AI POWERED FRAUD DETECTION IN FINANCIAL TRANSACTIONS

IBM COURSERA ADVANCED DATA SCIENCE CAPSTONE

https://github.com/fdicugno/Coursera_AdvAiCapstone

CASE STUDY

Advanced Data Science with IBM Specialization

March 2020

Francesco di Cugno
Managing Consultant

Detecting financial frauds using machine learning

Key facts

- PwC estimates losses due to frauds between 2018 and 2020 at **42bn USD**[1]
- Besides financial losses, fraud generate other hidden costs such as **brand damage**, **loss of market** position, and **loss of opportunities**
- Frauds perpetrated by **customers are the most frequent case**[1]
- Financial institutions prevented 1.9bn USD in unauthorised transactions in 2018[2]
- Robust process and **AI technologies** have proven to be **effective for fraud detection** and prevention

Challenges for predictive models

- Datasets are **highly imbalanced**. Fraudulent transactions account less than1% of the overall financial transactions
- Accuracy cannot be considered a good metric for such predictions (**Accuracy paradox**)
- **Pattern change over time** and are difficult to be modelled
- Datasets to be processed tend to be very large
- Features engineering might require several time

[1] Source: https://www.pwc.com/gx/en/forensics/gecs-2020/pdf/global-economic-crime-and-fraud-survey-2020.pdf
[2] Source: https://www.ukfinance.org.uk/system/files/Fraud%20The%20Facts%202019%20-%20FINAL%20ONLINE.pdf

Case Study: Frauds detection using ML

Case Study

Problem

Define a predictive model for fraud detection and develop a fraud scoring service on the could exposed as REST API.

Dataset

Dataset Synthetic Financial Datasets For Fraud Detection[1] (Version 2) from Kaggle has been used

Solution

Machine learning models have been evaluated and compared.

A deep learning model has been selected and deployed on IBM Watson Machine Learning

Benefits

Cloud computing provide a secure and cost effective way to face the challenges posed by the computational issues. Models can be easily scaled thanks to dynamic resource allocation.

[1] Source: https://www.kaggle.com/ntnu-testimon/paysim1

Data exploration

Dataset

- Synthetic Financial Datasets For Fraud Detection from Kaggle[1]
- **6.3mln simulated transactions** of type CASH_IN, CASH_OUT, DEBIT, PAYMENT, and TRANSFER
- 8.2k total **frauds (~0.13% of the total transactions)**

Findings

- Frauds affect **only transactions of types CASH_OUT and TRANSFER** and and **only transactions Consumer to Consumer**
- Fraud transactions average amount is 10 times the one of genuine ones
- 2673 out of the 8213 frauds are performed by (destination) customers that account only one transaction.
- **Generally a malicious customer performs only one fraud** and is generally the latest transaction of that customer. There are 44 cases **(~0.5%) of reiteration** in which malicious performed 2 frauds. No customer performed more than 2 frauds.
- Old original balance and new one as well as old destination balance and new are obviously highly correlated. All of them correlate with the transaction amount. Frauds correlate with the step feature. Anyway, not having information related to the meaning of the feature no informed conclusion can be taken.
- The isFraud flag does not present any relevant correlation with statistical significance with other variables.

[1] Source: https://www.kaggle.com/ntnu-testimon/paysim1

Model development: where to start?

- The problem subject of the study is a typical anomaly detection problem that can be threated as **(binary) classification problem**
- Due to its characteristics **different models need to be tested** in order to identify the best performer
- Accuracy cannot be considered a good metric for such predictions (**Accuracy paradox**)
- There exist a number of classification algorithms in machine learning that **leverage both supervised and unsupervised learning**. Five algorithms have been selected as starting point

	Supervised	Unsupervised
Machine Learning	**Logistic Regression** A statistical model that uses a logistic function to model a binary dependent variable. It is very efficient and does not requires extensive resources. **Random Forest** Random forest is a classification algorithm that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes or mean prediction of the individual trees[1].	**Isolation Forest** It is an unsupervised machine learning algorithm for anomaly detection. It isolates observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature[2]
Deep Learning	**Perceptron** It is an algorithm that leverages a fully connected neural network and a logistic function for supervised learning of binary classifiers.	**Autoencoder** It is a neural network that is used to learn the encoding of a dataset. Normally it leverages dimensionality reduction to learn how to ignore noise.

[1] Source: https://en.wikipedia.org/wiki/Random_forest
[2] Source: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html

Feature engineering

- As data was simulated, there were no missing values or relevant data quality issues
- Depending on the model, different features were selected. Deep learning models performed better when the ‘step’ feature was included and the data was normalized
- Redundant features that highly correlate were removed. Those include the new origin and destination balances as they highly correlate with the amount and the old origin and destination balances.
- The counterparty identifiers (CI) included the type: consumer (C) or merchant (M). Their were built with the first character indicating the type (T) and a 12 digits identifier (ID). To distribute customer with different types on different hyperplanes, the identifier was encoded with the formula $\widehat{CI} = ordinal(T) \cdot 10^{12} + ID$

Tested Model: Logistic Regression

Pros

- Computational efficiency
- Highly interpretable
- It does not require tuning
- The training is fast
- Fully supported by Spark MLLib

Cons

- It cannot be used to solve non-linear problems
- It requires the identification of all independent variables, as correlated variables affect negatively the performances
- It is not resilient to overfitting
- Does not perform well with unbalanced data

Features

- Transaction Amount
- Old balance of the origin
- Old Balance of the destination
- Origin Customer
- Destination Customer
- Type of Transaction (encoded)

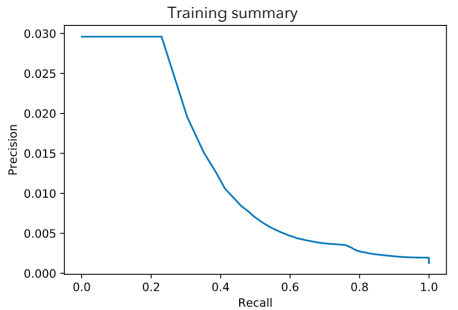
Results

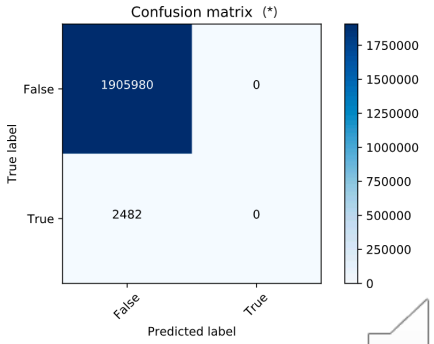
Logistic regression does not perform well with highly unbalanced data. Indeed, in the tests executed the model was not able to identify any fraud.

The model offered the worst performance among the selected ones.

Evaluation

Precision of True	0.0
Precision of False	0.99869948
Recall of True	0.0
Recall of False	1.0
F-1 Score	0.9986995





(*) Calculated fitting the model on 70% randomly selected samples and testing it on the remaining 30% of the whole dataset

Tested Model: Random Forest

Pros

- It ranks among the best supervised learning algorithms
- It provides a reliable feature importance estimate
- It has a wide range of hyperparameters that can be tuned via grid search to optimize the model performances
- It performs very well with anomaly detection
- Fully supported by Spark MLLib

Cons

- The model is more difficult to interpret with respect to a simple decision tree
- Training the model has high computational and memory costs.
- Dur to the resource needs real time prediction using the model can be challenging

Features

- Transaction Amount
- Old balance of the origin
- Old Balance of the destination
- Origin Customer
- Destination Customer
- Type of Transaction (encoded)

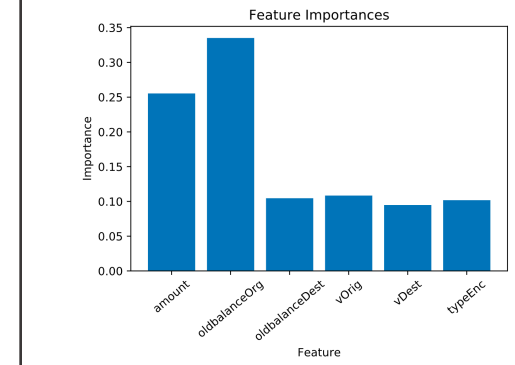
Results

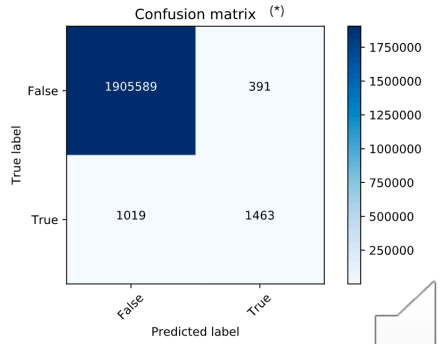
Considering the limited number of features available, the best model identified tuning the hyperparameters performed well identifying 1463 frauds over 2482 (59%) total with only 391 false positives (approximately 1 false positive every 5).

The old balance of the origin and the amount are the features with the highest predictive power.

Evaluation

Precision of True	0.7891046386
Precision of False	0.999465543
Recall of True	0.5894439968
Recall of False	0.9997948562
F-1 Score	0.9992611852





(*) Calculated fitting the model on 70% randomly selected samples and testing it on the remaining 30% of the whole dataset

Tested Model: Isolation Forest

Pros

- The training does not need previous labelling of the observations (unsupervised learning)
- It performs well for anomaly detection
- It has a wide range of hyperparameters that can be tuned via grid search to optimize the model performances
- Full implementation available in scikit-learn

Cons

- The model is difficult to interpret and visualize
- Training the model has high computational and memory costs
- Not natively supported by Spark MLLib. Third parts libraries do not offer enterprise performance.

Features

- Transaction Amount
- Old balance of the origin
- Old Balance of the destination
- Origin Customer
- Destination Customer
- Type of Transaction (encoded)

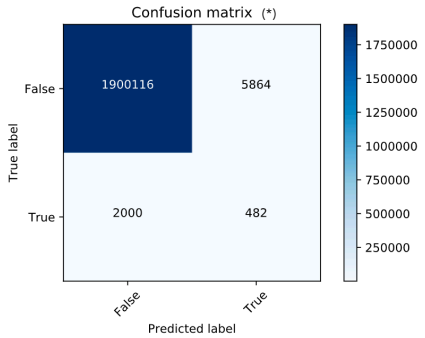
Results

Considering the limited number of features available, the best model identified tuning the hyperparameters had a good rate of true positive identified 482 frauds over 2482. Nevertheless there is a ratio of ~10 false negatives every true positive.

Tuning the hyperparameters it is possible to increase the amount of frauds identified, but the odd to the false positive increases above 100x.

Evaluation

Precision of True	0.0759533564
Precision of False	0.9989485394
Recall of True	0.1941982272
Recall of False	0.9969233675
F-1 Score	0.1091980063



(*) Calculated fitting the model on 70% randomly selected samples and testing it on the remaining 30% of the whole dataset

Tested Model: Autoencoder

Pros

- The training does not need previous labelling of the observations (unsupervised learning)
- Once trained, the predictions are very fast.
- It can be used to solve non-linear problems
- Easy to implement leveraging Keras and Tensorflow

Cons

- Modelling and tuning can be time consuming due to the endless network topologies and the high number of hyperparameters
- Quality of the predictions of neural networks depend a lot on training data. They can overfit the training data but perform poorly with unknown data
- Training is computationally expensive. The issue can be mitigated using Apache Spark or SystemML.

Features

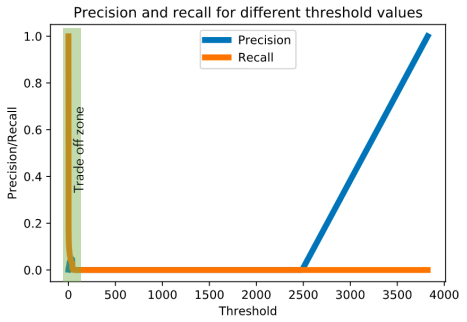
- Step
- Transaction Amount
- Old balance of the origin
- Old Balance of the destination
- Origin Customer
- Destination Customer
- Type of Transaction (encoded)

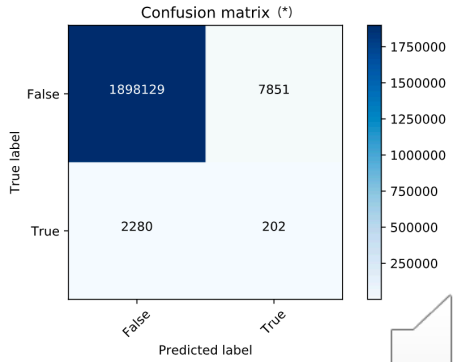
Results

The performance of the model are not satisfactory. A threshold of 10 provides the best trade off for the model. 10% of the positives is identified and the odd against false negatives is approximately 1:38, which is not ideal for a binary classifier. That ratio grows exponentially for decreasing thresholds. For a threshold of 0.1 it becomes 1:207.

Evaluation

Precision of True	0.0250838197
Precision of False	0.9988002583
Recall of True	0.0813859790
Recall of False	0.9958808592
F-1 Score	0.0383483626





(*) Calculated fitting the model on 70% randomly selected samples and testing it on the remaining 30% of the whole dataset

Tested Model: Perceptron

Pros

- Many performant implementation exists
- Once trained, the predictions are very fast.
- It can be used to solve non-linear problems
- Easy to implement leveraging Keras and Tensorflow

Cons

- Quality of the predictions of neural networks depend a lot on training data. They can overfit the training data but perform poorly with unknown data
- Training is computationally expensive. The issue can be mitigated using Apache Spark or SystemML.

Features

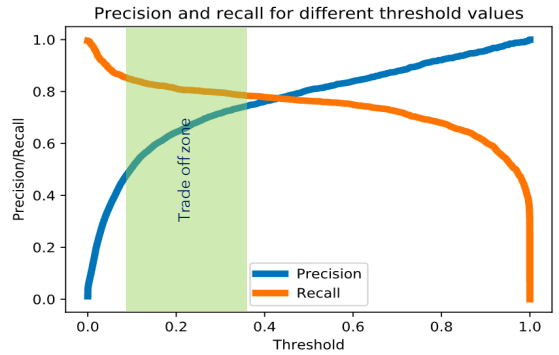
- Step
- Transaction Amount
- Old balance of the origin
- Old Balance of the destination
- Origin Customer
- Destination Customer
- Type of Transaction (encoded)

Results

The model resulted the best performer among the models tested, followed by the Random Forest. **For the scope of this case study, the model is selected for the deployment.**

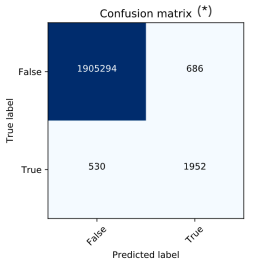
A business decision is needed to select a proper threshold. Whether in general a precision of 50% is not a good result, in this specific case it can still be considered in order to minimize the potential losses generated by the true positive. A threshold of 0.1 provides such kind of trade off. For lower thresholds the recall will increase quickly, leading to a very high amount of false positive. That is also not desired.

Evaluation

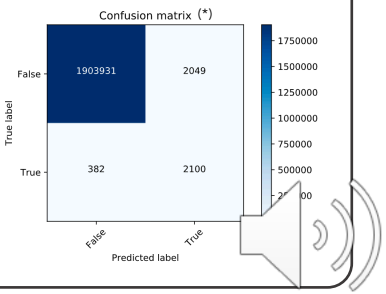


	Threshold: 0.35	Threshold: 0.1
Precision of True	0.73995451099	0.50614605929
Precision of False	0.99972190506	0.99979940272
Recall of True	0.78646253021	0.84609186140
Recall of False	0.99964008016	0.99892496249
F-1 Score	0.7625	0.63338862916

Threshold: 0.35



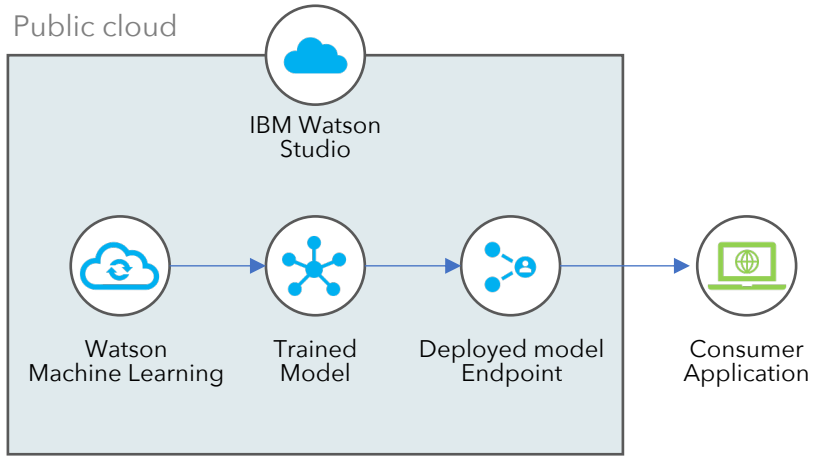
Threshold: 0.1



(*) Calculated fitting the model on 70% randomly selected samples and testing it on the remaining 30% of the whole dataset

Deployment

Architecture



Architecture

1. Deployment

```
[19] Mi
model_uid = wml_client.repository.get_model_uid(published_model_details)
print(f'Deploy model {model_uid}')
deployment = wml_client.deployments.create(model_uid, 'Coursera Advanced AI Model (Deployed)')

deployment

DEPLOY_IN_PROGRESS
DEPLOY_SUCCESS

Successfully finished deployment creation, deployment_uid='072c90fe-f912-4fca-8906-9ac680a7e5e9'

{'metadata': {'guid': '072c90fe-f912-4fca-8906-9ac680a7e5e9',
  'url': 'https://us-south.ml.cloud.ibm.com/v3/wml_instances/cabc8f15-b6a5-408f-bad1-41c87727ad81/deployments/072c90fe-f912-4fca-8906-9ac680a7e5e9',
  'created_at': '2020-03-24T21:07:04.991Z',
  'modified_at': '2020-03-24T21:07:05.247Z'},
 'entity': {'runtime_environment': 'python-3.6',
  'name': 'Coursera Advanced AI Model (Deployed)',
  'scoring_url': 'https://us-south.ml.cloud.ibm.com/v3/wml_instances/cabc8f15-b6a5-408f-bad1-41c87727ad81/deployments/072c90fe-f912-4fca-8906-9ac680a7e5e9/online',
  'deployable_asset': {'name': 'Coursera Advanced AI Model',
   'url': 'https://us-south.ml.cloud.ibm.com/v3/wml_instances/cabc8f15-b6a5-408f-bad1-41c87727ad81/published_models/a5e20f77-5289-4427-be14-e74e8e28330b',
   'guid': 'a5e20f77-5289-4427-be14-e74e8e28330b',
   'created_at': '2020-03-24T21:07:04.941Z',
   'type': 'model'},
  'description': 'Description of deployment',
  'status_details': {'status': 'DEPLOY_SUCCESS'},
  'model_type': 'tensorflow-1.15',
  'status': 'DEPLOY_SUCCESS',
  'type': 'online',
  'deployed_version': {'url': 'https://us-south.ml.cloud.ibm.com/v3/ml_assets/models/a5e20f77-5289-4427-be14-e74e8e28330b/versions/32a8bc0c-4c53-44e8-929a-3f5a017b08be',
   'guid': '32a8bc0c-4c53-44e8-929a-3f5a017b08be'}}}

[22] Mi
wml_client.deployments.list()

GUID                                  NAME                                  TYPE    STATE          CREATED                  FRAMEWORK        ARTIFACT TYPE
072c90fe-f912-4fca-8906-9ac680a7e5e9  Coursera Advanced AI Model (Deployed)  online  DEPLOY_SUCCESS  2020-03-24T21:07:04.991Z  tensorflow-1.15  model

[23] Mi
scoring_endpoint = wml_client.deployments.get_scoring_url(deployment)
print(f'Scoring endpoint: {scoring_endpoint}')

Scoring endpoint: https://us-south.ml.cloud.ibm.com/v3/wml_instances/cabc8f15-b6a5-408f-bad1-41c87727ad81/deployments/072c90fe-f912-4fca-8906-9ac680a7e5e9/online
```

2. Test

```
[39] Mi
print('Testing deployed model...')
payment_data = x_test[0].tolist()

scoring_payload = {'values': [payment_data]}
scores = wml_client.deployments.score(scoring_endpoint, scoring_payload)
print('Score: Predicted {} - Actual {}'.format(scores['values'][0], y_test[0]))

Testing deployed model...
Score: Predicted [0.0] - Actual 0

[46] Mi
it = np.where(y_test == 1)

y_test_t = y_test[it]

print(f'Fraudolent transaction: {x_test_t[0]}')

scoring_payload = {'values': [x_test_t[0].tolist()]}
scores = wml_client.deployments.score(scoring_endpoint, scoring_payload)

fraud_scoring_threshold = 0.1

is_fraud = scores['values'][0][0] > fraud_scoring_threshold

print(f'Prediction: Is Fraud? {is_fraud}')

Fraudolent transaction: [[-0.29694012 -0.28888892 -0.32524011  0.1595821  -0.71474652  1.98121586]]
Prediction: Is Fraud? True
```

Conclusion and future work

- The case study has provided a scoring model for fraud detection for financial transactions that can be deployed as REST API on IBM Watson Machine Learning
- As the selected model implement a neural network a periodical refitting/re-training of the model has to be foreseen to discover new patterns.
- The scoring model can be refined including additional features of the financial transactions or of the involved counterparties
- Multiple models can be combined to detect more frauds while keeping the false positive odd acceptable

AUTHOR NOTE

The case study presented in this document is solely an educational exercise. Any opinion, finding, conclusion, or recommendation is that of the author only and does not reflect by any means the view of any of the mentioned organizations, their employees or their administrations. The data disclosed in this document come from documentation, articles, and interviews publicly available or licensed to the author for this purpose. This document has been produced for the IBM Data Science Professional Certification Program only in order to evaluate and grade the knowledge of the author about the subjects of the 'Advanced Data Science Capstone' learning module.