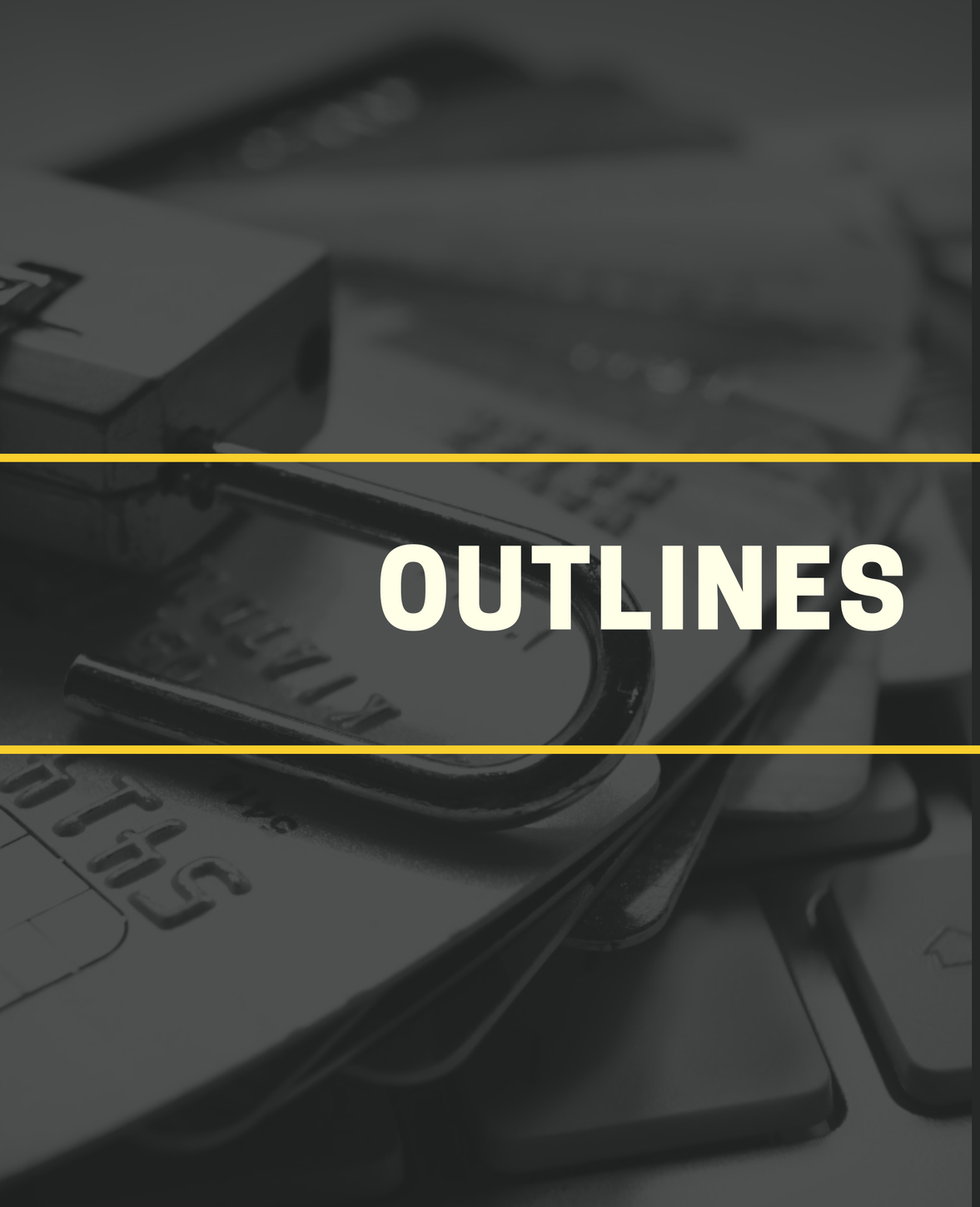
IBM COURSERA ADVANCED DATA SCIENCE CAPSTONE PROJECT

CREDIT CARD FRAUD DETECTION

Prepared by: Armando Medina



OUTLINES

- DATASET
- USE CASE
- DATA EXPLORATION
- DATA CLEANING
- MODEL DEFINITION & TRAINING
- MODEL EVALUATION
- DEPLOY
- CONCLUSION

DATASET

The datasets contains transactions made by credit cards in September 2013 by european cardholders.This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

HTTPS://WWW.KAGGLE.COM/MLG-ULB/CREDITCARDFRAUD

USE CASE

Annual global fraud losses reached $21.8 billion, according to Nilson Report. Probably you feel very lucky if you are a fraud. About every 12 cents per $100 were stolen in the US during the same year.

```
def traditional_fraud_detection(user_purchase_transaction):
  purchase = user_purchase_transaction

  if (purchase.location ==  casa_del_pld ) & ( tamano_compra  > 1000)
    return false
  else
    return true
```

USE CASE

90% OF BANKING DETECTION SOLUTIONS ARE RULE-BASED

USE CASE

FRAUD ANALYST VS MACHINE
HOW DO THEY COMPARE?

FRANKONFRAUD.COM

FRAUD ANALYST

Fraud Analysts work hundreds of cases a day, applying their expertise to find and identity fraud.

When they find a fraud they decline the loan or transaction.

STOP!
DECLINE!

THEIR BENFITS

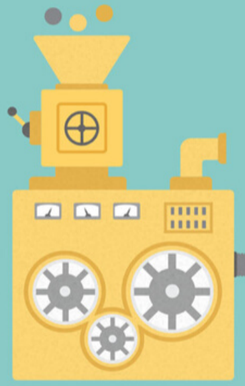
1) They can reason
2) They are adaptive
3) They can learn quickly
4) They stop fraud
5) They can communicate

THEIR FLAWS

1) They can be biased
2) They get tired
3) They workload is limited
4) Performance can vary
5) Hard to scale up with them

FRAUD MACHINE

Machines are trained on millions of fraud and non fraud transactions



They instantly scan for thousands of risk patterns and produce scores and alerts.

999
HIGH RISK!

THEIR BENEFITS

1) They are not biased
2) They are fast.
3) They work 24 hours a day
4) They can be very accurate
5) They help you scale

THEIR FLAWS

1) They have false positives
2) They can't stop fraud alone
3) Can't think creatively
4) Hard to understand
5) Bad data can corrupt them

USE CASE

Using machine learning algorithms can help fight against bank fraud detection.

DATA EXPLORATION

```
frauds = df_data_1[df_data_1.Class == 1]
normal = df_data_1[df_data_1.Class == 0]

print("Normal Transitions: {} \nFraud Transitions: {}".format(normal.shape, frauds.shape))
```

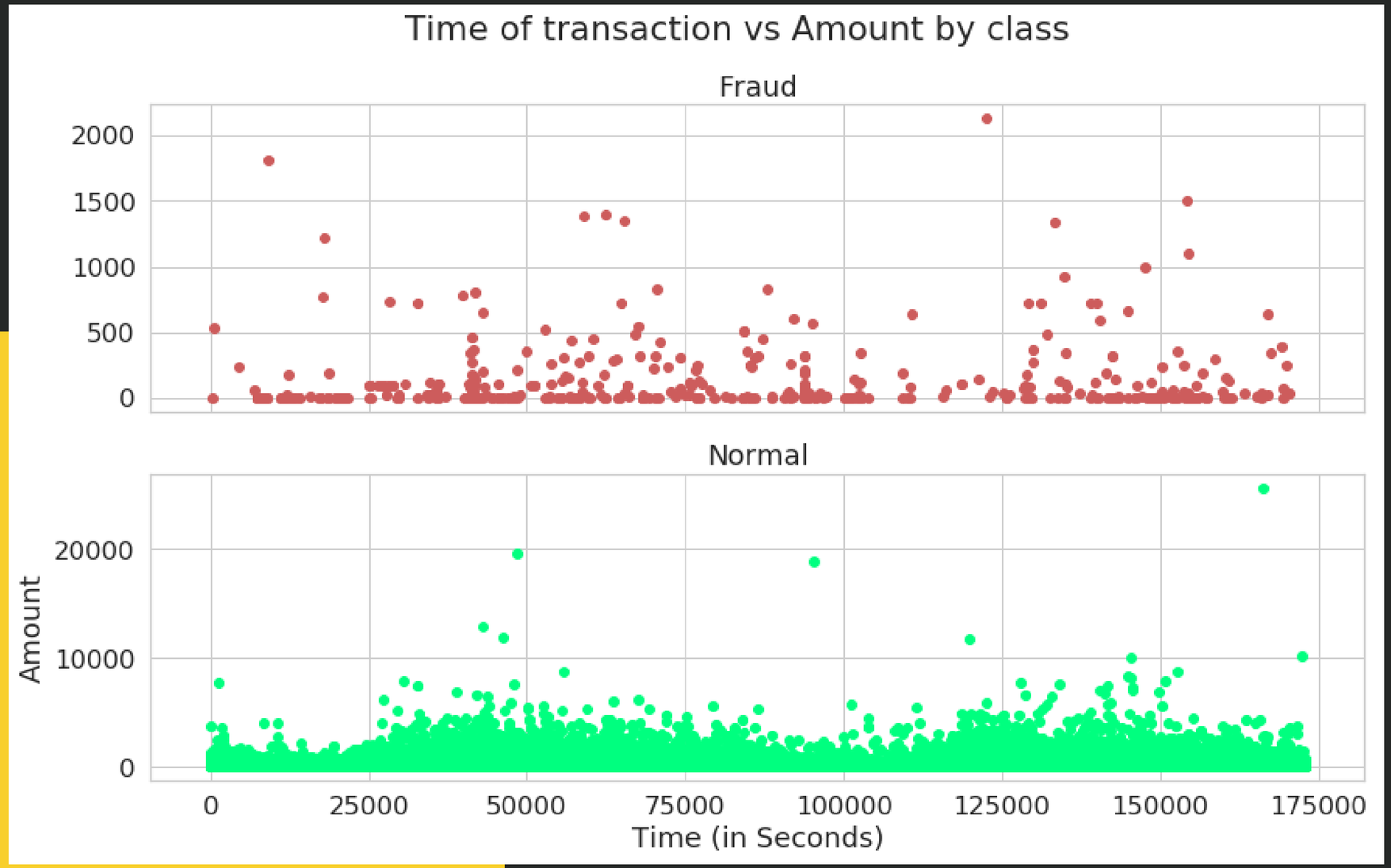
```
Normal Transitions: (284315, 31)
Fraud Transitions: (492, 31)
```

In a first approach we can determine that we have a highly unbalanced data set. Normal transactions are much more than fraudulent

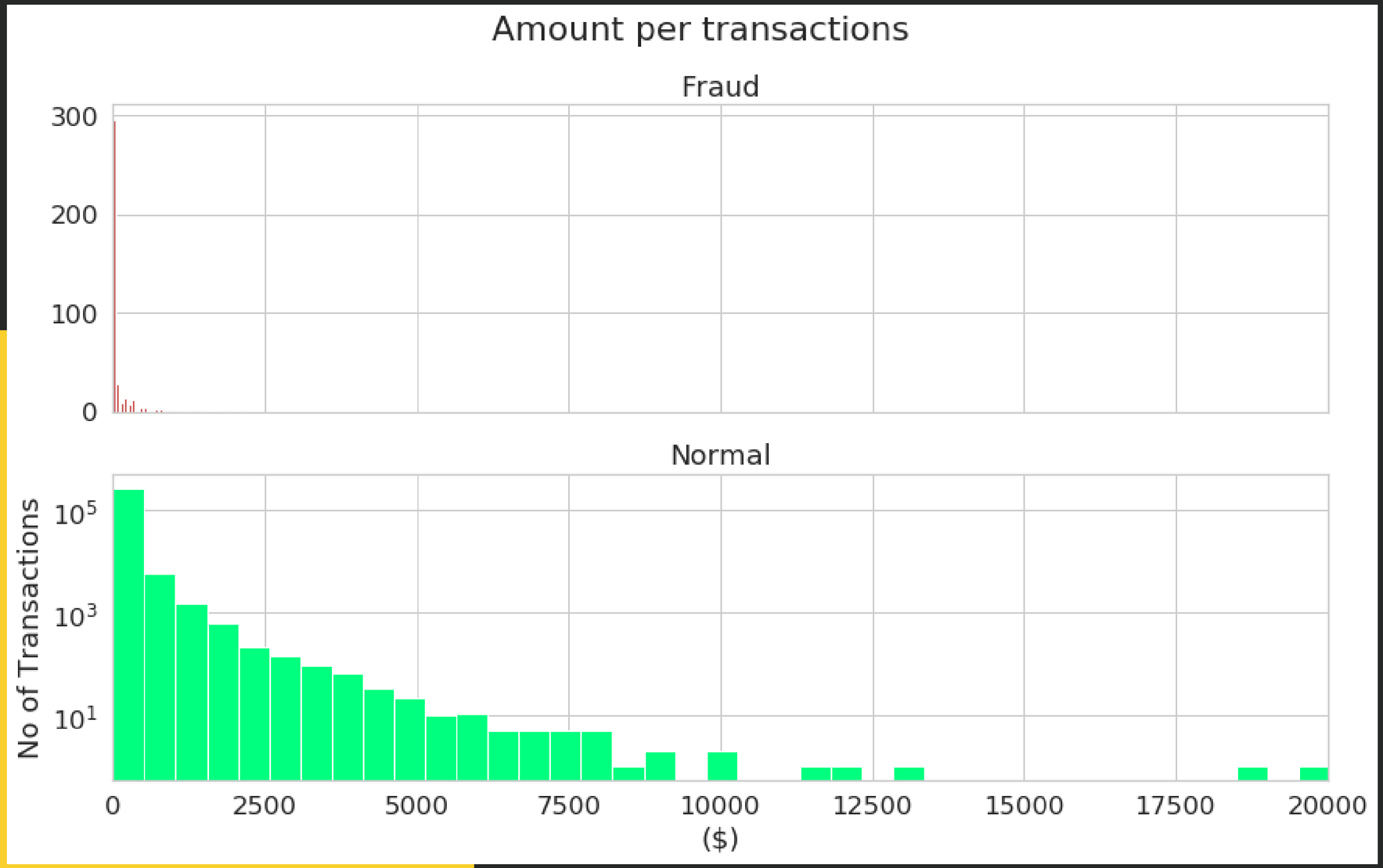
DATA EXPLORATION

```
Normal Transitions Amount Describe: count    284315.000000
mean         88.291022
std         250.105092
min           0.000000
25%           5.650000
50%          22.000000
75%          77.050000
max       25691.160000
Name: Amount, dtype: float64
--

Fraud Transitions Amount Describe: count     492.000000
mean      122.211321
std       256.683288
min         0.000000
25%         1.000000
50%         9.250000
75%       105.890000
max      2125.870000
Name: Amount, dtype: float64
```

DATA EXPLORATION



DATA EXPLORATION



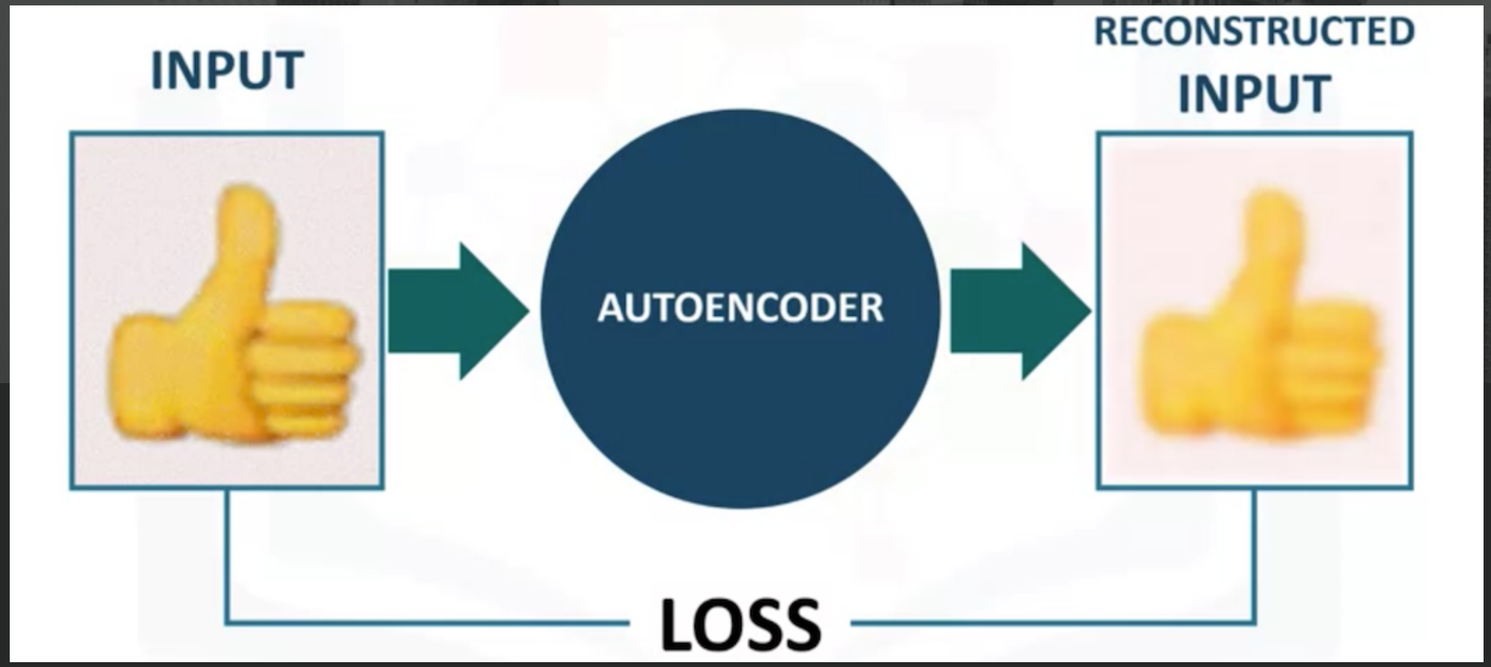
Data Cleaning

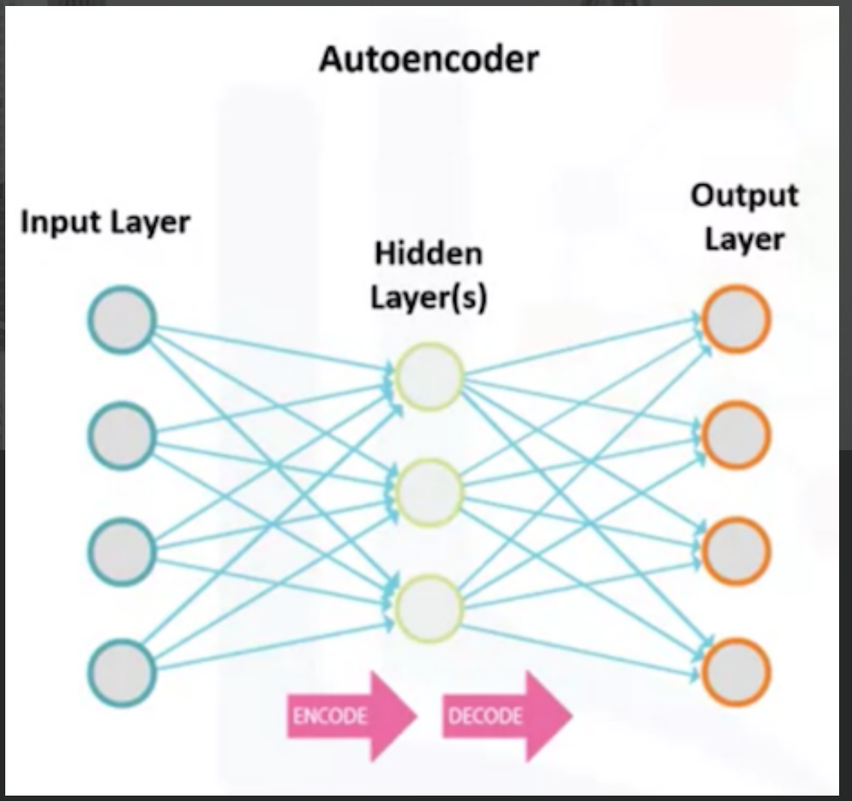
Data cleansing ¶

In [28]:
```
from sklearn.preprocessing import StandardScaler

df_data = df_data_1.drop(['Time'], axis=1)
df_data['Amount'] = StandardScaler().fit_transform(df_data['Amount'].values.reshape(-1, 1))

x_train, x_test = train_test_split(df_data, test_size=0.3, random_state=RANDOM_SEED)

x_train = x_train[x_train.Class == 0]
x_train = x_train.drop(['Class'], axis=1)
y_test = x_test['Class']
x_test = x_test.drop(['Class'], axis=1)
x_train = x_train.values
x_test = x_test.values


print("x_train: {} \nx_test: {}".format(x_train.shape, x_test.shape))
```

```
x_train: (199008, 29)
x_test: (85443, 29)
```

MODEL DEFINITION





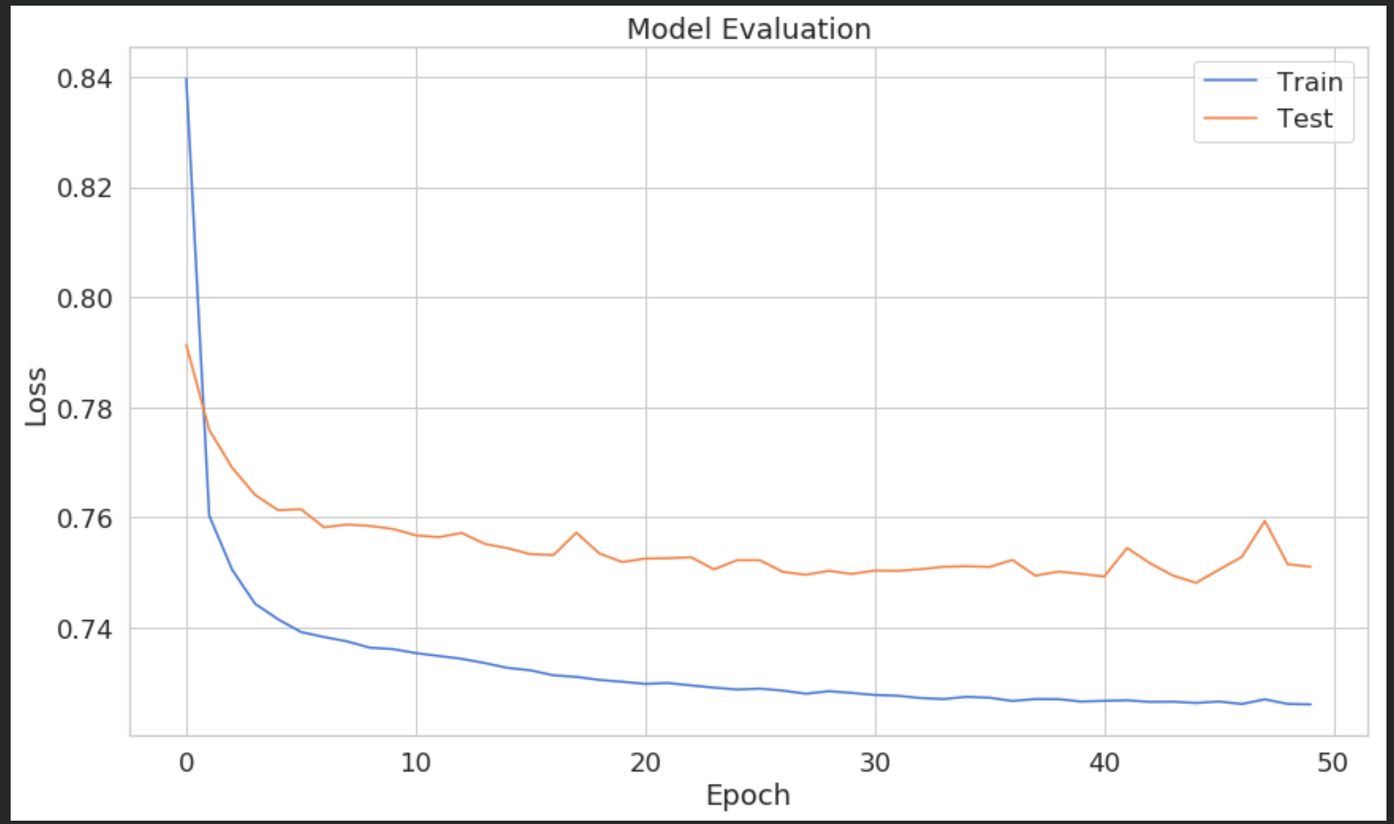
MODEL TRAINING

TRAIN THE MODEL FOR 50 EPOCHS WITH A BATCH SIZE OF 42 SAMPLES

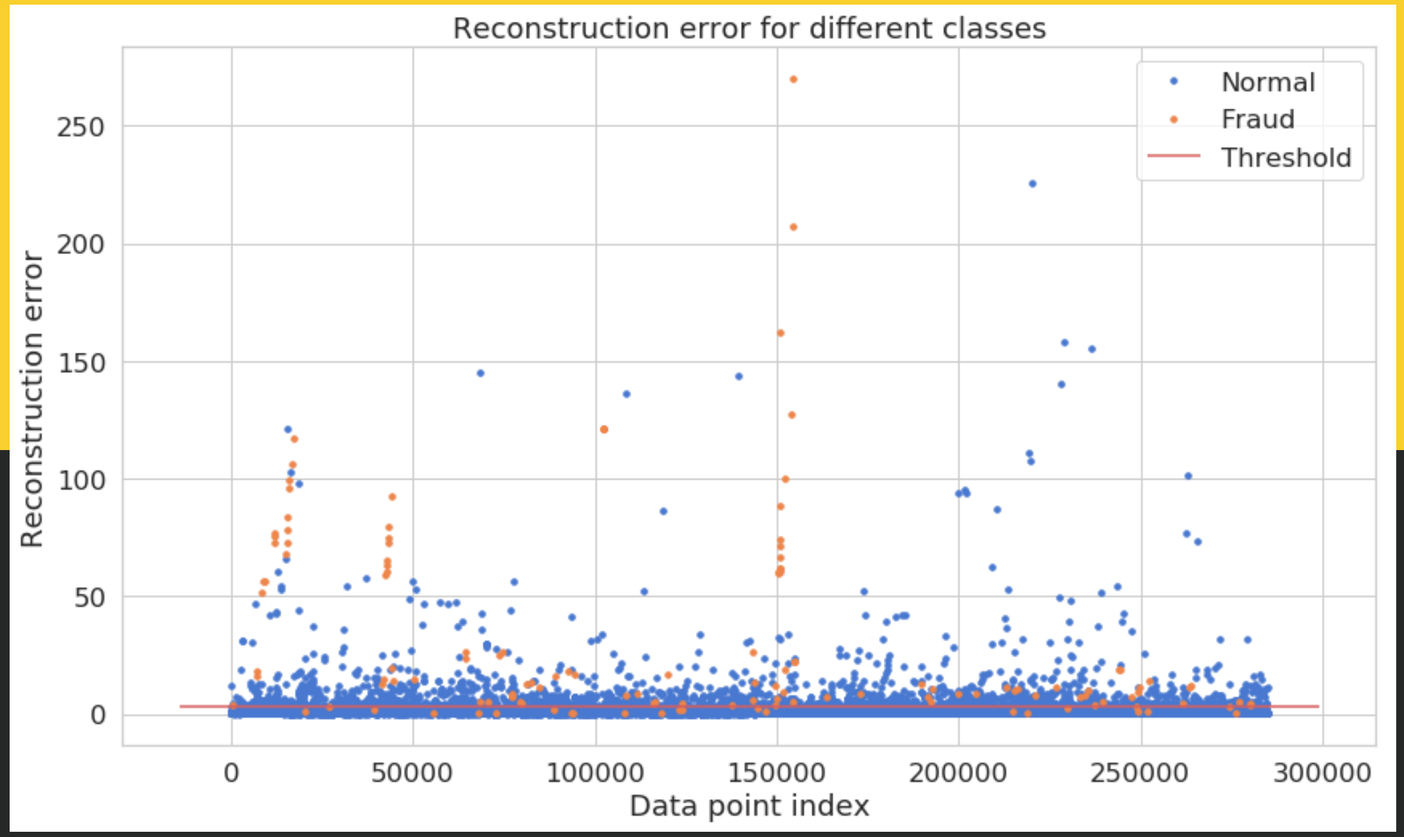
```
Train on 199008 samples, validate on 85443 samples
Epoch 1/50
199008/199008 [==============================] - 171s 858us/step - loss: 0.8397 - acc: 0.5774
Epoch 2/50
199008/199008 [==============================] - 120s 603us/step - loss: 0.7603 - acc: 0.6693
Epoch 3/50
199008/199008 [==============================] - 124s 622us/step - loss: 0.7505 - acc: 0.6748
Epoch 4/50
199008/199008 [==============================] - 146s 734us/step - loss: 0.7443 - acc: 0.6791
Epoch 5/50
199008/199008 [==============================] - 141s 708us/step - loss: 0.7415 - acc: 0.6799
Epoch 6/50
199008/199008 [==============================] - 139s 698us/step - loss: 0.7392 - acc: 0.6810
Epoch 7/50
199008/199008 [==============================] - 145s 730us/step - loss: 0.7383 - acc: 0.6825
Epoch 8/50
199008/199008 [==============================] - 144s 724us/step - loss: 0.7375 - acc: 0.6832
Epoch 9/50
199008/199008 [==============================] - 152s 766us/step - loss: 0.7363 - acc: 0.6842
Epoch 10/50
```

MODEL EVALUATION

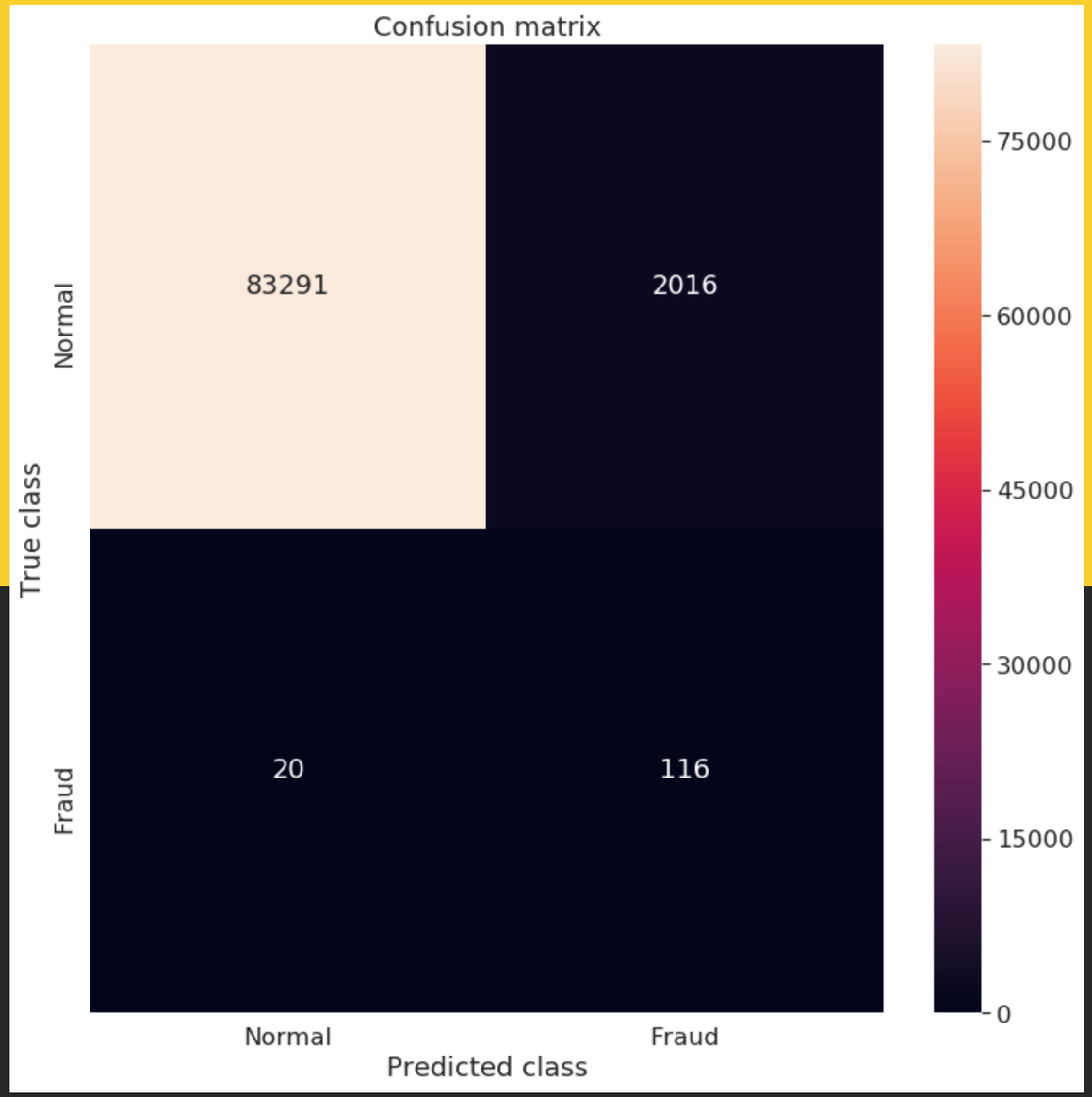
	reconstruction_error	true_class
count	85443.000000	85443.000000
mean	0.740300	0.001592
std	3.349338	0.039865
min	0.045763	0.000000
25%	0.247906	0.000000
50%	0.387917	0.000000
75%	0.618083	0.000000
max	269.822409	1.000000



MODEL EVALUATION



To predict whether a new transaction is normal or fraudulent, we will calculate the rebuild error from the transaction data. If the error is greater than a predefined threshold, we will mark it as fraud



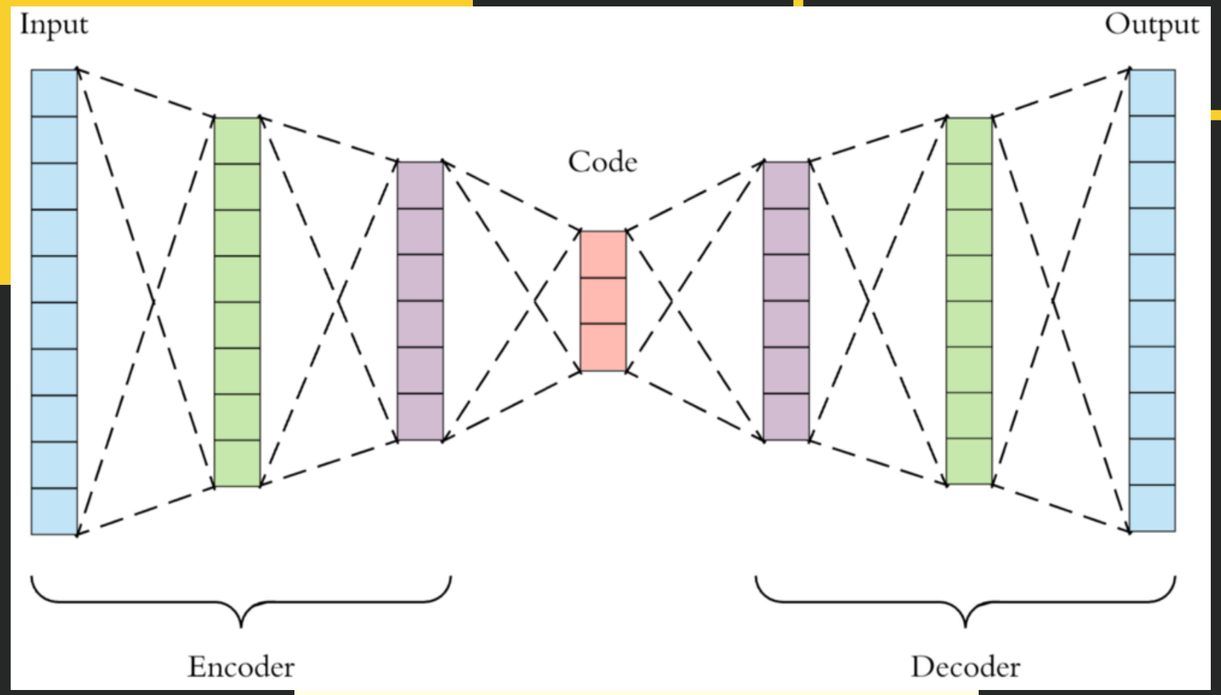
MODEL EVALUATION

The model detects many of the fraudulent cases but the problem is that the number of normal attempts classified as fraud is really high.

DEPLOY THE KERAS MODEL

```
In [69]: created_deployment = client.deployments.create(published_model

         ########################################################

         Synchronous deployment creation for uid: '22cd017e-b301-4315

         ########################################################


         INITIALIZING
         DEPLOY_IN_PROGRESS
         DEPLOY_SUCCESS


         ------------------------------------------------------------
         Successfully finished deployment creation, deployment_uid='3
         ------------------------------------------------------------



In [70]: #scoring_endpoint = client.deployments.get_scoring_url(created
         scoring_endpoint = created_deployment['entity']['scoring_url']
         print(scoring_endpoint)

         https://us-south.ml.cloud.ibm.com/v3/wml_instances/5cc01d3f-
         0f809a/online
```

CONCLUSION



I presented the business case for card payment fraud detection. I then used some basic exploratory data analysis techniques.

I then explained a simple autoencoder and created a very simple autoencoder in Keras that can reconstruct what non-fraudulent transactions looks like.