Real-Time fraud detection system at scale

Team Member

Pawel Urbanowicz 108015016
Martin Ledl 108012012
Tobias Kick 108998413

Github Repository

The Github repository for this project can be found here:
https://github.com/mledl/BDMA_HW/tree/master/Project

Goal

The main goal of the project was to simulate real world use case of streaming processing for anomaly detection using dataset of fraud transactions. The sub goal was to prepare modern architecture for scalable big data analytics .

Responsibilities

Pawel Urbanowicz

- Preparing the whole architecture flow.
- Putting all components inside docker containers

Martin Ledl

- Creating application using spark structured streaming
- Creating application using kafka-streams

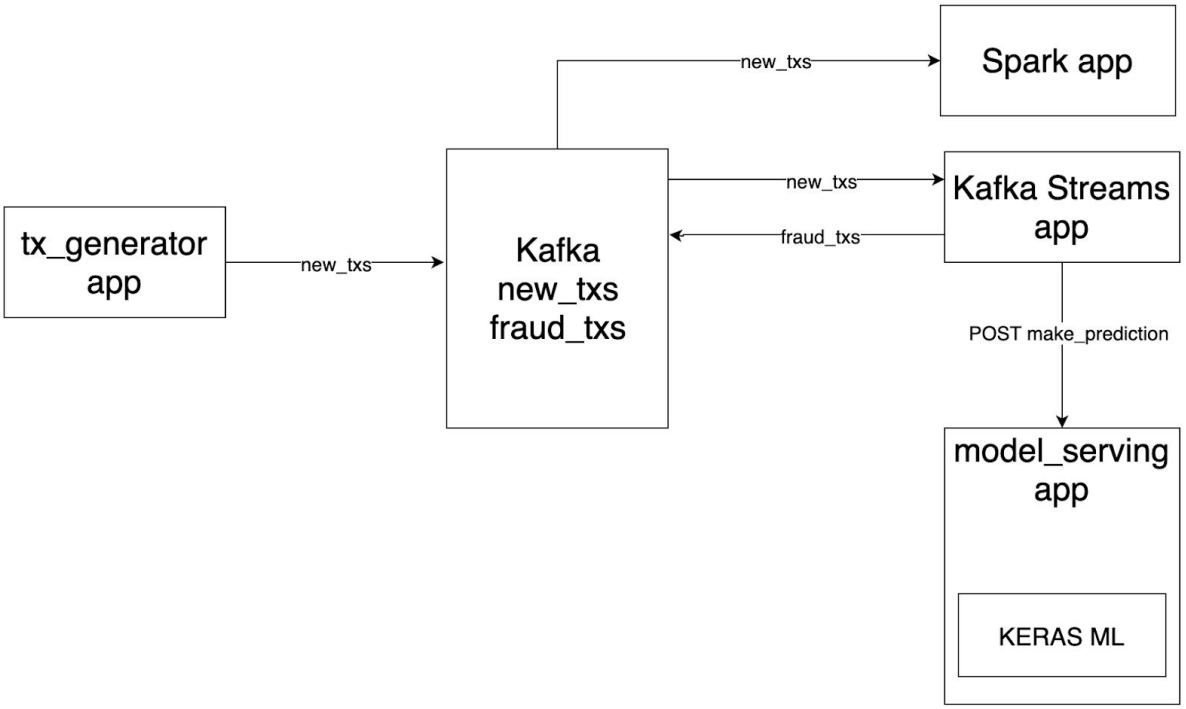
Tobias Kick

- Creating deep learning model using Keras + Tensorflow
- Creating flask app for exposing machine learning model

Technologies:

- Kafka + Kafka Streams + zookeeper
- Spark structured streaming
- Keras + TensorFlow
- Docker
- MlFlow
- Java
- Flask + python

Architecture



- tx_generator app - responsible for generating txs and sending them to the new_txs topic inside kafka. Written using python.
- kafka - central system of the whole architecture working as message broker.
- spark_app - application for doing real time stream processing using spark structured streaming. Application is listening on new_tex topic inside kafka. Written using Java
- kafka_streams app - application for doing real time prediction using kafka streams.Application is listening on new_txs topic and sending results to fraud_txs topic. Written using Java.
- model_serving app - application responsible for exposing deep learning model. Written using Flask and python.
- Keras ml - deep learning model written using Keras and Tensorflow.
- zookeeper - responsible for keeping information about kafka producers and consumers.

All the above components prepared to work inside docker containers.

Dataset

For testing our architecture we used dataset from recent competition on Kaggle which ended on 3th of October, 2019.

Data consists of two following tables which are joined by a unique transaction identifier and divided into training and test dataset:

- Identity: consists of 41 features
- Transaction: consists of 394 features

Environment Setup

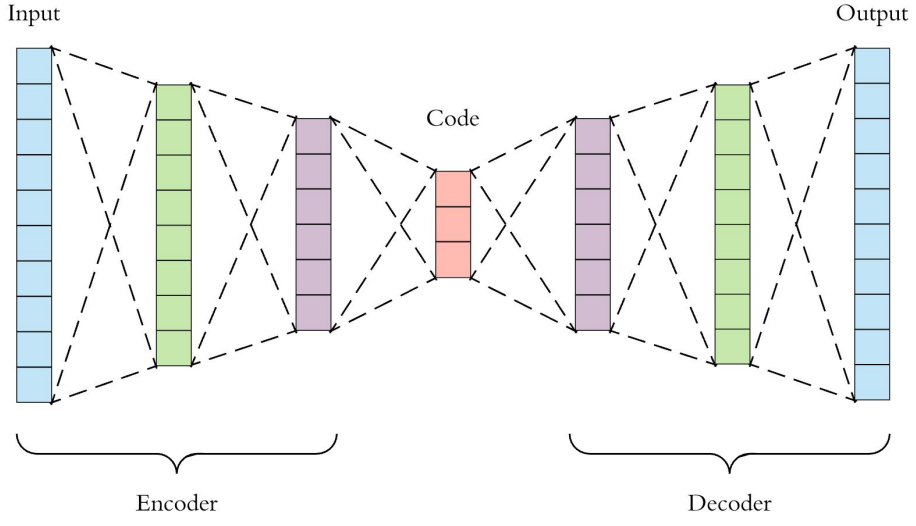
For local development we tested our code on a locally installed components and for target stage we used docker technology to wrap all components into reusable containers. To run the whole architecture is needed only to run dokcer-compose up inside console.

```
Killing zookeeper       ... done
MBP-Pawel:master-thesis pawelurbanowicz$ docker-compose up
Starting kafka          ... done
Starting zookeeper ... done
Starting schemaregistry ... done
Attaching to zookeeper, kafka, schemaregistry
zookeeper          | ZooKeeper JMX enabled by default
schemaregistry     | ===> ENV Variables ...
zookeeper          | Using config: /opt/zookeeper-3.4.13/bin/../conf/zoo.cfg
kafka              | waiting for kafka to be ready
schemaregistry     | ALLOW_UNSIGNED=false
schemaregistry     | COMPONENT=schema-registry
schemaregistry     | CONFLUENT_DEB_VERSION=1
schemaregistry     | CONFLUENT_MAJOR_VERSION=5
```

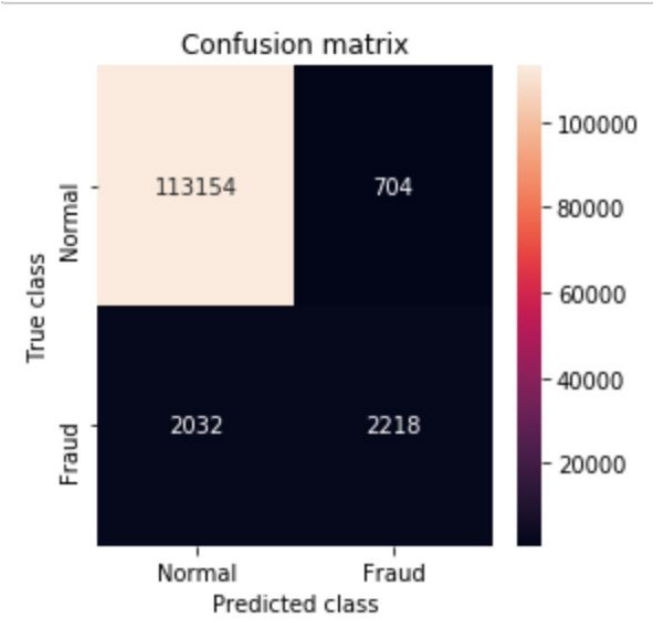
```
MBP-Pawel:master-thesis pawelurbanowicz$ docker ps
CONTAINER ID        IMAGE                          COMMAND                  CREATED             STATUS
34445da7257c        purbanow/spark-worker:latest   "/bin/bash /worker.sh"   22 seconds ago      Up 19 seconds
1dfd58824d06        purbanow/spark-worker:latest   "/bin/bash /worker.sh"   22 seconds ago      Up 20 seconds
4dcd0535315f        kafka-stream-predictor         "java -jar /usr/loca…"   22 seconds ago      Up 20 seconds
32f08ef3118e        model-service                  "/bin/sh -c 'flask r…"   24 seconds ago      Up 22 seconds
2b34044f0caa        purbanow/spark-master:latest   "/bin/bash /master.sh"   24 seconds ago      Up 22 seconds
1775caebffd6        wurstmeister/zookeeper         "/bin/sh -c '/usr/sb…"   24 seconds ago      Up 22 seconds
108ea1b76b23        wurstmeister/kafka             "start-kafka.sh"         24 seconds ago      Up 22 seconds
```

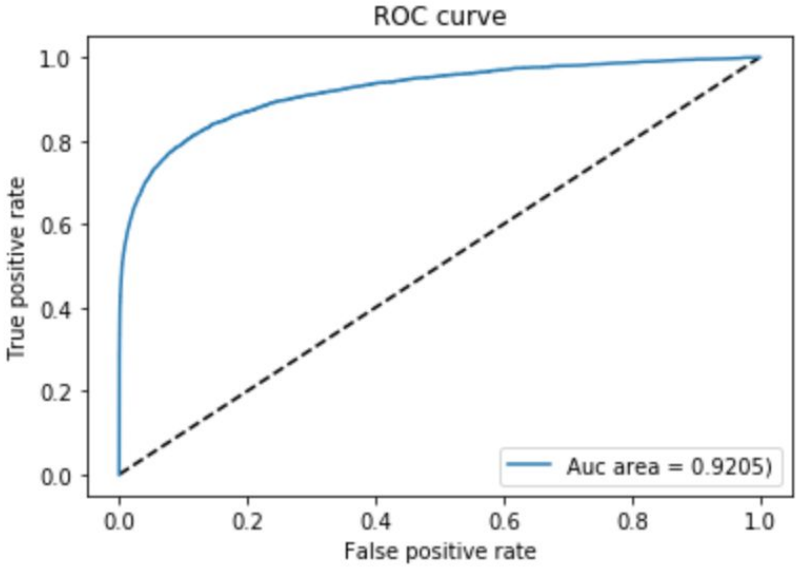
Deep learning model:

We create deep learning model for fraud detection using Keras which consist of autoencoder neural network and simple neural network.

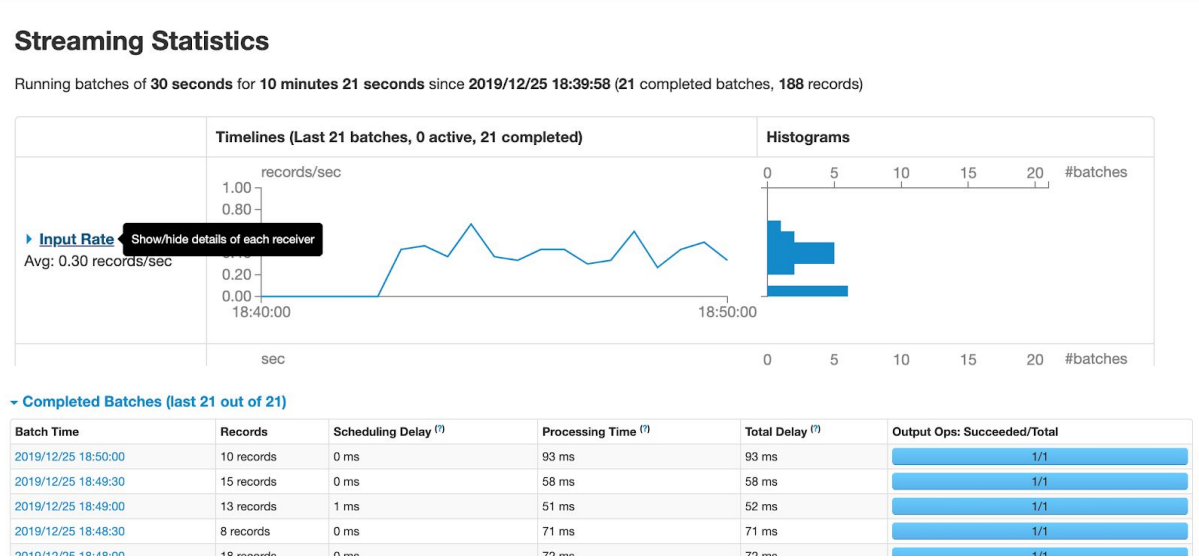


The accuracy of our machine learning model is 97.7% but in the case of anomaly detection we don't care so much about accuracy but more about precision, recall and AUC(area under curve). The confusion matrix and ROC diagram can be found below:





Spark



Further improvements:

- Start using Schema registry together with Apache Avro
- More stream processing with spark
- Add consumer application which would be using Apache Flink
- Start using Kubernetes for even more scalable and fault tolerant architecture
- Improvements for deep learning model.
- Try other machine learning techniques.