Valéo: Anomaly detection in Industry 4.0

Nouamane Arhachoui
Werner Dia Black
Julio Guzmán

TELECOM SudParis

INSTITUT POLYTECHNIQUE DE PARIS

Summary

- Introduction
- Pre-Processing
- Logistic Regression
- Neural Networks
- Ensemble Learning
- Unsupervised Learning
- Conclusion

Introduction

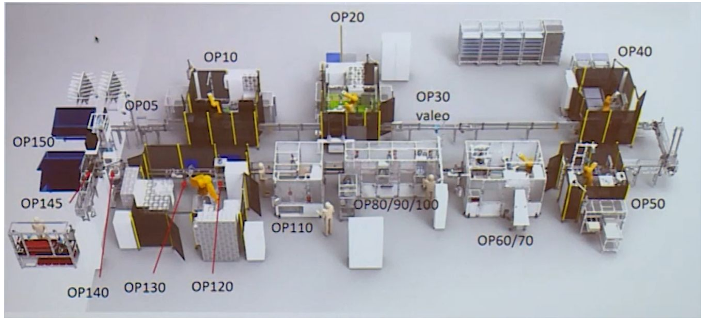
Valeo & the problem at hand

Introduction

Machine Learning applied to the Industry

- Production Line
- 15 stations
- One exit a motor

The Issue at Hand



Introduction

The Dataset

- 13 features/engine
- One output
- 30000+ entries

Many features

All features are related to data measured on a specific automaton.

One Goal

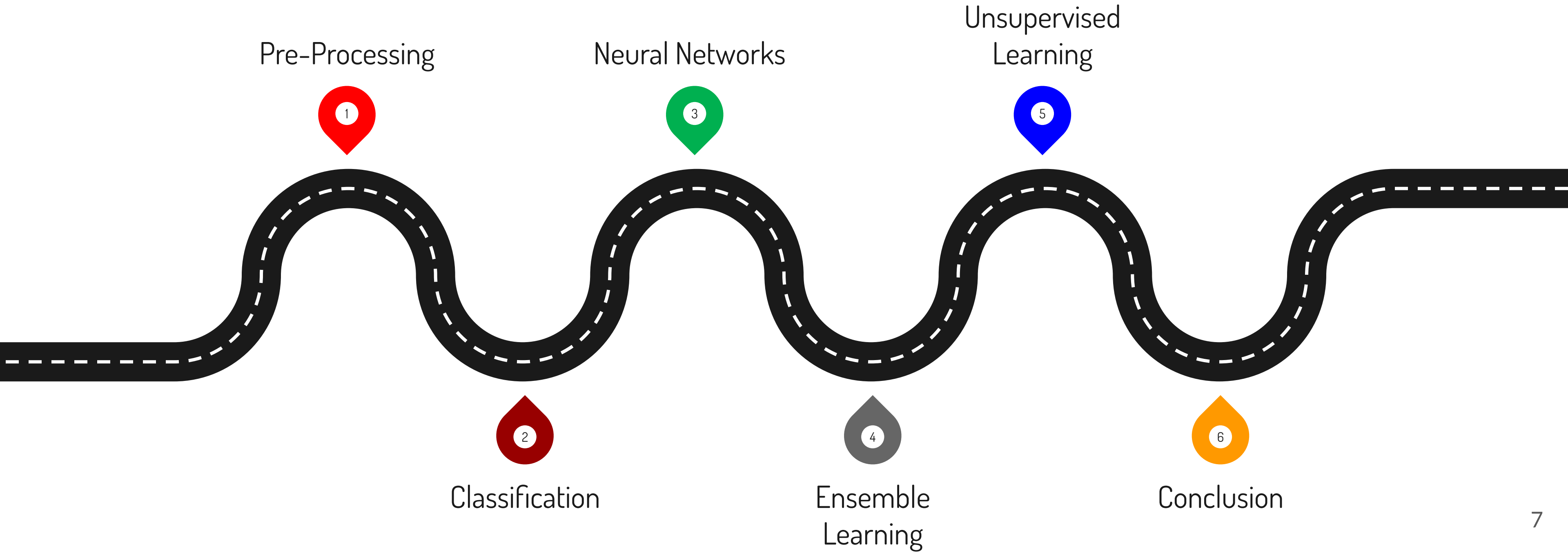
- Input analysis
  - Features
  - Correlation
- To achieve a predictive Model:

  F(input) = output

Introduction

```
Data columns (total 15 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   PROC_TRACEINFO                  34515 non-null  object
 1   OP070_V_1_angle_value           34515 non-null  float64
 2   OP090_SnapRingPeakForce_value   34515 non-null  float64
 3   OP070_V_2_angle_value           34515 non-null  float64
 4   OP120_Rodage_I_mesure_value     34515 non-null  float64
 5   OP090_SnapRingFinalStroke_value 34515 non-null  float64
 6   OP110_Vissage_M8_torque_value   34515 non-null  float64
 7   OP100_Capuchon_insertion_mesure 15888 non-null  float64
 8   OP120_Rodage_U_mesure_value     34515 non-null  float64
 9   OP070_V_1_torque_value          34515 non-null  float64
 10  OP090_StartLinePeakForce_value  34515 non-null  float64
 11  OP110_Vissage_M8_angle_value    34515 non-null  float64
 12  OP090_SnapRingMidPointForce_val 34515 non-null  float64
 13  OP070_V_2_torque_value          34515 non-null  float64
 14  results                         34515 non-null  int64
```

Introduction



Pre-Processing

Adapting our Dataset to our needs

Pre-Processing

First Look

We do not take into account the column:

OP100_Capuchon_insertion_mesure with the missing values (53% is to many missing values)

```
df.columns[df.isnull().any()].tolist()
#Output : ['OP100_Capuchon_insertion_mesure']

print("Percentage of missing values =
",round(df['OP100_Capuchon_insertion_mesure'].isna().sum()*100/34514,2),"%")
#Output: Percentage of missing values =  53.97 %

df_cleaned = df.drop(columns= ['OP100_Capuchon_insertion_mesure'])
```

Other observation

36% of the failing engines have missing values

⇒ keep the column by putting binary values in it (is the value missing or not)

```
missing_capuchon = df['OP100_Capuchon_insertion_mesure'].isna()
missing_capuchon = missing_capuchon.astype(np.int)
df['OP100_Capuchon_insertion_mesure'] = missing_capuchon
```

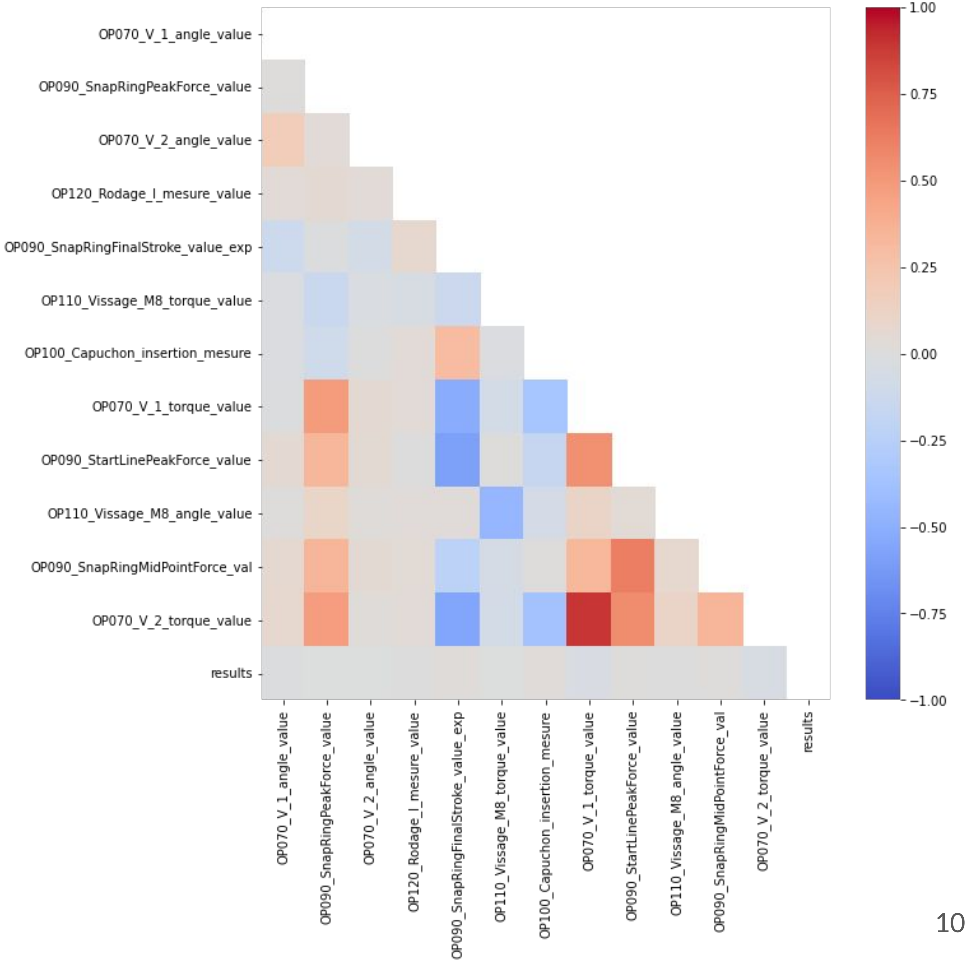
Pre-Processing

Statistical analysis

- Compute various statistical values for the features
- Hard to evaluate since the meaning of the dataset is unknown

Correlation

- Pearson's Correlation number gives us few correlation between parameters



Pre-Processing

Link between features

Without “capuchon_mesure”

```python
values_corr = feature_engineering.get_most_correlated(df_cleaned)
values_corr[:20]
```

```
OP070_V_1_torque_value           OP070_V_2_torque_value             0.897
OP090_StartLinePeakForce_value   OP090_SnapRingMidPointForce_val    0.621
                                 OP070_V_2_torque_value             0.562
OP070_V_1_torque_value           OP090_StartLinePeakForce_value     0.543
OP090_SnapRingPeakForce_value    OP070_V_1_torque_value             0.490
                                 OP070_V_2_torque_value             0.482
OP110_Vissage_M8_torque_value    OP110_Vissage_M8_angle_value      -0.446
OP090_SnapRingFinalStroke_value  OP070_V_2_torque_value            -0.408
                                 OP090_StartLinePeakForce_value    -0.381
                                 OP070_V_1_torque_value            -0.381
OP090_SnapRingMidPointForce_val  OP070_V_2_torque_value             0.347
OP090_SnapRingPeakForce_value    OP090_SnapRingMidPointForce_val    0.345
                                 OP090_StartLinePeakForce_value     0.337
OP070_V_1_torque_value           OP090_SnapRingMidPointForce_val    0.335
OP070_V_1_angle_value            OP070_V_2_angle_value              0.187
OP120_Rodage_U_mesure_value      OP070_V_2_torque_value             0.172
                                 OP070_V_1_torque_value             0.169
OP090_SnapRingPeakForce_value    OP120_Rodage_U_mesure_value        0.135
                                 OP110_Vissage_M8_torque_value     -0.135
OP120_Rodage_I_mesure_value      OP120_Rodage_U_mesure_value       -0.119
```

With “capuchon_mesure” as binary

```
OP070_V_1_torque_value              OP070_V_2_torque_value             0.897
OP100_Capuchon_insertion_mesure     OP070_V_2_torque_value             0.668
                                    OP070_V_1_torque_value             0.665
OP090_StartLinePeakForce_value      OP090_SnapRingMidPointForce_val    0.621
OP090_SnapRingFinalStroke_value_exp OP090_StartLinePeakForce_value    -0.593
OP090_StartLinePeakForce_value      OP070_V_2_torque_value             0.562
OP090_SnapRingFinalStroke_value_exp OP070_V_2_torque_value            -0.559
OP070_V_1_torque_value              OP090_StartLinePeakForce_value     0.543
OP090_SnapRingFinalStroke_value_exp OP070_V_1_torque_value            -0.523
OP090_SnapRingPeakForce_value       OP100_Capuchon_insertion_mesure    0.519
                                    OP070_V_1_torque_value             0.490
                                    OP070_V_2_torque_value             0.482
OP110_Vissage_M8_torque_value       OP110_Vissage_M8_angle_value      -0.446
OP100_Capuchon_insertion_mesure     OP090_StartLinePeakForce_value     0.350
OP090_SnapRingMidPointForce_val     OP070_V_2_torque_value             0.347
OP090_SnapRingPeakForce_value       OP090_SnapRingMidPointForce_val    0.345
                                    OP090_StartLinePeakForce_value     0.337
OP070_V_1_torque_value              OP090_SnapRingMidPointForce_val    0.335
OP100_Capuchon_insertion_mesure     OP090_SnapRingMidPointForce_val    0.250
OP090_SnapRingFinalStroke_value_exp OP090_SnapRingMidPointForce_val   -0.223
```

Logistic Regression

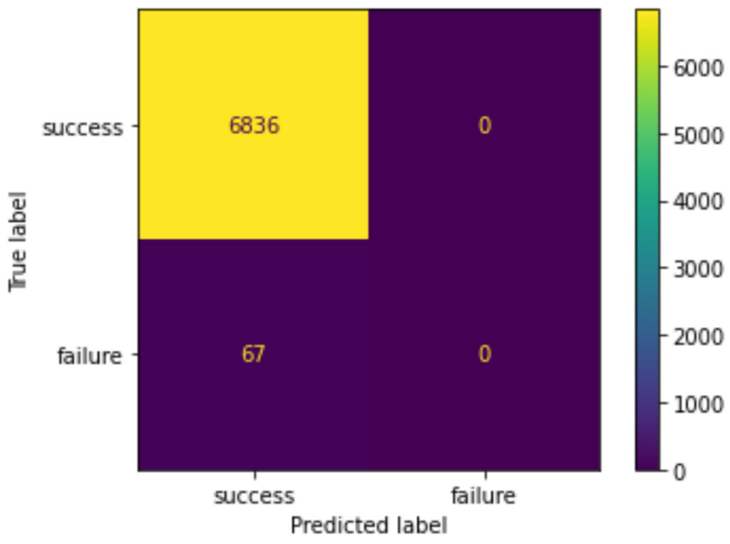
Logistic Regression

Binary Classification

We only have two ***labels*** :

Pass or Not.

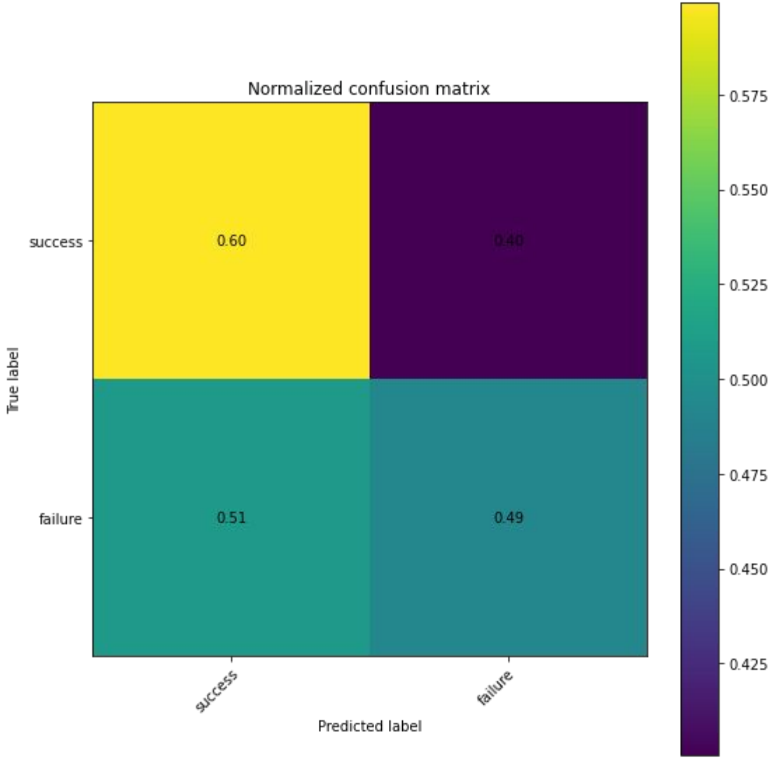
First, we tried Logistic Regression without regularization.



⇒ everything is predicted as a success engine

The accuracy is good but the model is bad regarding our goal

Logistic Regression



Scaled dataset	yes
Balanced dataset	yes

Only 0.9% of our Dataset is failures

⇒ SMOTE in order to balance the classes

accuracy = 0.64

idea: Grid Search Cross Validation to find the best regularization parameter to improve the model

Logistic Regression

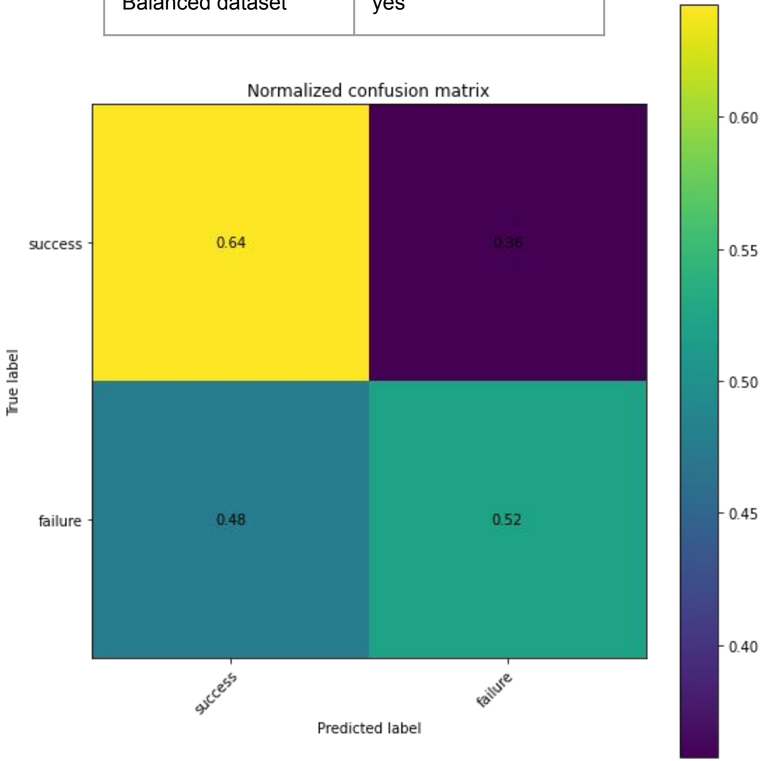
Scaled dataset	yes
Balanced dataset	yes

Grid Search CV

We found the best regularization weight:

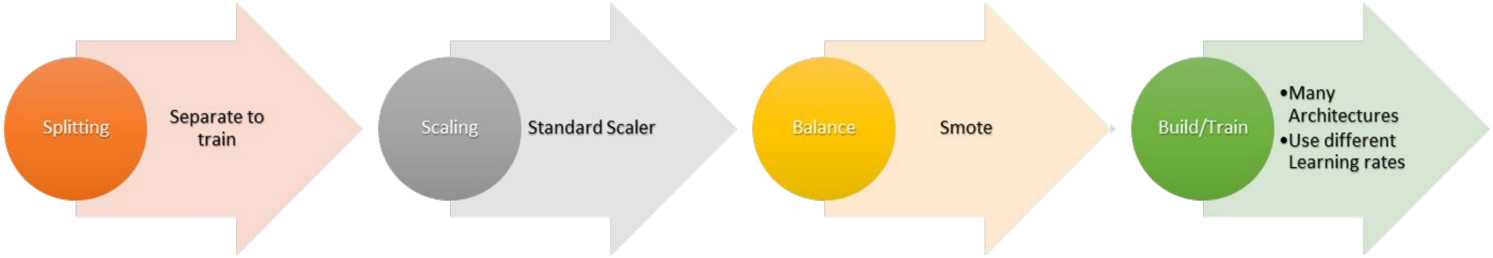
C = 0.1

The recall on each class is a little bit better



Neural Networks

Neural Networks



Neural Networks

Building a Model

- Sequential Model
- Relu as activation function and SoftMax for output
- 3 types of architecture :
    - Shallow
    - Wide
    - Deep

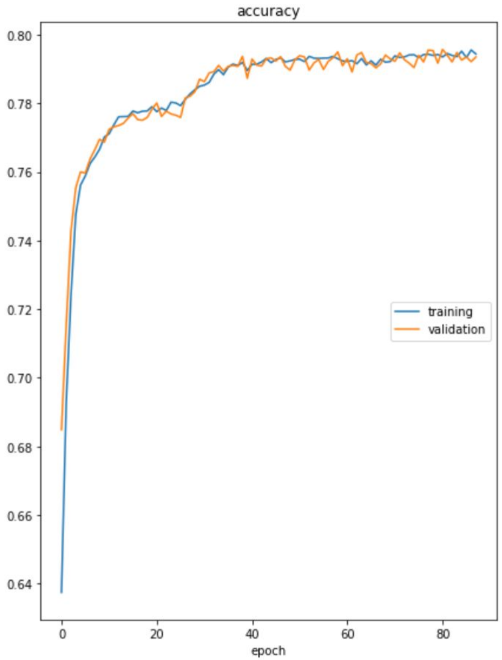
Training the model

- Modify Learning Rate
- Reach the Highest accuracy
- Then apply the prediction

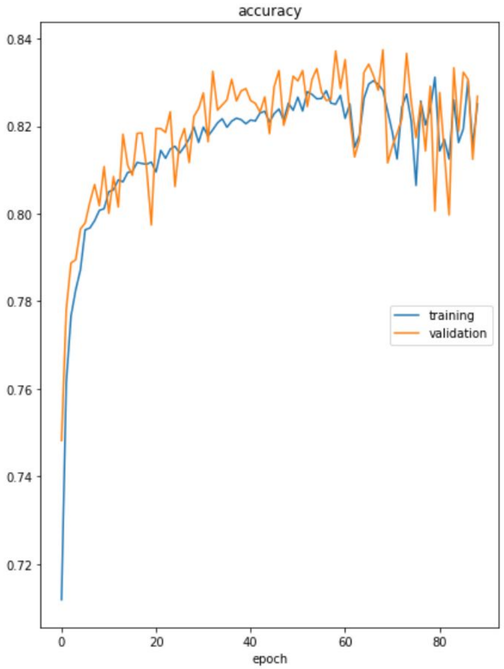
```
def trainmodel(learning_rate):
  model = make_sequential_model(sample_size, shallow_architecture,
                                learning_rate=0.001)
  nn_file = my_path + 'nn-'+str(learning_rate)+'.h5'
  model.summary

  history = train_model(model, nn_file, X_train_balanced, y_train1, seed=5,
max_epochs=200)
```

Neural Networks

Shallow[8,8,8,8,2]

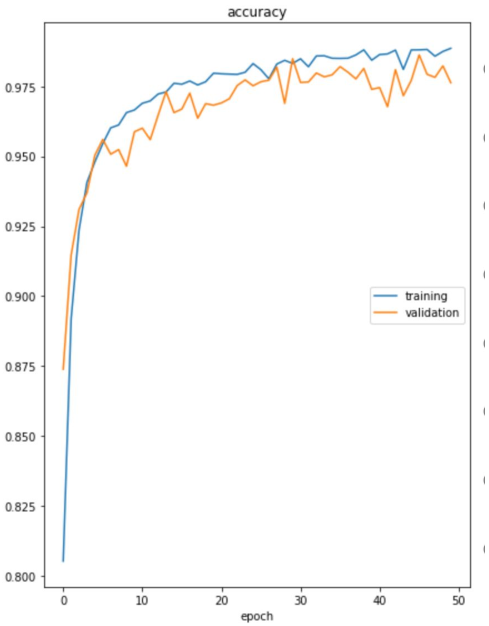


Deeper[8, 8, 8, 8, 8, 8, 8, 8, 8, 2]

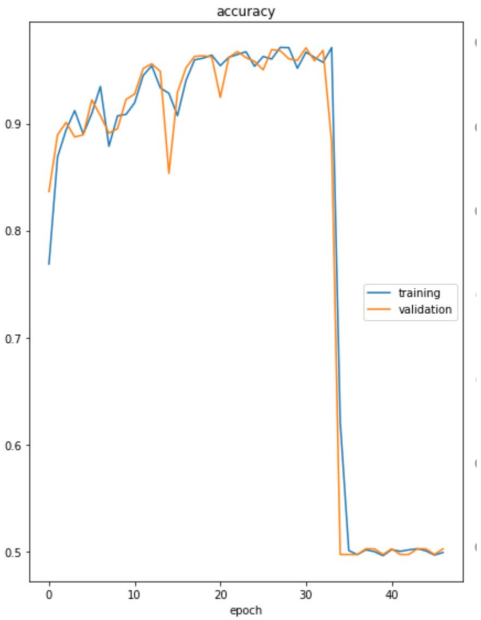


Neural Networks

Wide [72, 72,72,2]

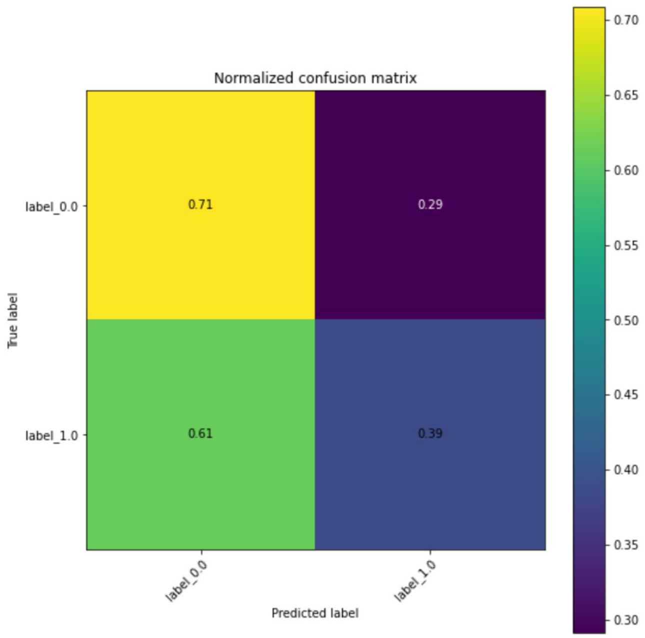


Wide and Deeper[48, 48,48,48,48,48,48,48,48,2]

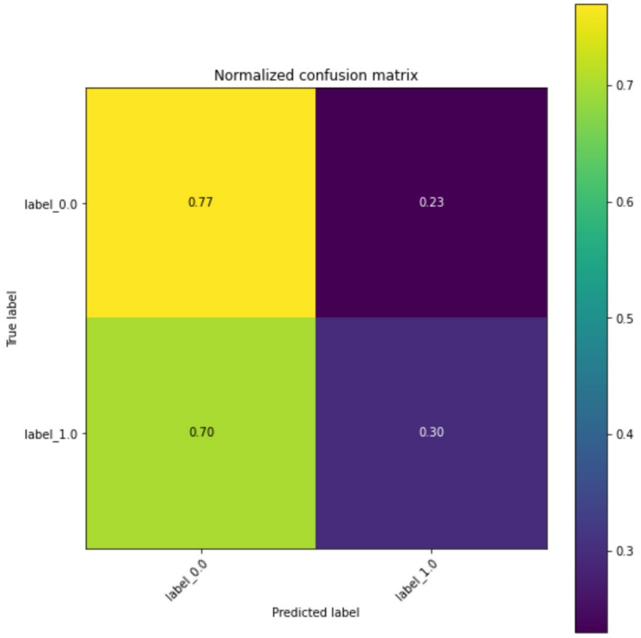


Neural Networks

For The Shallow Network [8,8,8,8,2]

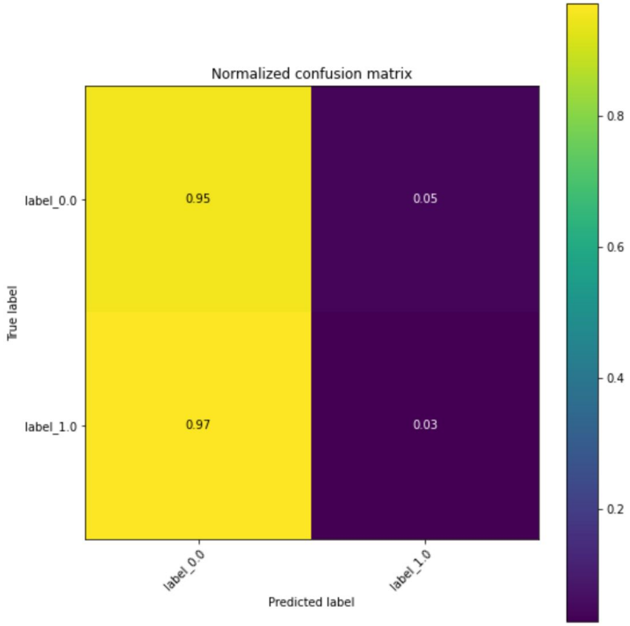


For our Deeper[8, 8 ,8, 8, 8, 8, 8, 8, 8, 2]

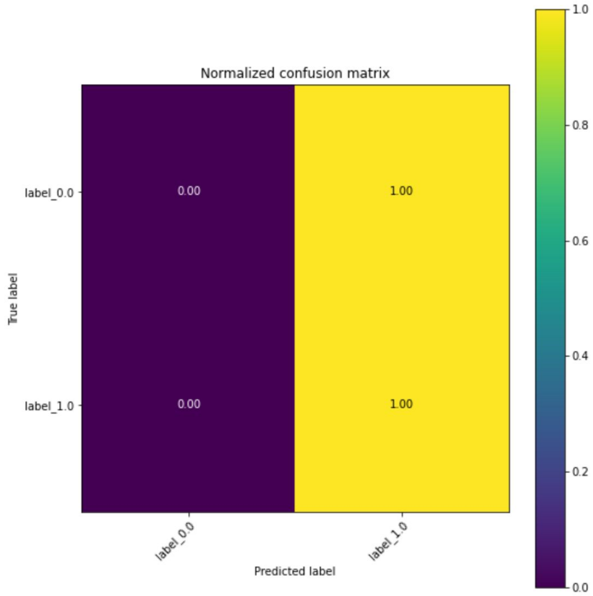


Neural Networks

For The Wide [72, 72,72,2]



For our Hybrid Network [48,48,48,48,48,48,48,48,48,2]



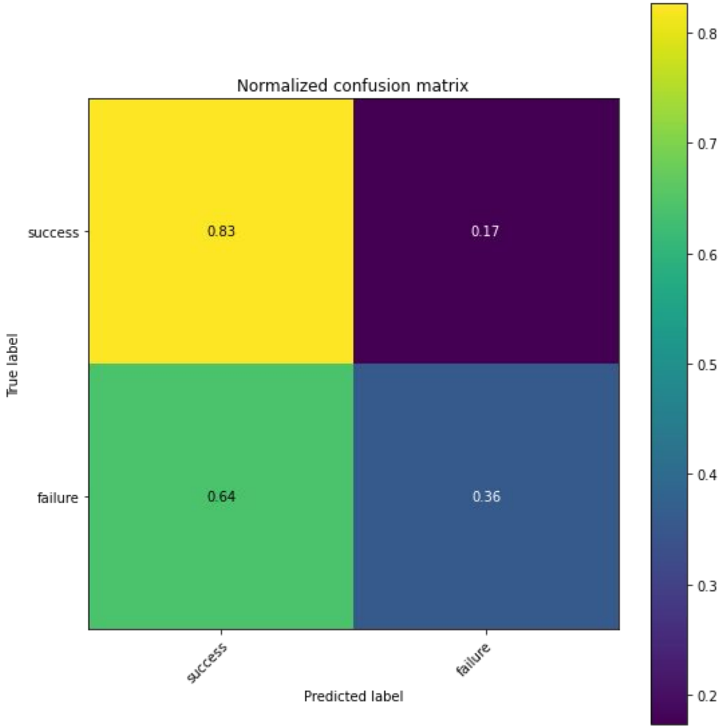
Ensemble Learning

Random Forest

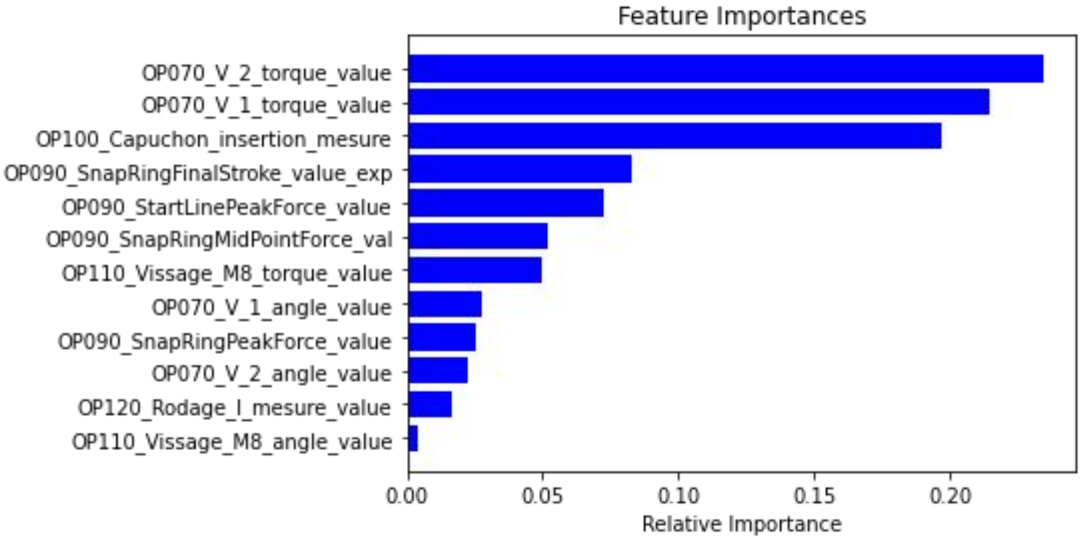
Balanced dataset	yes
Scaled dataset	no

```
smote = SMOTE()

X_train_balanced, y_train_balanced = smote.fit_resample(X_train, y_train)

model = RandomForestClassifier(n_estimators=100,
                      criterion='gini',
                      max_leaf_nodes=16,
                      random_state=5,
                      n_jobs=-1,
                      max_features = 'auto' # sqrt(n_features)
                      )

model.fit(X_train_balanced, y_train_balanced)
```

Better results than LogisticRegression but still not satisfying



Feature importance



- Some features seem much more important than others

Randomized Search

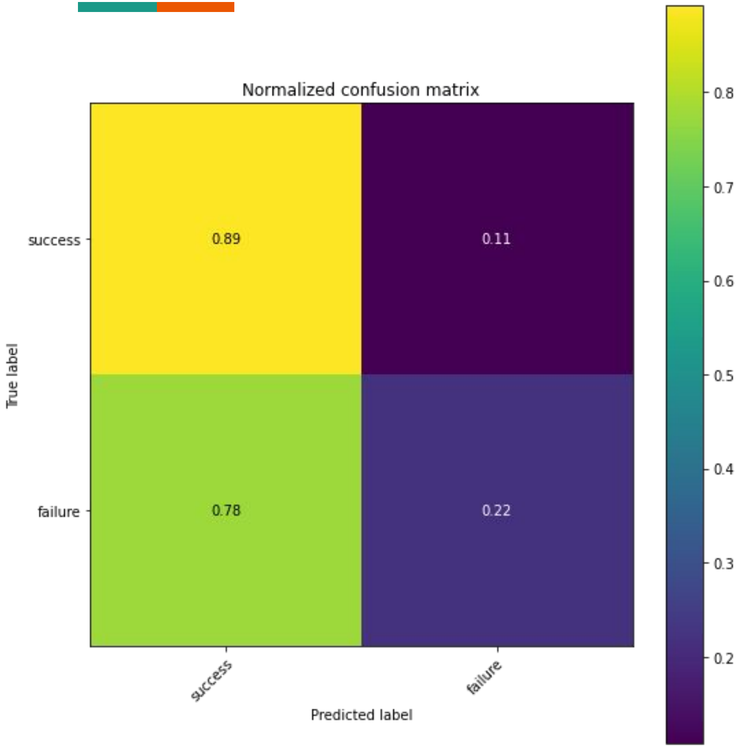
```python
forest = RandomForestClassifier(n_estimators=100, random_state = 4, n_jobs=1,)


search = RandomizedSearchCV(
              scoring = 'balanced_accuracy', # we also tried with 'f1'
              estimator=forest,
              param_distributions=param_grid,
              n_iter=50,
              verbose=2,random_state=42,
              n_jobs=-1,
              cv=5)

model = search_or_load(filename, search, X_train_balanced, y_train_balanced)
```

```python
param_grid = {
    'criterion':['gini', 'entropy'],
    'max_features':list(range(1,13)),

    'max_leaf_nodes':[16, 32, 64, 128, 256, 512, 1024, 2048],
    'min_impurity_decrease' : [0, 0.001, 0.01, 0.1, 0.2],

    'max_depth':[1,10,100,1000,10000,100000],
    'min_weight_fraction_leaf' : [0.1, 0.01, 0.001, 0]

}
```

Goal:

- Trying many configurations automatically
- Get the best hyperparameters

Randomized Search



The recall on failures is lower than before

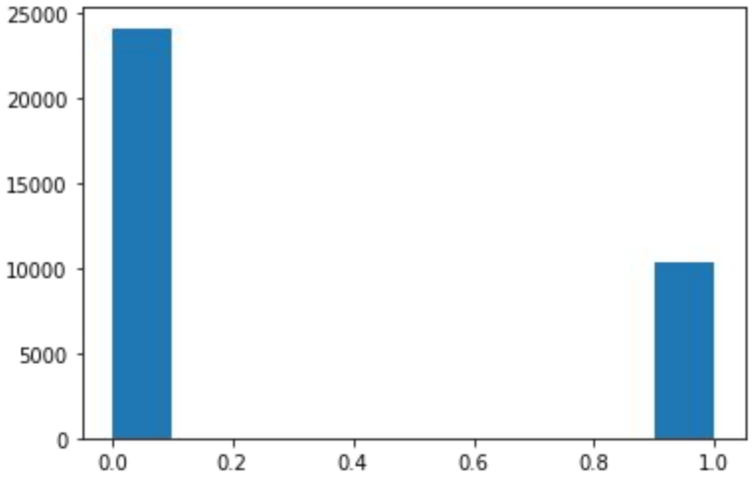
model parameters (found by RandomizedSearchCV):

```
{ 'min_weight_fraction_leaf': 0,
  'min_impurity_decrease': 0,
  'max_leaf_nodes': 128,
  'max_features': 5,
  'max_depth': 10,
  'criterion': 'gini'}
```

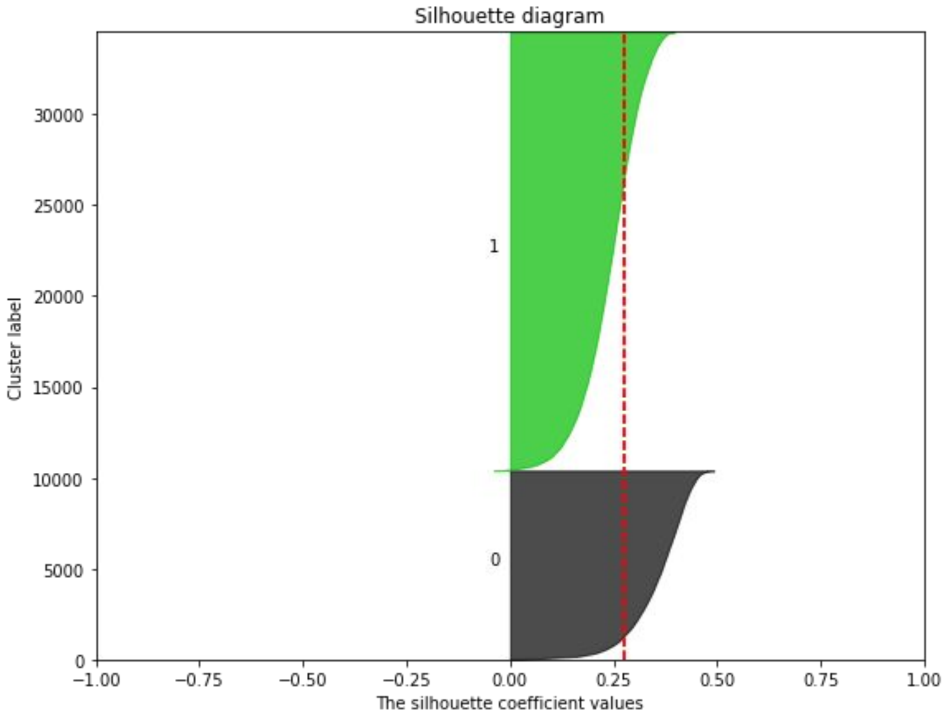
Unsupervised Learning

KMeans Clustering, Isolation Forrest and Autoencoders

K-Means Clustering



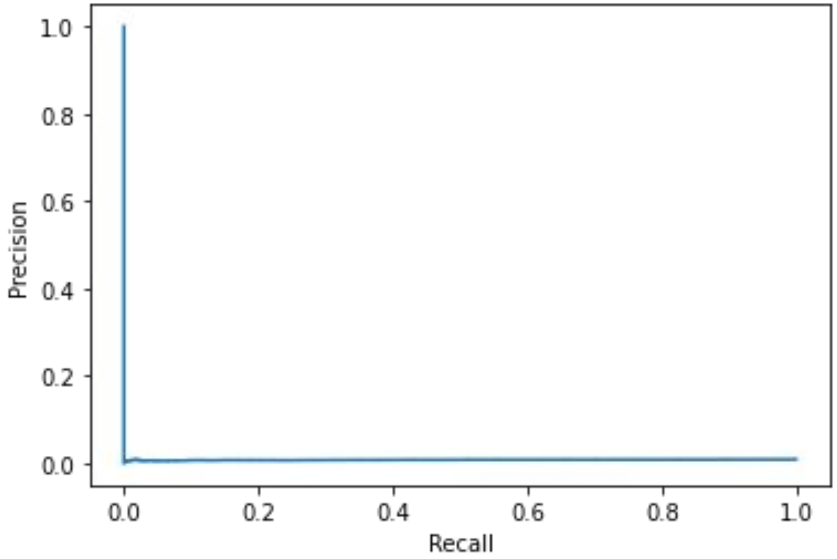
Scaled dataset	yes
Balanced dataset	no

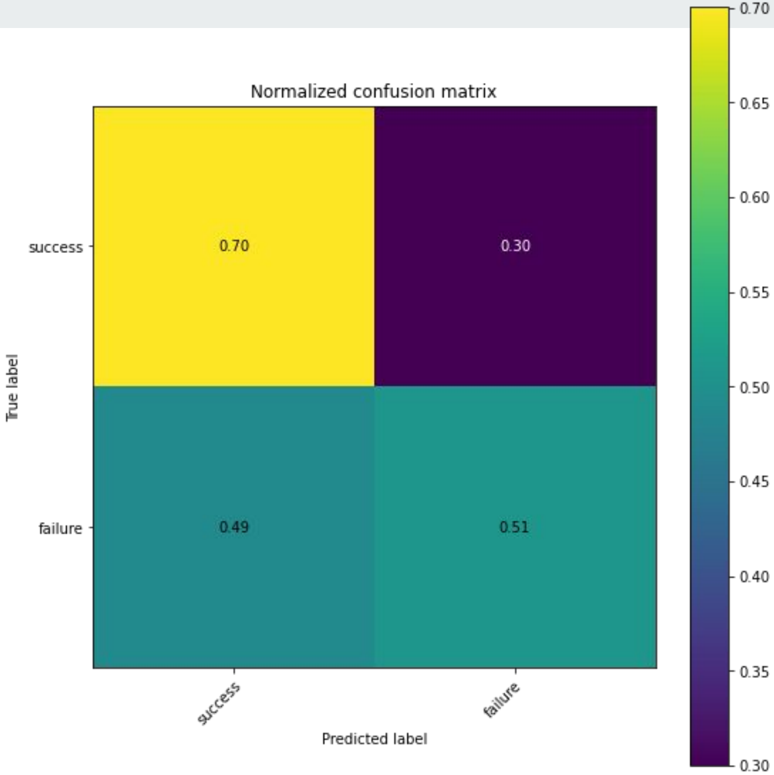


Silhouette analysis for KMeans Clustering with clusters = 2

Avg = 0.2763

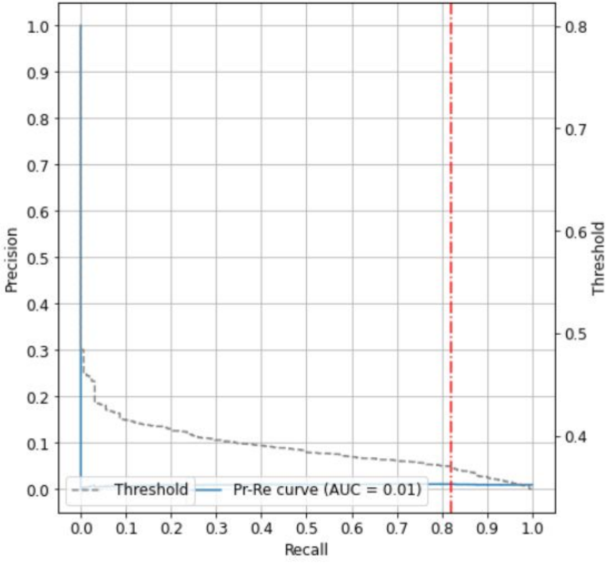
K-Means Clustering

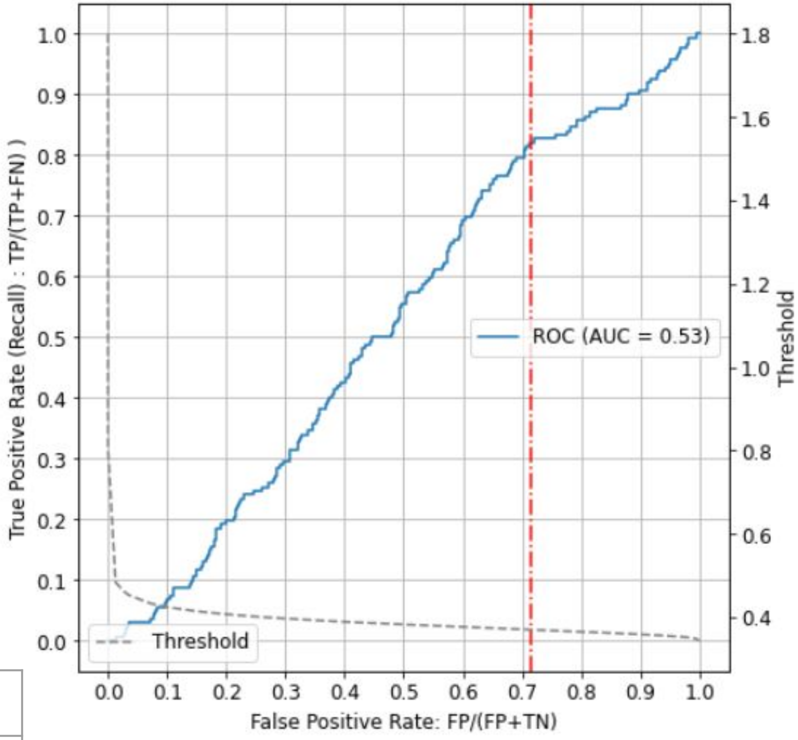




- Scaling with Standard Scaler
- K=2 and repetitions = 130
- All the dataset used to train the model

Isolation Forests





- Trees = 1500
- Threshold = 0.37

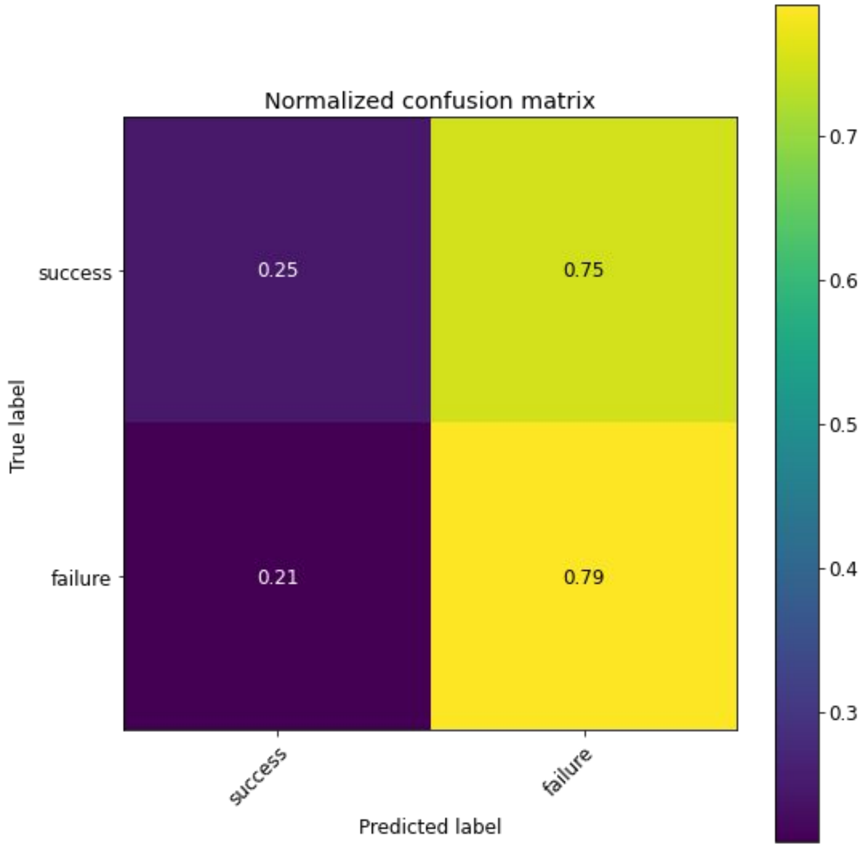
Scaled dataset	no
Balanced dataset	no

Receiver-Operating Characteristic (ROC) Curve

Isolation Forests

Scaled dataset	no
Balanced dataset	no

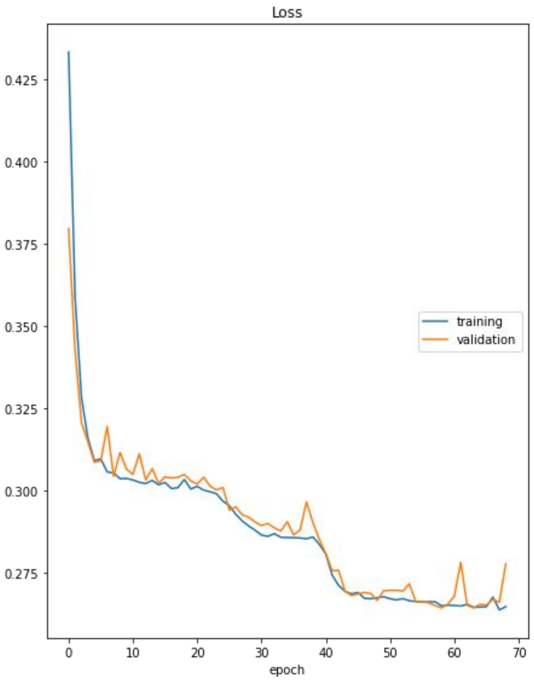
- Trees = 1500
- Threshold = 0.37
- Precision = 0.0087
- Recall = 0.79



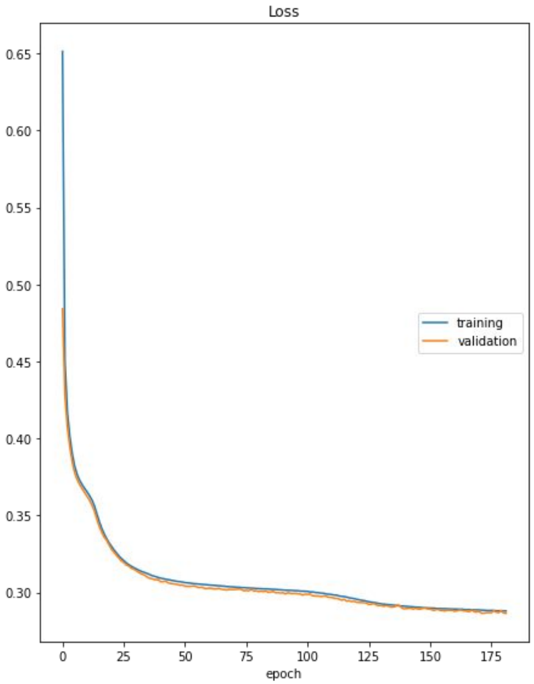
Auto-Encoders

Scaled dataset	yes
Balanced dataset	no

Loss function: MSE

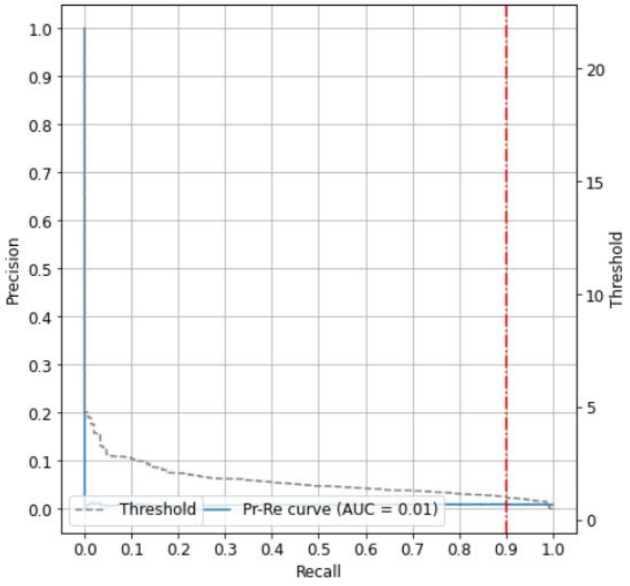


Learning rate = 0.01

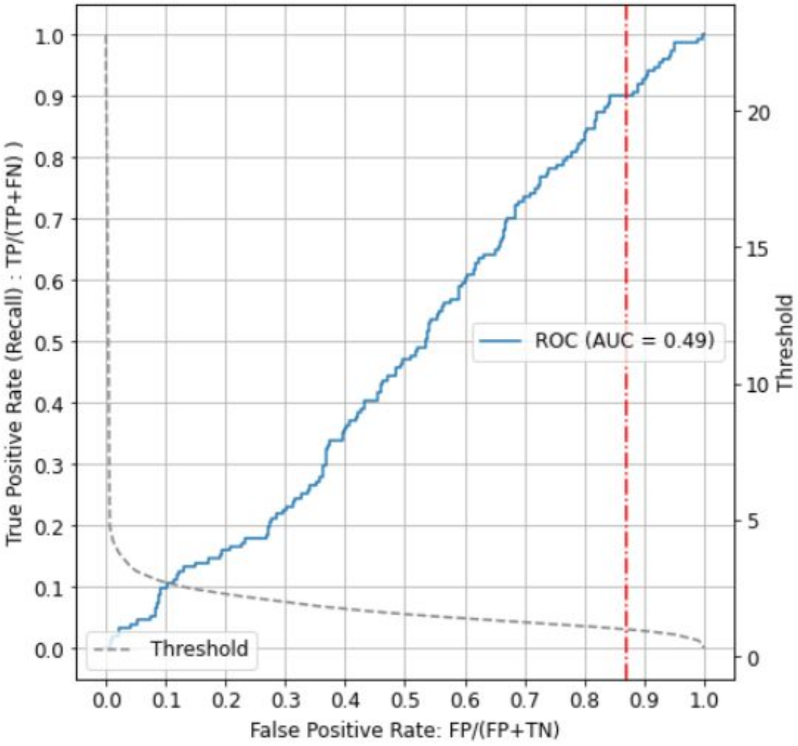


Learning rate = 0.001

Auto-Encoders



Scaled dataset	yes
Balanced dataset	no



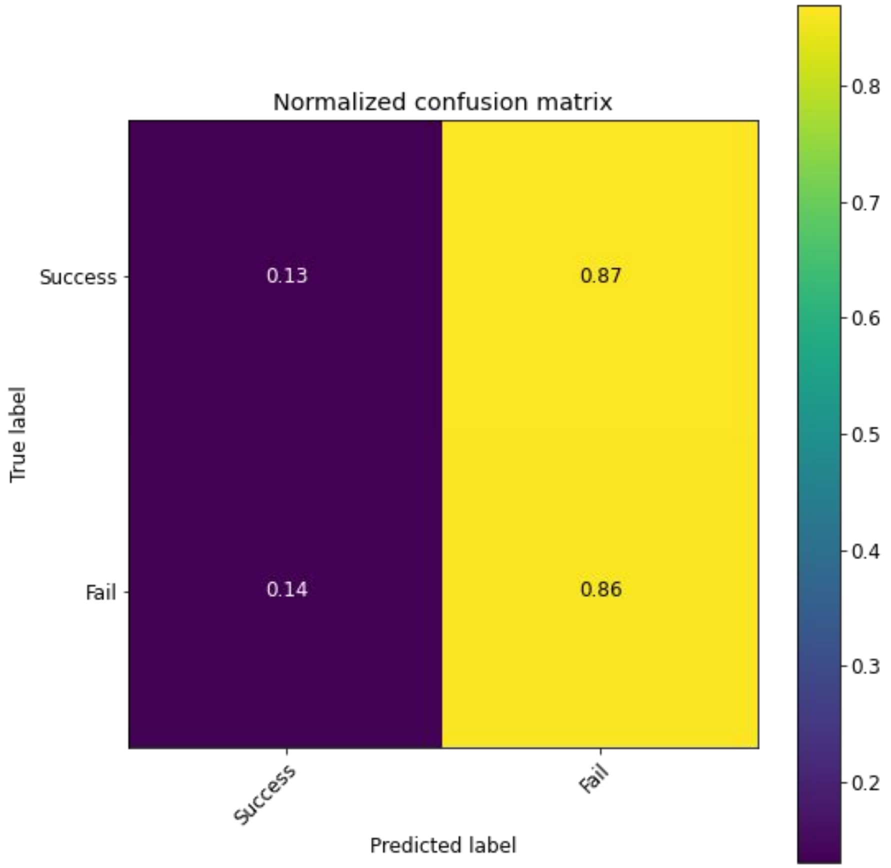
Receiver-Operating Characteristic (ROC) Curve

Threshold = 1

Auto-Encoders

Scaled dataset	yes
Balanced dataset	no

- Loss function: MSE
- Learning rate = 0.001
- Epochs = 182
- Precision = 0.0089
- Recall = 0.86



Conclusion

Comparison Between the Techniques

Machine Learning Technique	Precision (class: failure)	Recall (class: failure)
Logistic Regression	0.01	0.52
Neural Networks	0.01	0.30
Ensemble Learning	0.02	0.33
KMeans	0.01	0.51
Isolation Forrest	0.0087	0.79
Auto-Encoders	0.0089	0.86

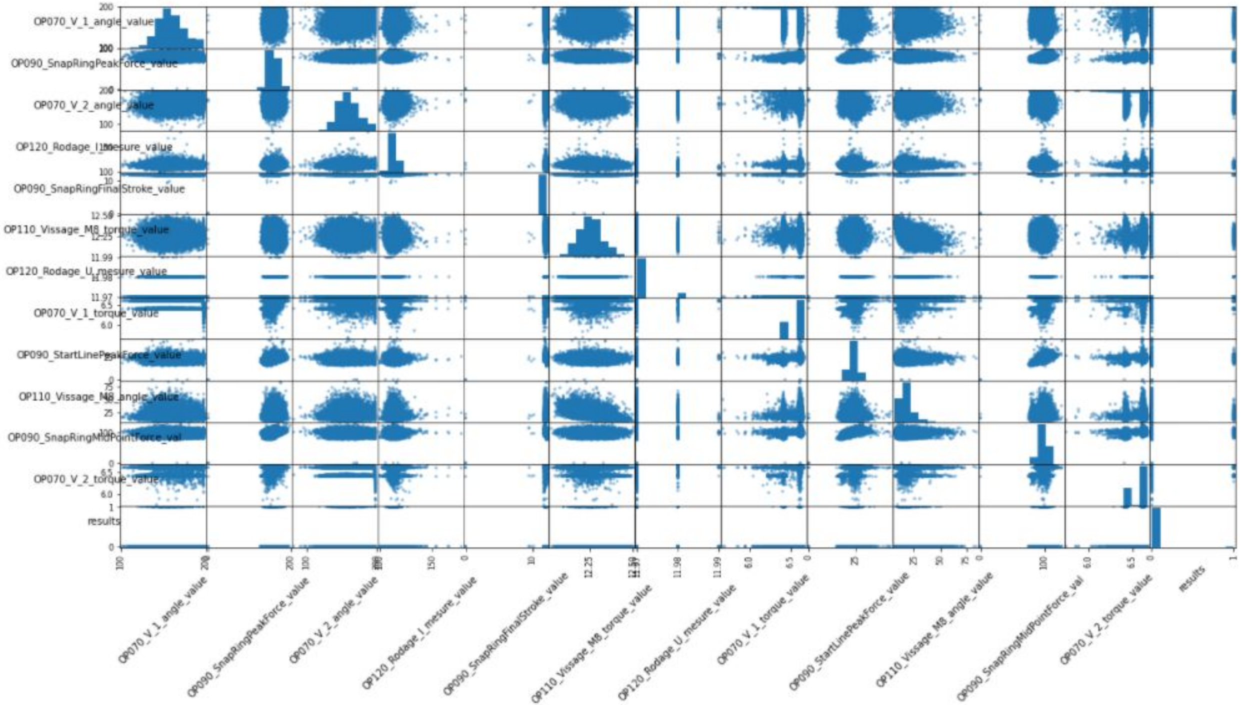
THANKS!

Any questions?

```python
sm = scatter_matrix(df_cleaned, figsize=(20,10))
visualization.rotate_labels(sm)
plt.show()
```

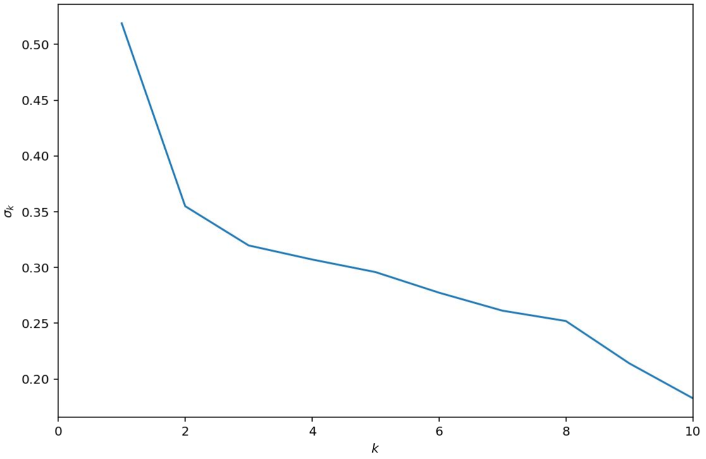
Appendix

[1].Scatter matrix obtained with the dataset after Pre-Processing



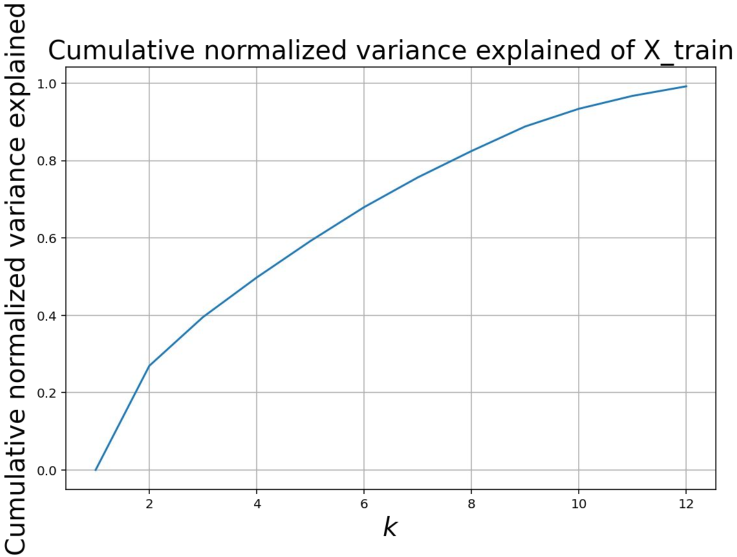
Appendix

Dimensionality Reduction



10 largest singular values, normalized

- there is no "knee"
- hard to choose the right number of components



- all the variables are important
- maybe, try with the 10 most important

Logistic Regression