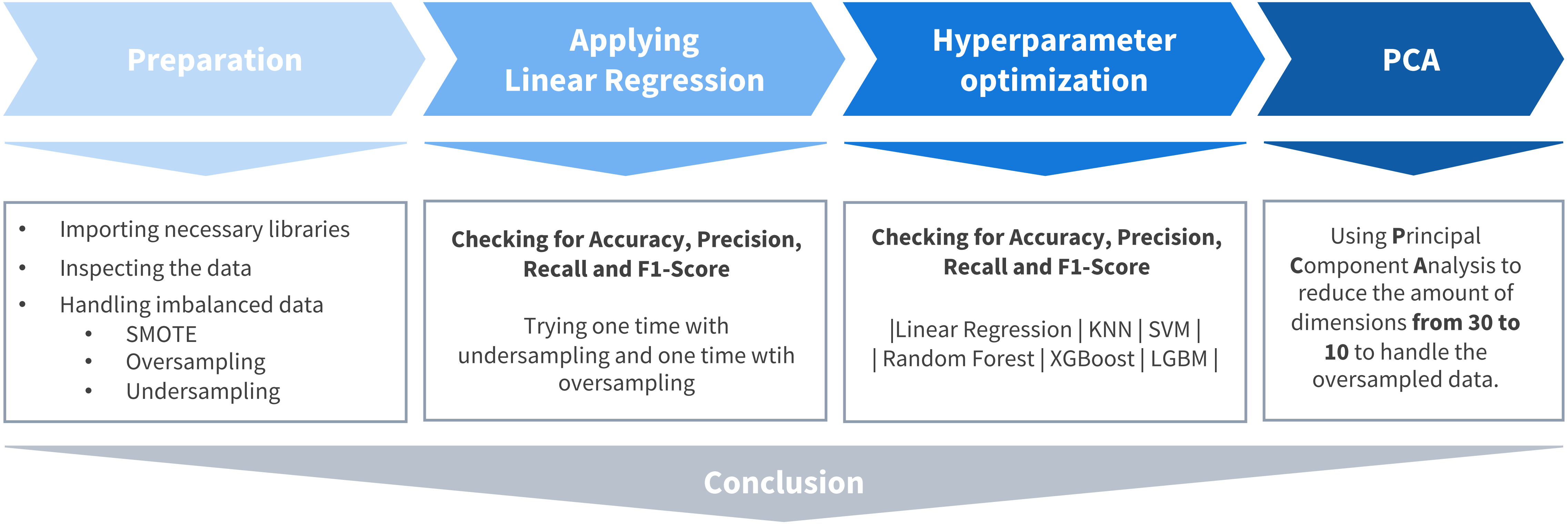
EXECUTIVE SUMMARY – CREDIT CARD FRAUD DETECTION

By using different machine learning models on a dataset with credit card transactions, our group tried to figure out which models deliver the best values for the detection of fraudulent transactions.



Preparation

- Importing necessary libraries
- Inspecting the data
- Handling imbalanced data
  - SMOTE
  - Oversampling
  - Undersampling

Applying Linear Regression

Checking for Accuracy, Precision, Recall and F1-Score

Trying one time with undersampling and one time wtih oversampling

Hyperparameter optimization

Checking for Accuracy, Precision, Recall and F1-Score

|Linear Regression | KNN | SVM |
| Random Forest | XGBoost | LGBM |

PCA

Using **P**rincipal **C**omponent **A**nalysis to reduce the amount of dimensions **from 30 to 10** to handle the oversampled data.

Conclusion

Best accuracy: LGBM with undersampling (Accuracy: 0.9963)
Best precision: Random forest with PCA and oversampling (Precision: 0.999050407533)
Best recall: XGboost with undersampling (Recall: 0.964285714286)
Best F1-Score: LGBM with PCA and oversampling (F1-Score: 0.889608397805)
Best CPU time: Logistic Regression with undersampling (CPU time: 690ms)

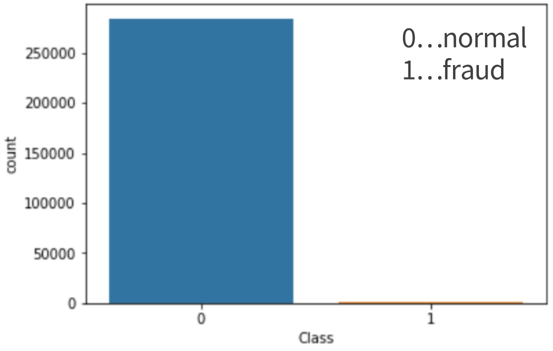
PREPARATION

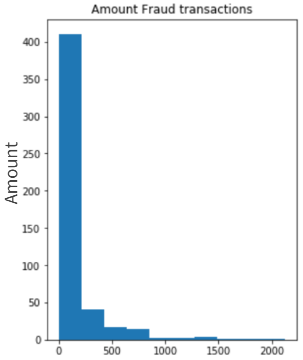
Plotting the data showed that we needed a soltuion to handle imbalanced data. Therefore we applied different methods do deal with this issue.

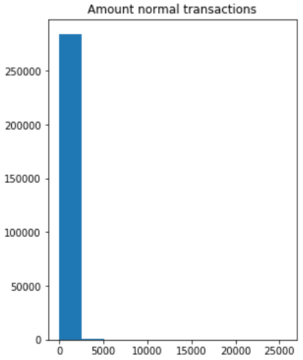
Importing libraries

pandas for data manipulation
matplotlib.pyplot to plot graphs
seaborn for intractve graphs
numpy for linear algebra
datetime to dela with date and time
sklearn.preprocessing for preprocessing the data
sklearn.ensemble for Random forest classifier
sklearn.tree for Decision Tree classifier
sklearn.svm for SVM classification
sklearn.linear_model for LRegression
sklearn.model_selection for splitting, crossvalidation, hyperparameter tuning
sklearn.neighbors for KNN
sklearn.metrics for the confusion matrix
plotly.offline
lightgbm for LGBM
xgboost for XGBoost

Inspection of the data







Handling of imbalanced data

Undersampling

Trying out of different ratios of normal to fraudulent transactions.
80:20 (1968:492) delivers the best results.

Oversampling

Oversampled so that we achieve a ratio of **73:27** (213235:66600).

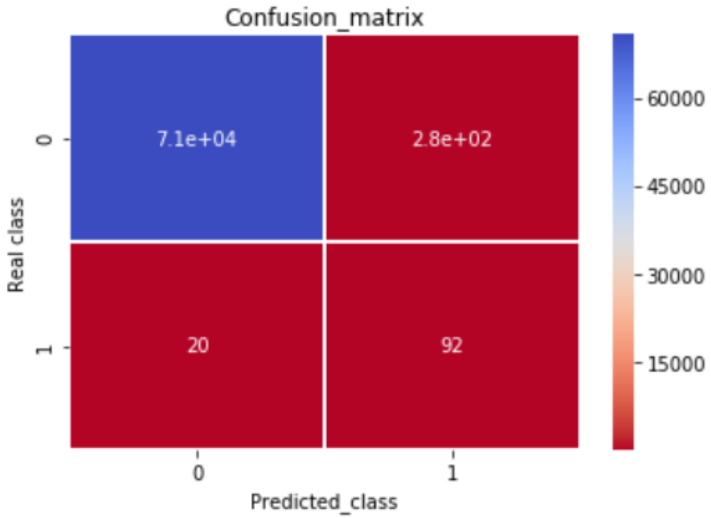
SMOTE

Unfortunaltely the code for Synthetic Minority Over-sampling Technique makes the jupyter notebooks kernel die.

WORKING WITH LINEAR REGRESSION MODELS

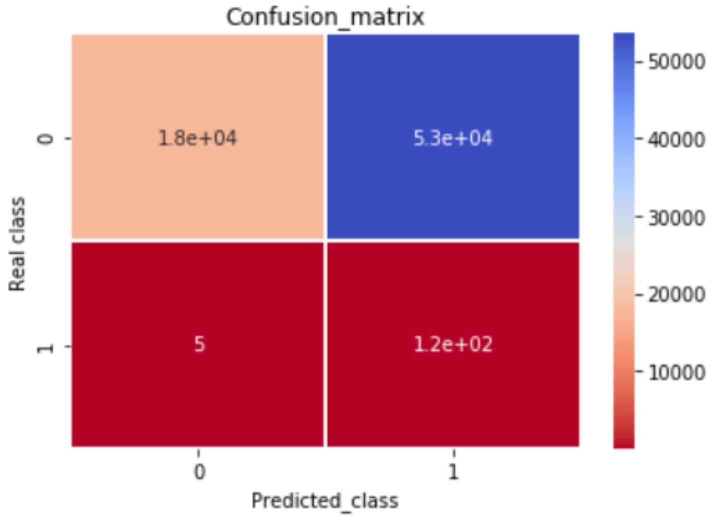
At first we wanted to see how good/bad normal linear regression models work with our credit card transaction data.

Undersampling



```
-----------------------------------------------------------
Accuracy: 0.99571641246
Precision: 0.244031830239
Recall: 0.821428571429
F1-Score: 0.376278118609

-----------------------------------------------------------
```

Oversampling



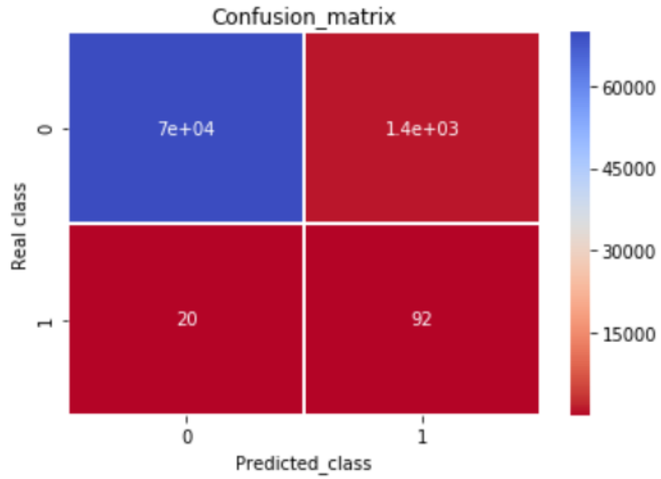
```
-----------------------------------------------------------
Accuracy: 0.249094126569
Precision: 0.00218373212886
Recall: 0.959016393443
F1-Score: 0.00435754189944

-----------------------------------------------------------
```

HYPERPARAMETER OPIMIZATION WITH DIFFERENT MODELS

To get a feeling for the different machine learning techniques we tested out 5 different models in combination with hyperparameter optimization and compared their outcomes.

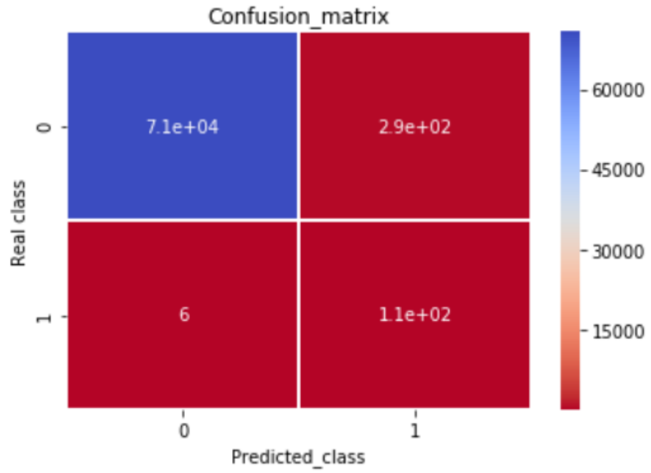
Comparison of different models (undersampled dataset)

SVM



```
----------------------------------------------------------------
Accuracy: 0.980506165557
Precision: 0.0630136986301
Recall: 0.821428571429
F1-Score: 0.117048346056

----------------------------------------------------------------
```

Random Forest



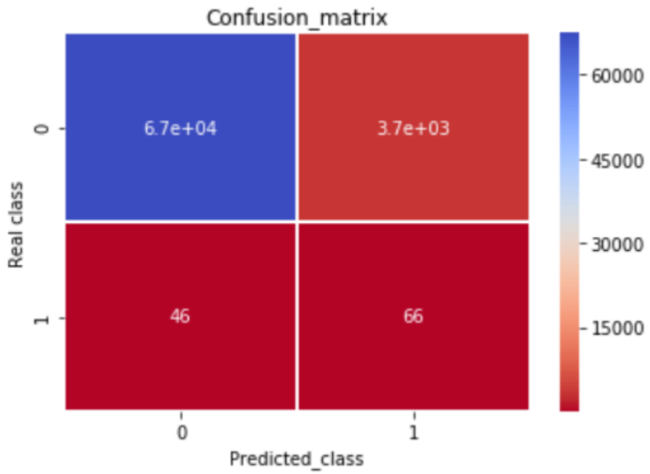
```
----------------------------------------------------------------
Accuracy: 0.995786635207
Precision: 0.265
Recall: 0.946428571429
F1-Score: 0.4140625

----------------------------------------------------------------
```

HYPERPARAMETER OPIMIZATION WITH DIFFERENT MODELS

To get a feeling for the different machine learning techniques we tested out 5 different models in combination with hyperparameter optimization and compared their outcomes.

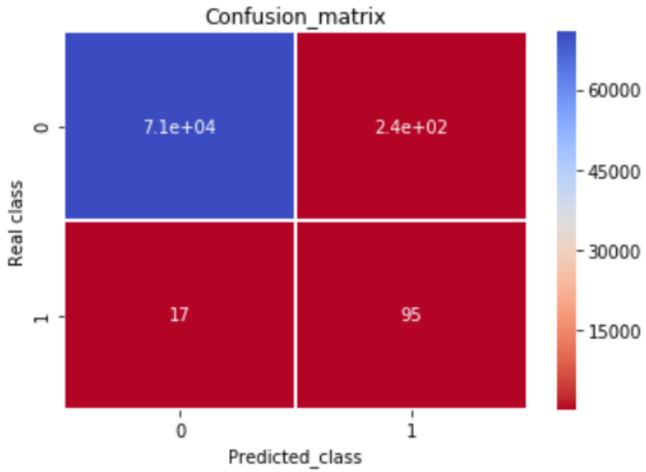
Comparison of different models (undersampled dataset)

KNN



```
------------------------------------------------------------
Accuracy: 0.947993033904
Precision: 0.0177276390008
Recall: 0.589285714286
F1-Score: 0.0344198174707

------------------------------------------------------------
```

LGBM



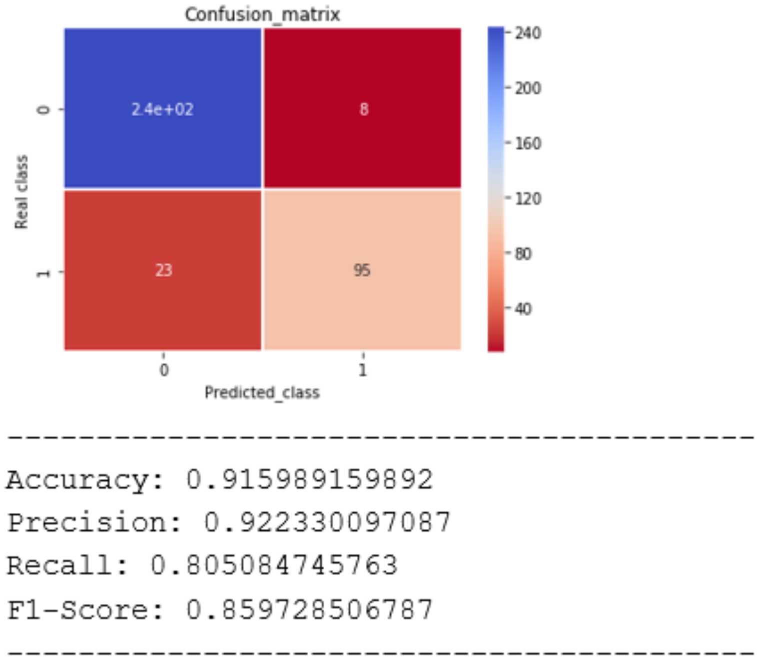
```
------------------------------------------------------------
Accuracy: 0.996348417179
Precision: 0.281065088757
Recall: 0.848214285714
F1-Score: 0.422222222222

------------------------------------------------------------
```

HYPERPARAMETER OPIMIZATION WITH DIFFERENT MODELS

To get a feeling for the different machine learning techniques we tested out 5 different models in combination with hyperparameter optimization and compared their outcomes.

Comparison of different models (undersampled dataset)

XGBoost



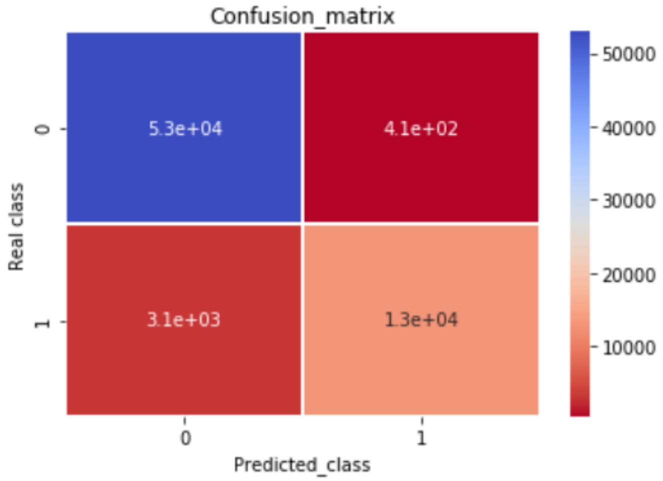
```
------------------------------------------
Accuracy: 0.915989159892
Precision: 0.922330097087
Recall: 0.805084745763
F1-Score: 0.859728506787
------------------------------------------
```

PCA

To make our oversampling work and to have a look at the trade-off between accuracy and the amount of used components, we worked with Principal Component Analysis.

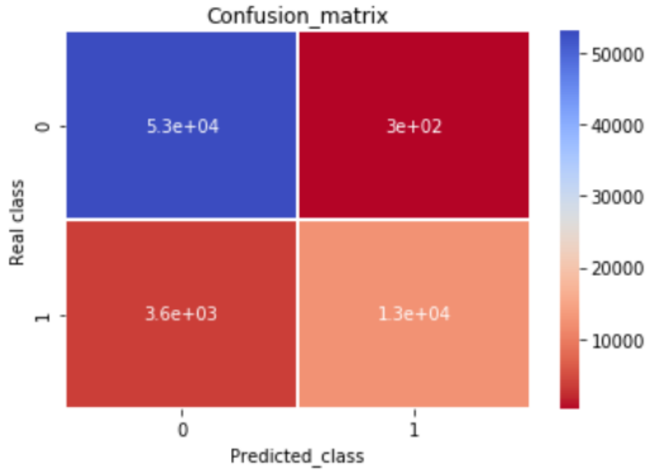
Reducing from 30 components to 10 (with oversampled data)

Logistic Regression



```
------------------------------------------------------------
Accuracy: 0.949260559328
Precision: 0.969372584002
Recall: 0.807180439492
F1-Score: 0.880872766575

------------------------------------------------------------
```

SVM



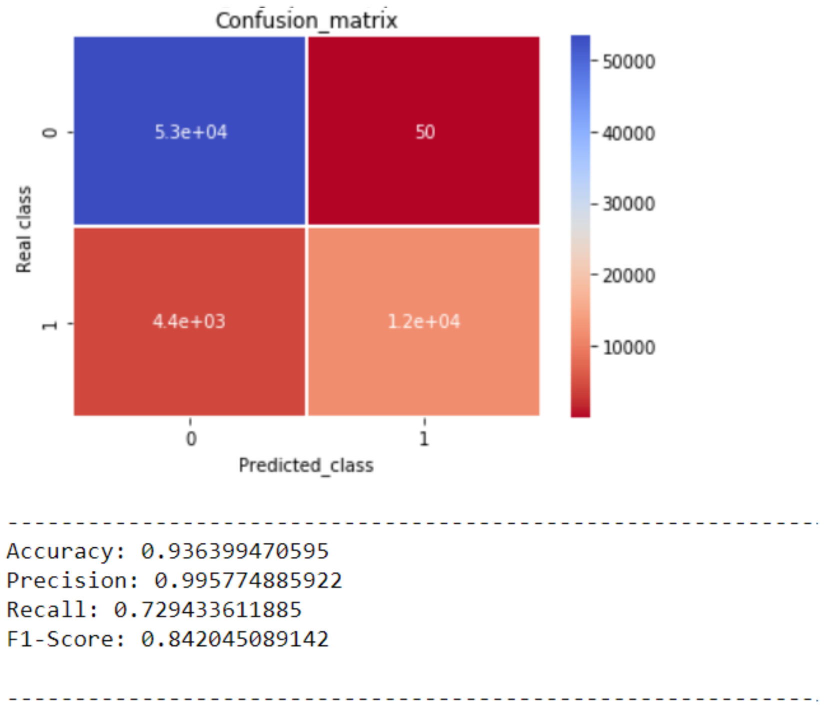
```
------------------------------------------------------------
Accuracy: 0.94326159512
Precision: 0.976508233825
Recall: 0.774497059734
F1-Score: 0.863849765258

------------------------------------------------------------
```

PCA

To make our oversampling work and to have a look at the trade-off between accuracy and the amount of used components, we worked with Principal Component Analysis.

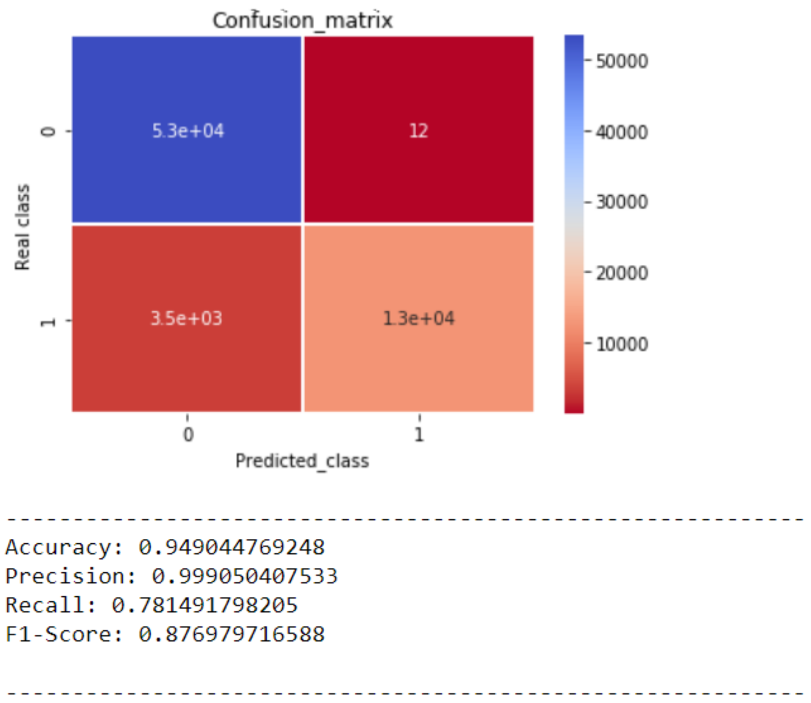
Reducing from 30 components to 10 (with oversampled data)

KNN



```
---------------------------------------------------------------
Accuracy: 0.936399470595
Precision: 0.995774885922
Recall: 0.729433611885
F1-Score: 0.842045089142

---------------------------------------------------------------
```

Random Forest



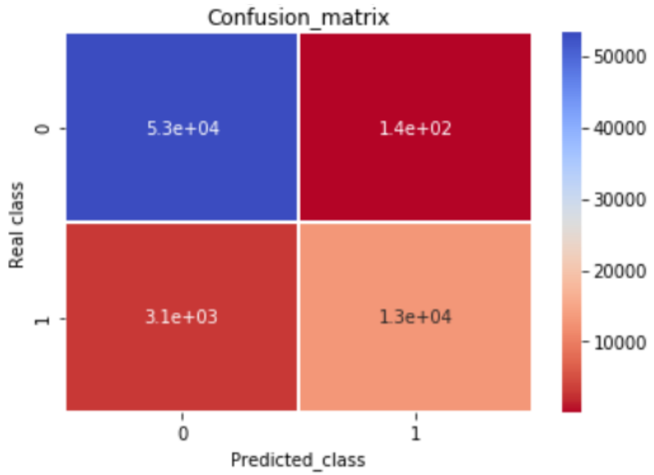
```
---------------------------------------------------------------
Accuracy: 0.949044769248
Precision: 0.999050407533
Recall: 0.781491798205
F1-Score: 0.876979716588

---------------------------------------------------------------
```

PCA

To make our oversampling work and to have a look at the trade-off between accuracy and the amount of used components, we worked with Principal Component Analysis.

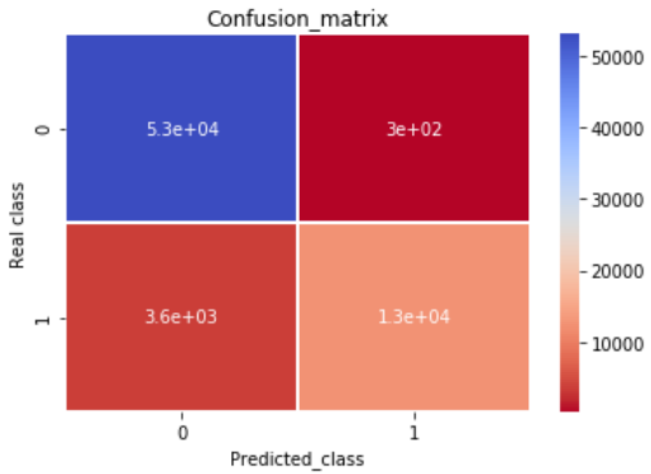
Reducing from 30 components to 10 (with oversampled data)

LGBM



```
------------------------------------------------------------
Accuracy: 0.953403728853
Precision: 0.989761868649
Recall: 0.807861343237
F1-Score: 0.889608397805

------------------------------------------------------------
```

XGBoost



```
------------------------------------------------------------
Accuracy: 0.94326159512
Precision: 0.976508233825
Recall: 0.774497059734
F1-Score: 0.863849765258

------------------------------------------------------------
```