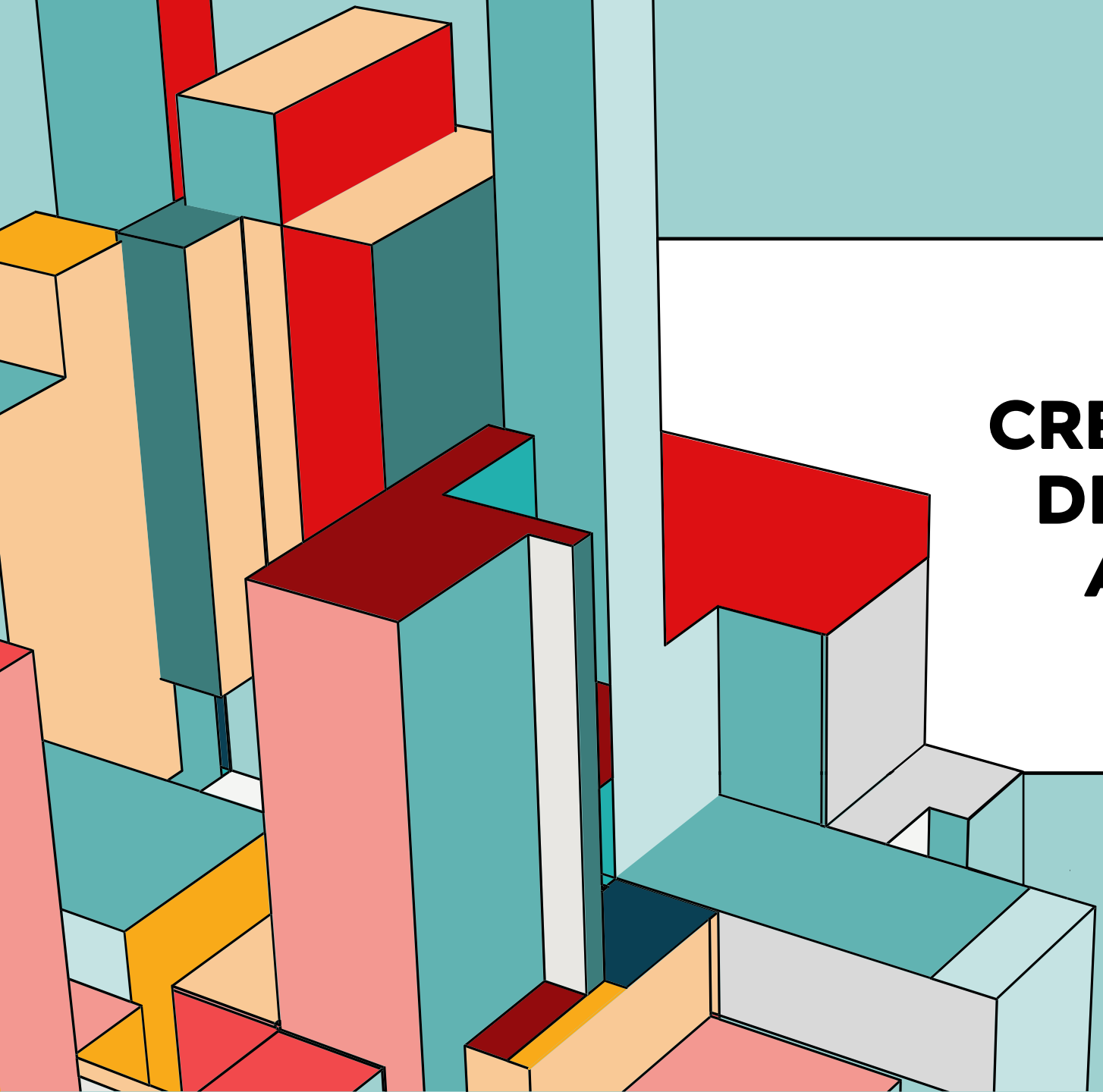


CREDIT CARD FRAUD DETECTION USING AUTOENCODER

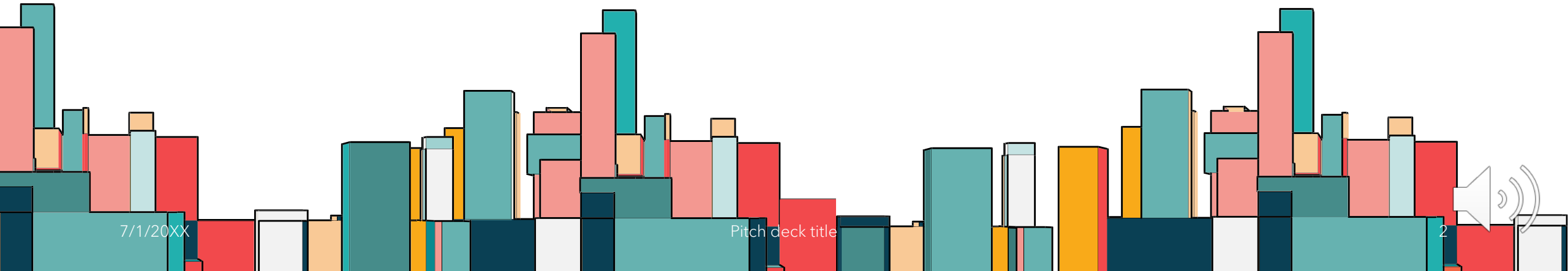
AGENDA

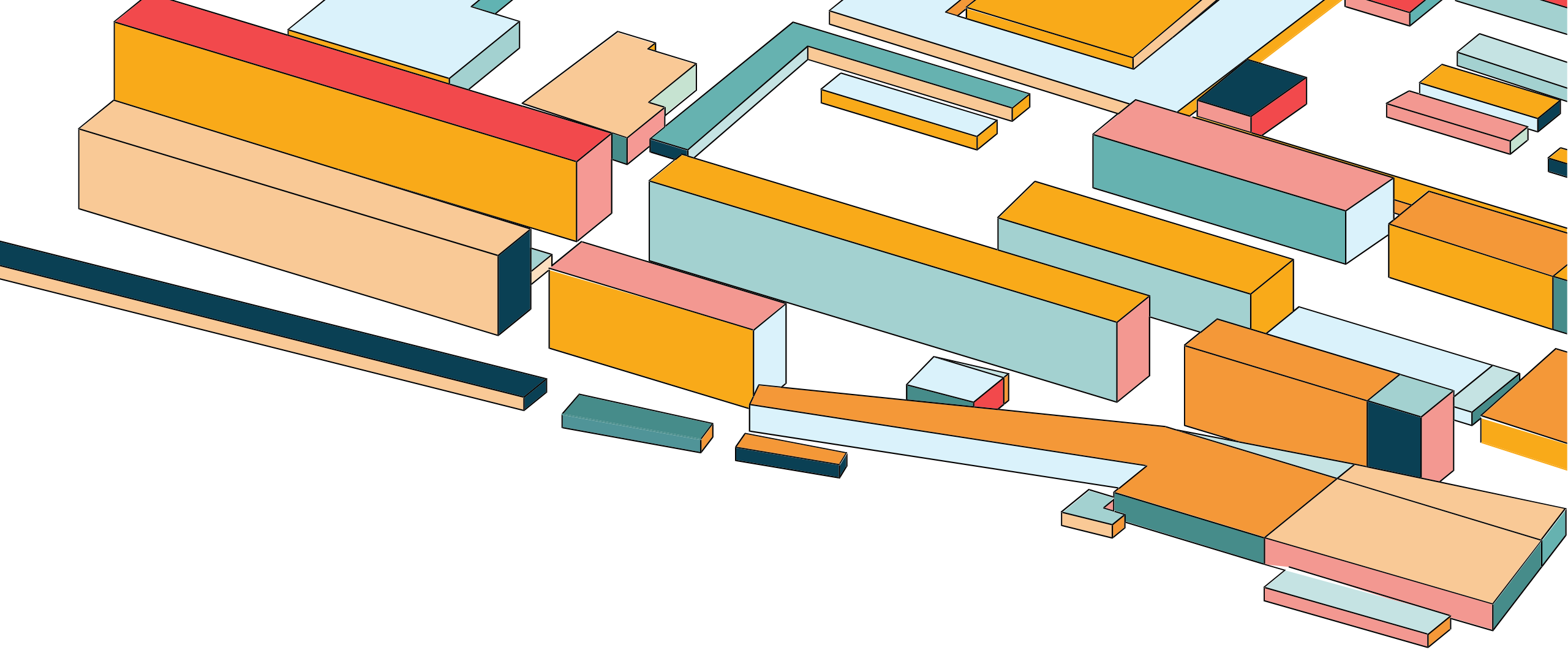
01 **BACKGROUND AND DATA**

02 **THEORY**

03 **PROCESSING**

04 **CONCLUSIONS**

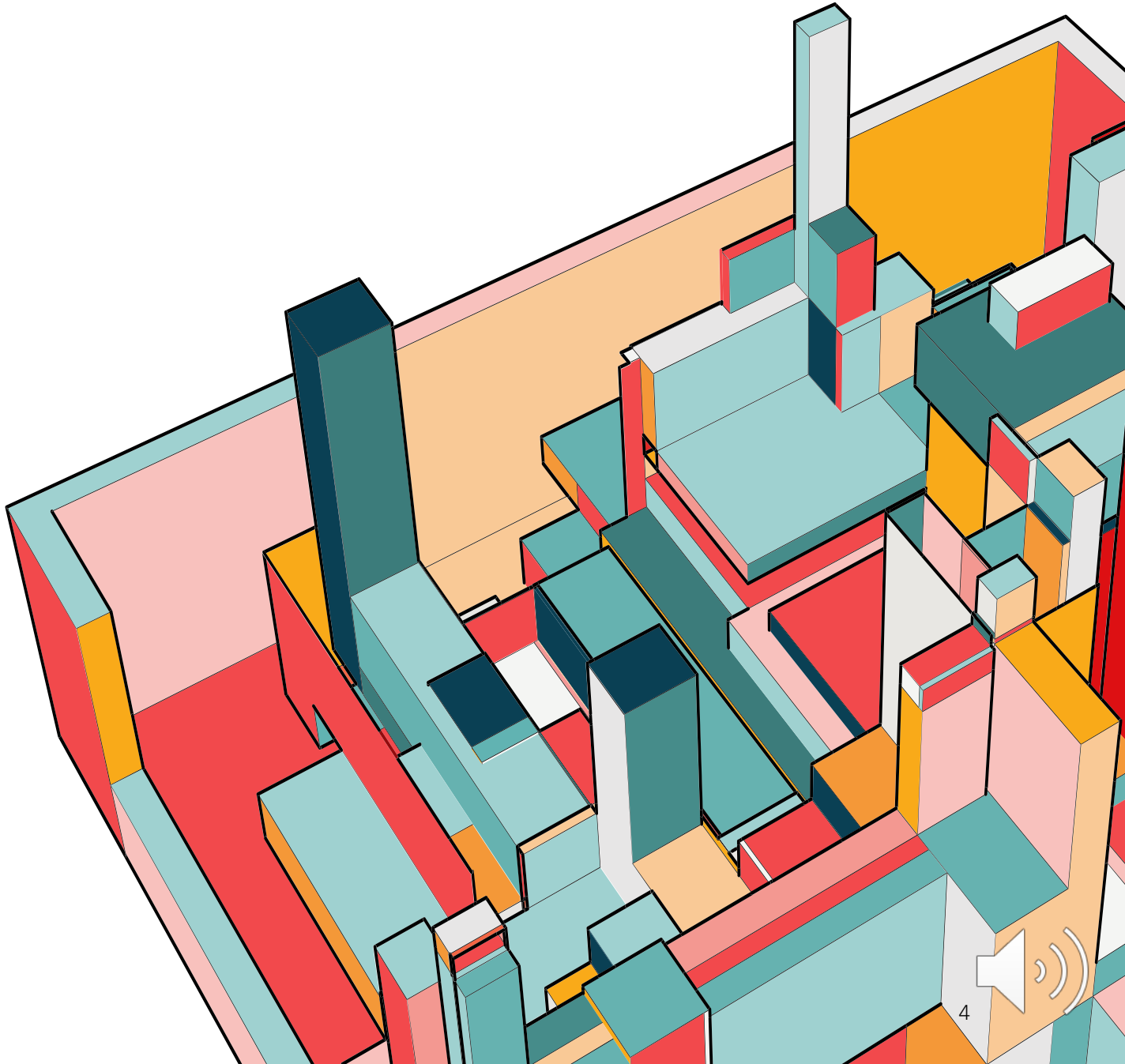


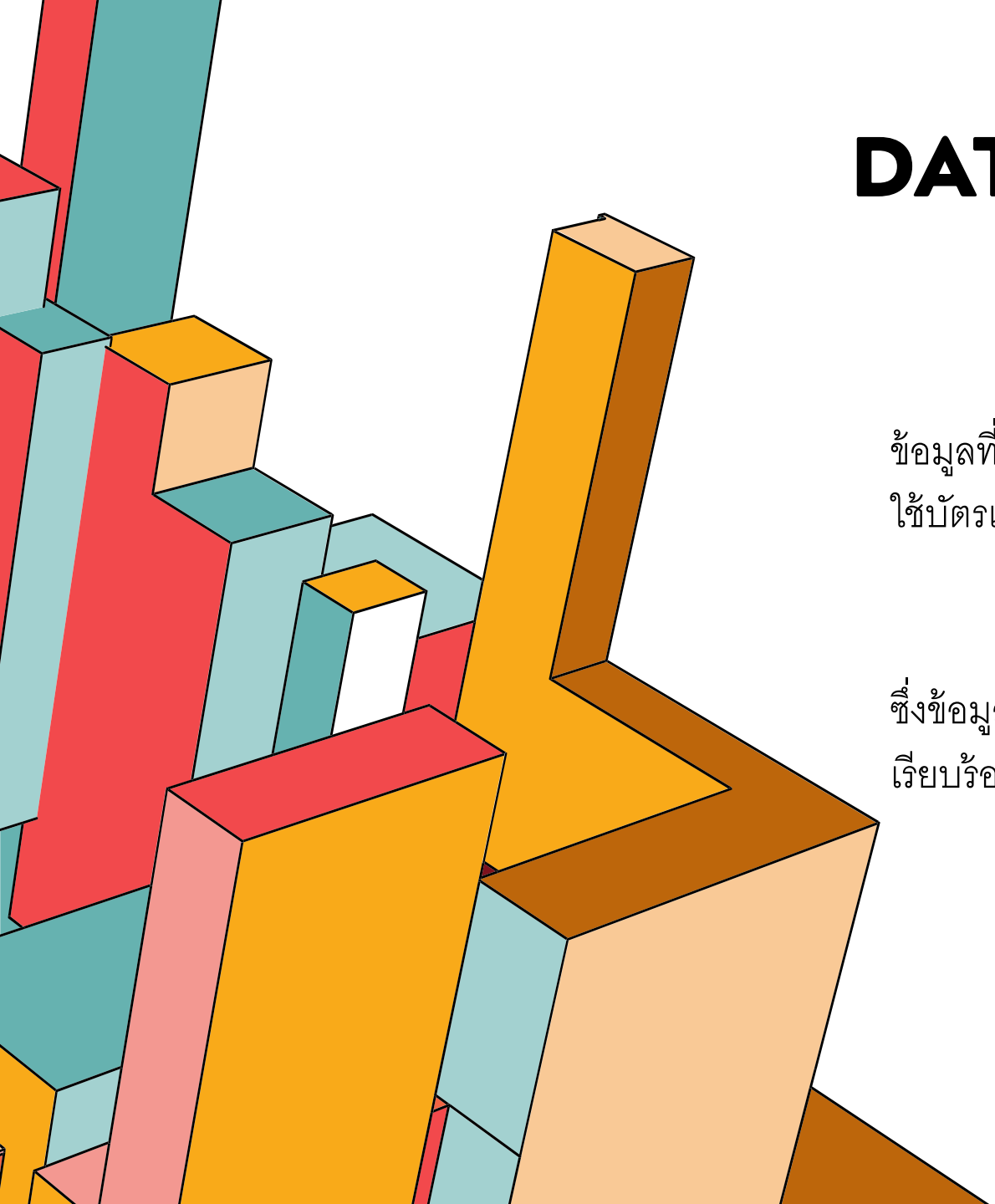


BACKGROUND AND DATA

BACKGROUND

เนื่องจากการตรวจสอบความผิดปกติของการใช้บัตรเครดิตนั้นอาจจะต้องใช้เวลาในการตรวจสอบต่างๆ จากผู้ที่ตรวจสอบจากทางธนาคารและอาจจะผิดพลาดจากความผิดพลาดของมนุษย์ (Human error) เราประยุกต์การใช้ Deep learning ในการช่วยค้นหาการใช้บัตรเครดิตที่มีความผิดปกติ





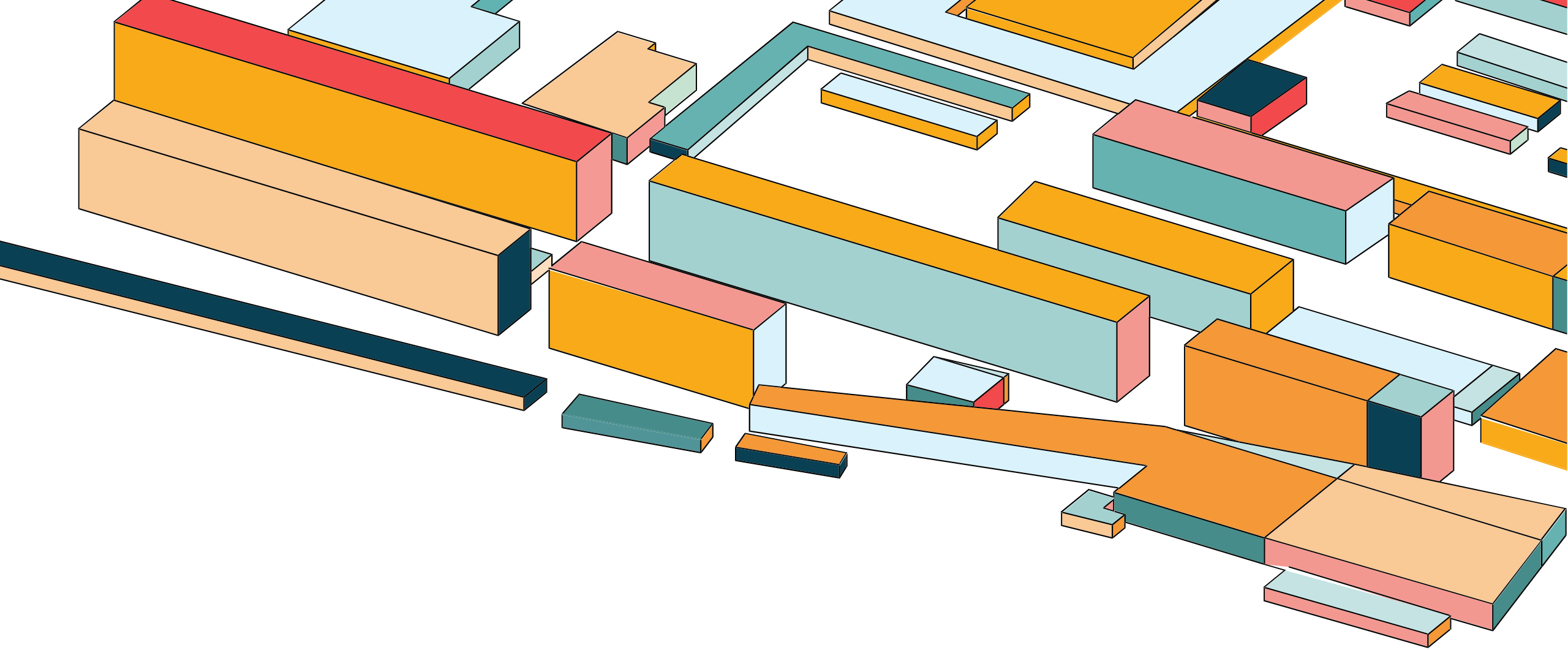
DATA

Credit Card using

ข้อมูลที่ใช้ในการวิเคราะห์นั้นจะเป็นประวัติการใช้บัตรเครดิตย้อนหลังของผู้ที่มีการใช้บัตรเครดิต โดยจะประกอบไปด้วย

1. ช่วงเวลาของการใช้งานบัตรเครดิต
2. จำนวนการถอนของการใช้งานในครั้งนั้นๆ

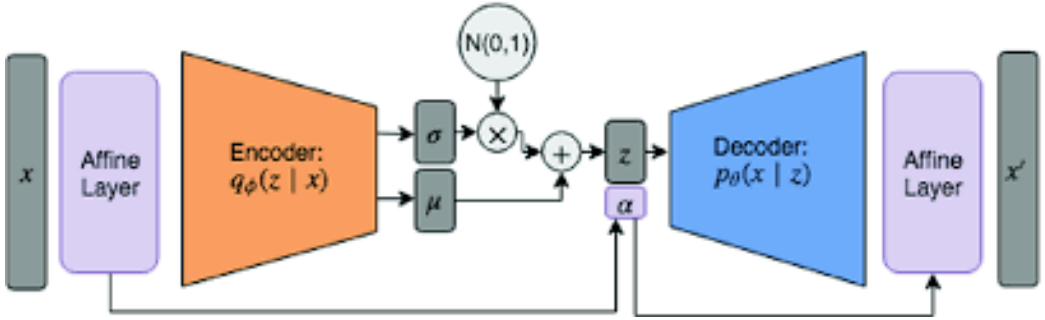
ซึ่งข้อมูลต่างๆได้ผ่านการทำ Principal component analysis : PCA เรียบร้อยแล้ว โดยจะได้ Features ทั้งหมด 28 ตัวแปร



THEORY

THEORY

Autoencoder



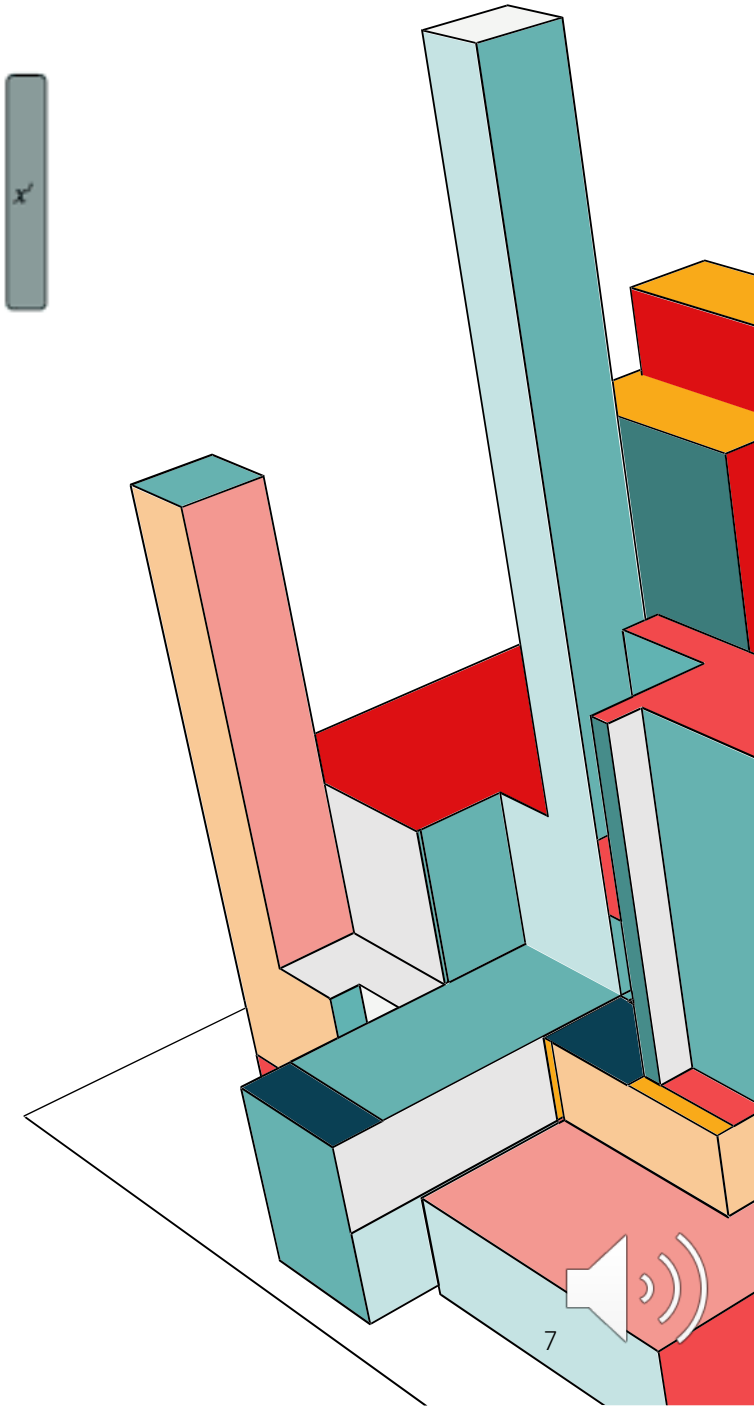
Autoencoder เป็น Neural Network แบบหนึ่งที่รับ Input เข้ามา แล้ว Output ออกมาเป็นข้อมูลเดิม แต่ความพิเศษก็คือ ที่ตรงกลางของ Network โดยทั่วไปแล้วจะบีบอัดข้อมูลให้มีขนาดเล็กลง โดยมีขั้นตอนดังนี้

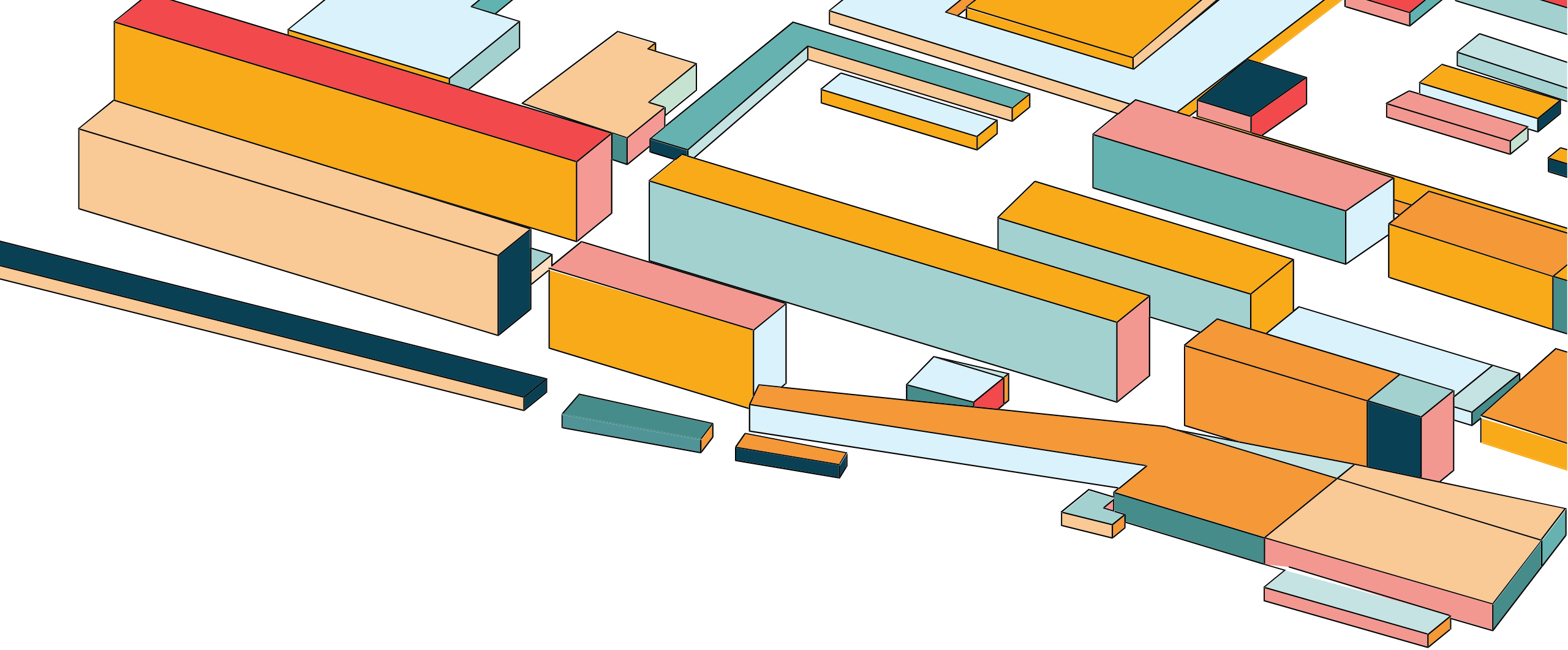
1. ตรวจสอบ Data ก่อนว่า Data มีความเป็น Imbalance อยู่มากๆ ซึ่งส่วนใหญ่ Anomaly ก็จะมีคุณลักษณะนี้

2. นำ Data ที่ได้กรองเอา Data ที่เป็น Anomaly ออกให้หมด และนำไป Train ใน Network ของเราโดยใช้ mean squared error loss function (MSE)

3.หลังจาก Train เสร็จแล้วเราจะได้ Model ที่มีค่า MSE ที่ต่ำมาก ๆ เมื่อนำ Data ที่เป็น Anomaly เข้าไป Test จะได้ค่า MSE ที่สูงมาก ๆ

4. หลังจากนั้นเราจึงตั้ง Threshold ขึ้นเพื่อแยก Anomaly และ Normally ออกจากกัน





PROCESSING

PROCESSING

Data Understanding

การทำความเข้าใจข้อมูลและตัวแปรต่างๆ ที่จะนำมาวิเคราะห์

Pre-processing

การเตรียมข้อมูลเพื่อให้พร้อมที่จะนำไปทำ การวิเคราะห์ต่อไปเพราะหากข้อมูลที่นำไปไม่ถูกต้องหรือข้อมูลที่ไม่เป็นประโยชน์นั้นอาจจะส่งผลต่อการวิเคราะห์ได้

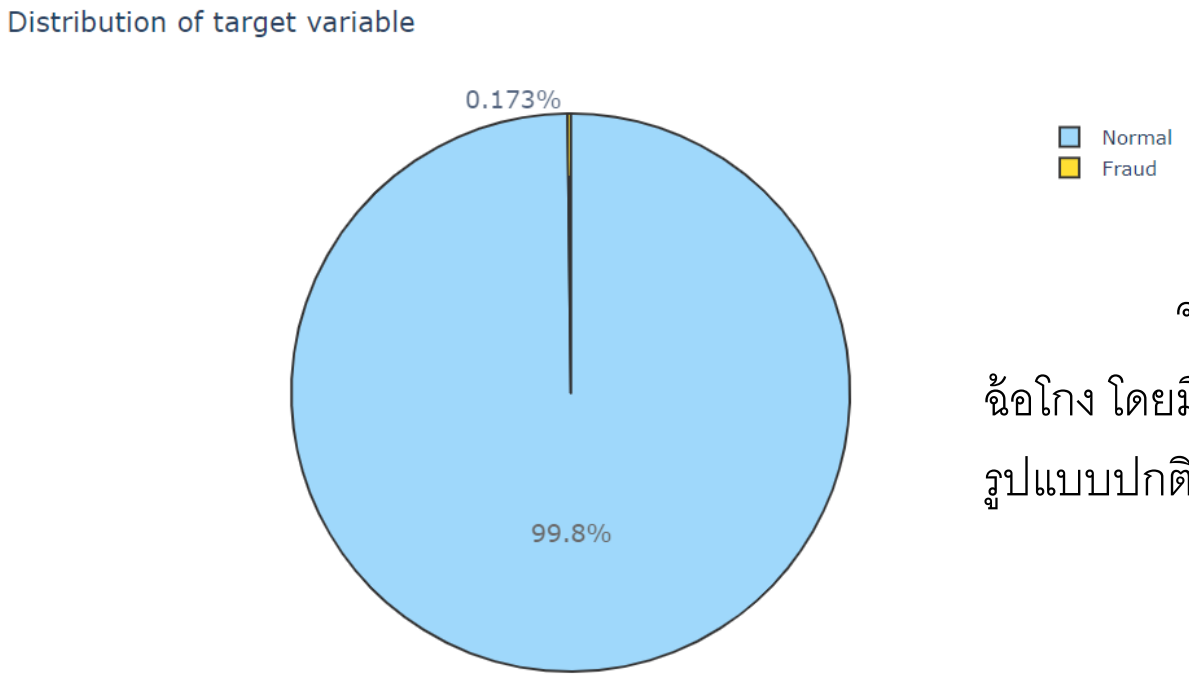
Modeling

การสร้างรูปแบบความสัมพันธ์ (Relational Pattern) และจะอยู่ในรูปของแบบจำลองบนซอฟต์แวร์ (Computer Model) หรือสมการความสัมพันธ์

Evalulation

การวัดประสิทธิภาพในการทำงานของโมเดล

DATA UNDERSTANDING



จากข้อมูลการใช้งานบัตรเครดิต ทั้งรูปแบบปกติ และฉ้อโกง โดยมีข้อมูลทั้งหมด 284,807 รายการ โดยมีสัดส่วนที่ใช้งานในรูปแบบปกติอยู่ที่ 99.8 % และ ใช้งานในรูปแบบฉ้อโกงอยู่ที่ 0.173 %

PRE-PROCESSING

Normalization

```
[ ]  from sklearn import preprocessing
     from sklearn.model_selection import train_test_split
     from sklearn.metrics import confusion_matrix , roc_auc_score, roc_curve

[ ]  min_max_scaler = preprocessing.MinMaxScaler()
     df_cred=df_cred.drop("Time",axis=1)
     df_cred_scaled = min_max_scaler.fit_transform(df_cred.iloc[:,:-1])
     df_cred_normalized = pd.DataFrame(df_cred_scaled)

[ ]  df_cred_normalized["Class"]=df_cred["Class"]

[ ]  df_cred_normalized["Class"].value_counts()
```

PRE-PROCESSING

Splitting strategy

```
[ ]  df_cred_normalized_test_part_1=df_cred_normalized_train.sample(frac=0.05)
     df_cred_normalized_train=df_cred_normalized_train.drop(df_cred_normalized_test_part_1.index)
     df_cred_normalized_test_part_2=df_cred_normalized_train.sample(frac=0.05)
     df_cred_normalized_train=df_cred_normalized_train.drop(df_cred_normalized_test_part_2.index)
```

Merging

```
[ ]  df_cred_normalized_test_set=df_cred_normalized_test_part_1.append(df_cred_normalized_test_class_1)
     df_cred_normalized_validation_set=df_cred_normalized_test_part_2.append(df_cred_normalized_validation_class_1)
```

PRE-PROCESSING

Splitting for train set and test set

```
X_train, X_test = train_test_split(df_cred_normalized_train, test_size=0.2, random_state=2020)
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
X_train.shape
```

MODELING

Setting Encoder and Decoder

```
input_dim = X_train.shape[1]
encoding_dim = 20
input_layer = Input(shape=(input_dim, ))
encoder = Dense(encoding_dim*2, activation="sigmoid")(input_layer)
encoder = Dense(encoding_dim, activation="sigmoid")(input_layer)
encoder = Dense(8,activation="sigmoid")(encoder)
decoder = Dense(20, activation='sigmoid')(encoder)
decoder = Dense(40, activation='sigmoid')(encoder)
decoder = Dense(input_dim, activation='sigmoid')(decoder)
autoencoder = Model(inputs=input_layer, outputs=decoder)
```

MODELING

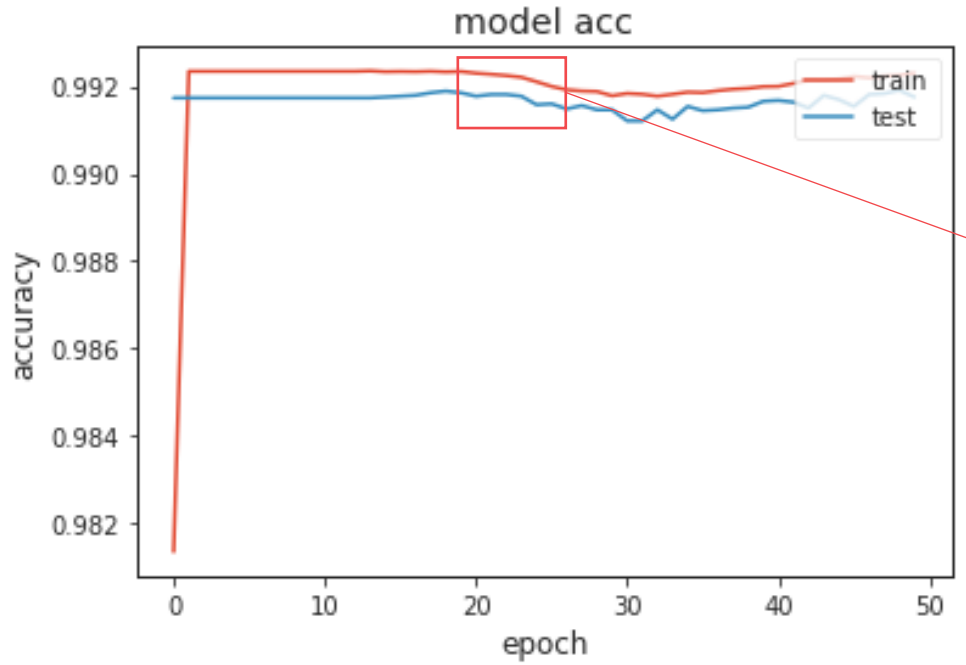
Setting Hyperparameters

```
nb_epoch = 50
batch_size = 32
autoencoder.compile(optimizer='adam',
                    loss='mean_squared_error',
                    metrics=['accuracy'])
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1,patience=15)

checkpointer = ModelCheckpoint(filepath="model.h5",
                               verbose=0,
                               save_best_only=True)
history = autoencoder.fit(X_train, X_train,
                    epochs=nb_epoch,
                    batch_size=batch_size,
                    shuffle=True,
                    validation_data=(X_test, X_test),callbacks=[es,checkpointer],
                    verbose=1)
```

MODELING

Result



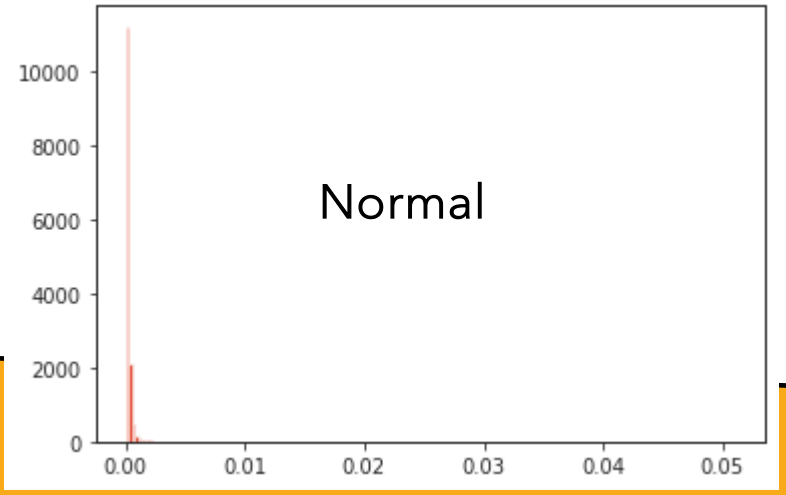
Epochs = 22
Accuracy of 99.23%

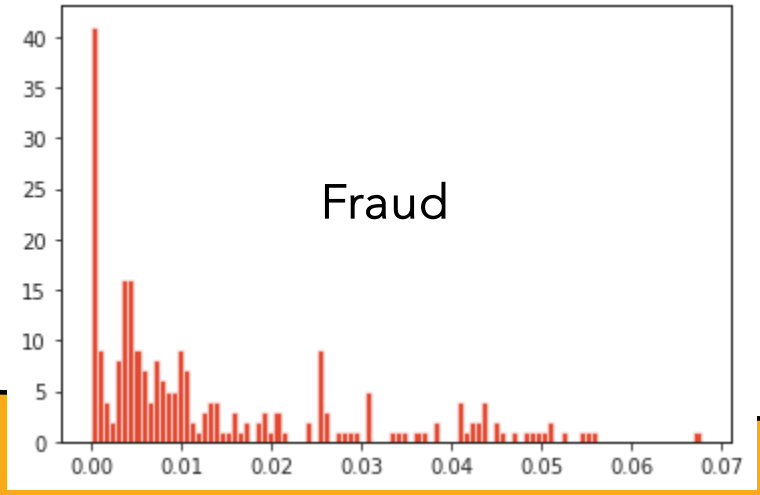
EVALUATION

Using Mean square error: MSE

```
y_test=df_cred_normalized_test_set["Class"]
df_cred_normalized_test_set=df_cred_normalized_test_set.drop("Class",axis=1)

predictions = autoencoder.predict(df_cred_normalized_test_set)
mse = np.mean(np.power(df_cred_normalized_test_set - predictions, 2), axis=1)
error_df_test = pd.DataFrame({'reconstruction_error': mse,
                              'true_class': y_test})
error_df_test.describe()
```



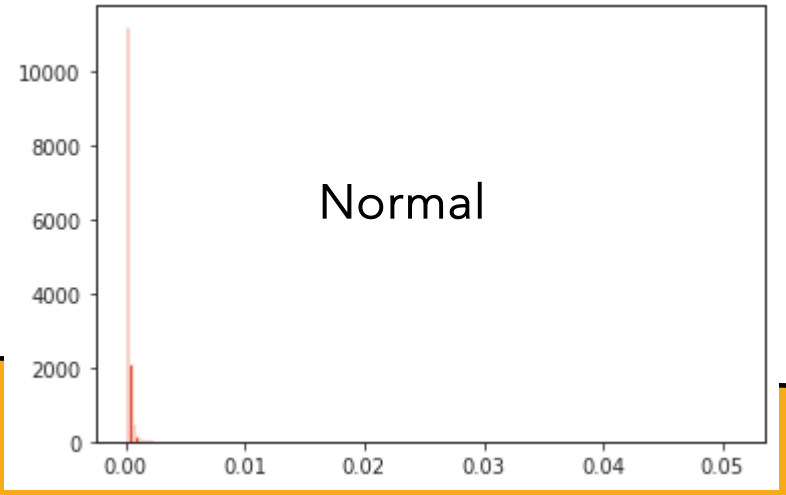


EVALUATION

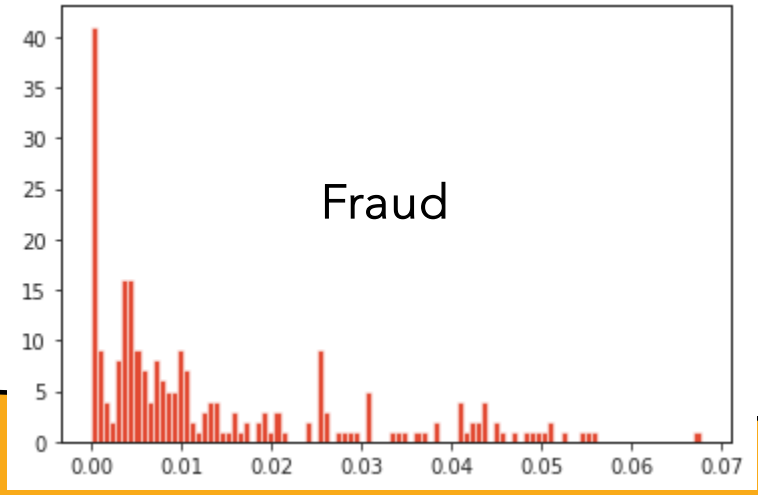
Using Mean square error: MSE

```
y_test=df_cred_normalized_test_set["Class"]
df_cred_normalized_test_set=df_cred_normalized_test_set.drop("Class",axis=1)

predictions = autoencoder.predict(df_cred_normalized_test_set)
mse = np.mean(np.power(df_cred_normalized_test_set - predictions, 2), axis=1)
error_df_test = pd.DataFrame({'reconstruction_error': mse,
                              'true_class': y_test})
error_df_test.describe()
```



Normal



Fraud

EVALUATION

Using Mean square error: MSE

```
fraud_error_df.describe() ### frauds cases
```

	reconstruction_error	true_class
count	246.000000	246.0
mean	0.013627	1.0
std	0.015128	0.0
min	0.000051	1.0
25%	0.003276	1.0
50%	0.007354	1.0
75%	0.020510	1.0
max	0.067655	1.0

```
[ ] normal_error_df.describe() ### non fraud cases
```

	reconstruction_error	true_class
count	14216.000000	14216.0
mean	0.000249	0.0
std	0.000773	0.0
min	0.000018	0.0
25%	0.000092	0.0
50%	0.000148	0.0
75%	0.000248	0.0
max	0.051072	0.0

EVALUATION

Decided evaluation metrics over the threshold

```
[ ]  error_df_test["predicted_class"]=[1 if x > 0.001 else 0 for x in error_df_test["reconstruction_error"]]
```

```
error_df_test["predicted_class"]=[1 if x >  0.004  else 0 for
x in error_df_test["reconstruction_error"]]
```

```
error_df_test["predicted_class"]=[1 if x > 0.0039888 else 0 for
 x in error_df_test["reconstruction_error"]]
```

```
error_df_test["predicted_class"]=[1 if x > 0.003 else 0 for x in
 error_df_test["reconstruction_error"]]
```

EVALUATION

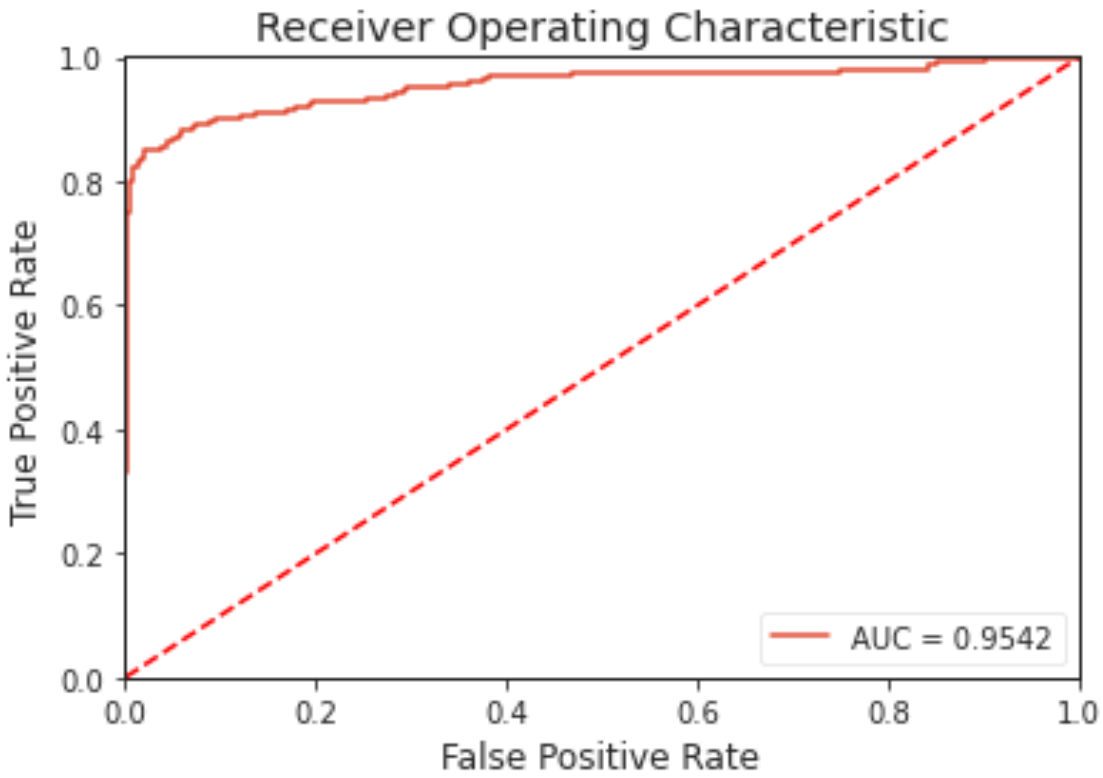
Final evaluation

```
[ ]  error_df_test["predicted_class"]=[1 if x > 0.003 else 0 for x in error_df_test["reconstruction_error"]]
```

```
LABELS = ["Normal", "Fraud"]
y_pred = [1 if e >  0.00398888 else 0 for e in error_df_test.reconstruction_error.values]
conf_matrix = confusion_matrix(error_df_test.true_class,error_df_test.predicted_class)
plt.figure(figsize=(8, 8))
sns.heatmap(conf_matrix, xticklabels=LABELS, yticklabels=LABELS, annot=True, fmt="d");
plt.title("Confusion matrix")
plt.ylabel('True class')
plt.xlabel('Predicted class')
plt.show()
```

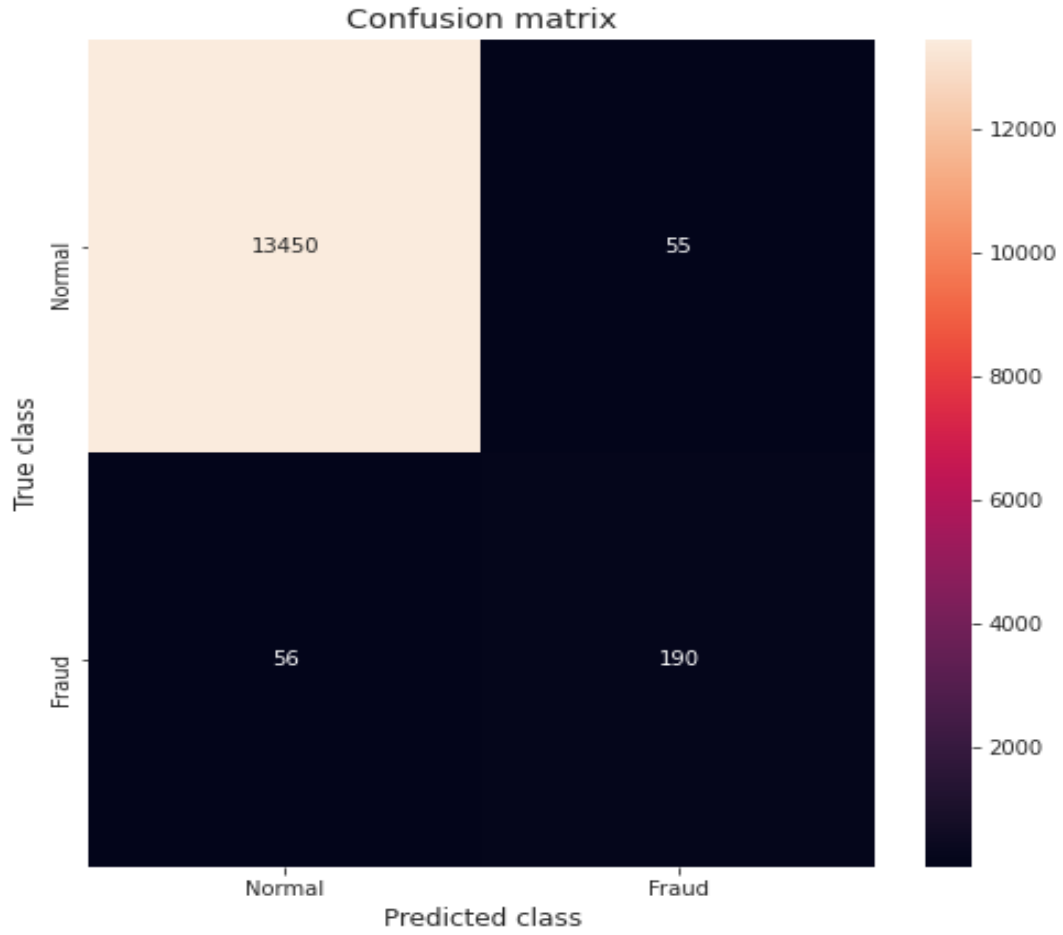
EVALUATION

Result



EVALUATION

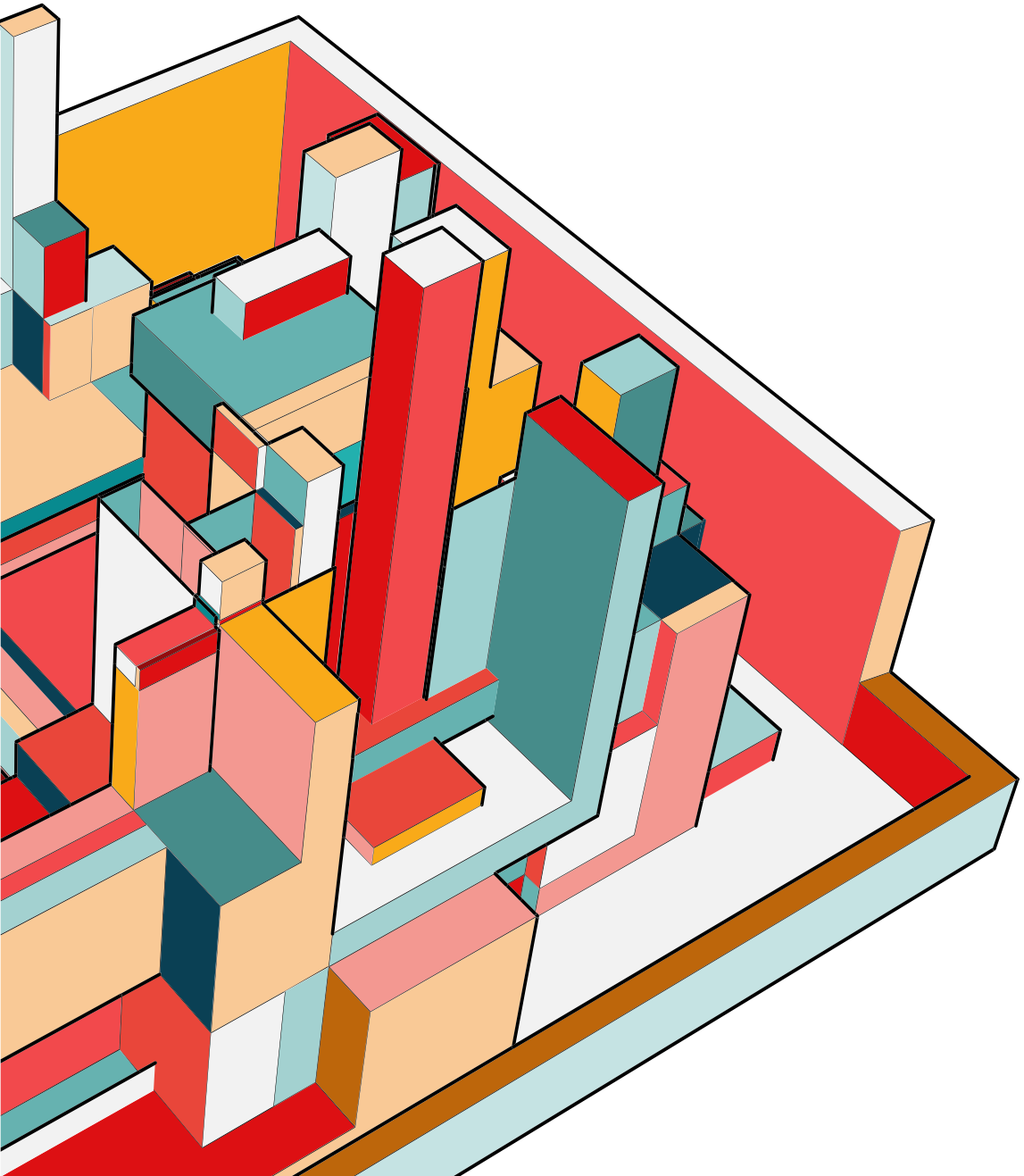
Result



EVALUATION

Result

	precision	recall	f1-score	support
0	1.00	1.00	1.00	13505
1	0.78	0.77	0.77	246
accuracy			0.99	13751
macro avg	0.89	0.88	0.88	13751
weighted avg	0.99	0.99	0.99	13751



CONCLUSIONS

จากผลลัพธ์จากโปรแกรมจะเห็นว่าการทำนายความผิดปกติด้วย Autoencoder นั้นมีความแม่นยำอยู่ที่ 0.99 ซึ่งสูงมาก ดังนั้นในการแบ่งประเภทของการใช้บัตรเครดิตครั้งถัด ๆไปขอธนาคารอาจจะใช้ Model นี้เข้ามาช่วยในการจำแนก ซึ่งในส่วนของที่ทำนายผิด 0.01 อาจจะต้องนำส่วนนี้มาพิจารณาเพิ่มเติมเนื่องจากหาก ผู้คนส่วนนี้ไม่สามารถใช้บัตรเครดิตได้จากการทำนายที่ผิดของ model อาจจะทำให้ธนาคารเสียชื่อเสียง รวมถึงอาจจะเสียลูกค้าได้ ซึ่งอาจจะแก้ไขโดยหากการทำนายจาก model นั้นตกอยู่ในส่วนของผิดปกติ อาจจะส่งข้อมูลนี้ไปให้ผู้วิเคราะห์อีกครั้งหนึ่งเพื่อพิจารณาระงับการใช้บัตรเครดิต