07.04.2022

Master
Project Management
and Data Science

htw

SECOM ANALYTICS - PART 1

TEAM 3

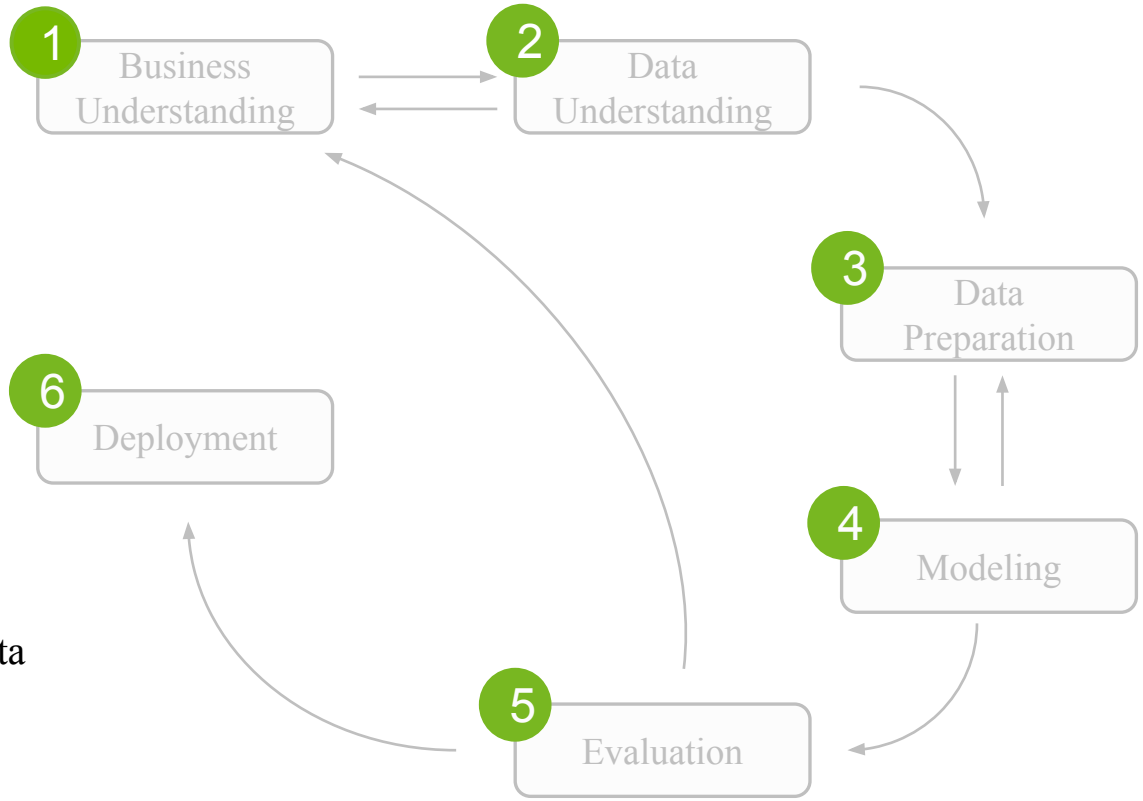
Gupta, Himansha

Pomay Polat, Ekin

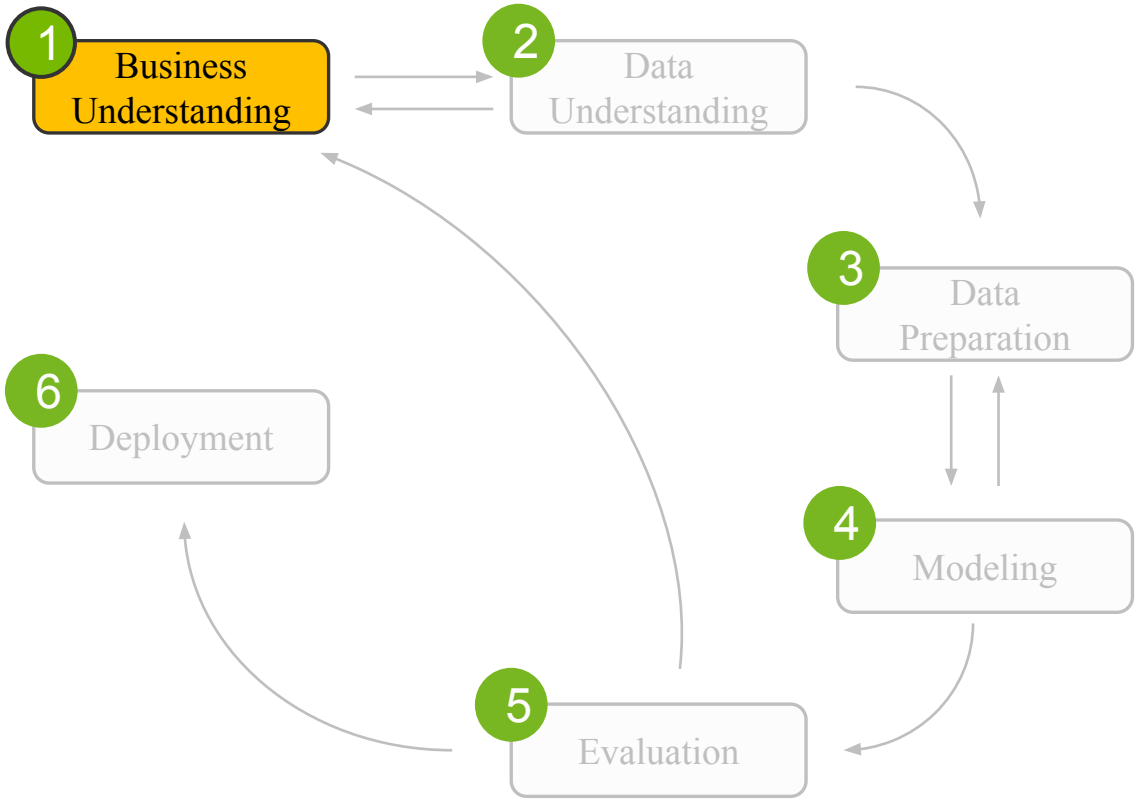
Dsouza, Rashmi Carol

Pham, Quynh Dinh Hai

CONTENT



01
Business Understanding



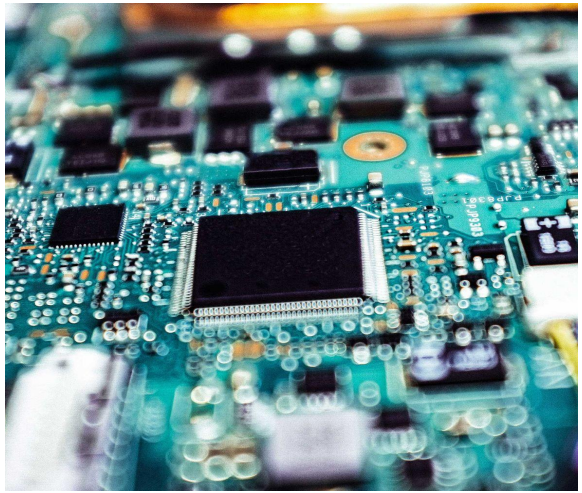
Semiconductor Manufacturing

Semiconductor manufacturing involves

- highly complex and lengthy processes with 300–500 steps
- large number of interrelated variables

The industry constantly needs to find ways to

- monitor processes effectively and efficiently
- extract useful information



Semiconductor Manufacturing

Process monitoring and fault detection is critical to meet the increasing demand for high-quality products and reliable processes

- Many features are monitored simultaneously
- Processes are monitored closely to quickly detect abnormal behaviors & assign causes in order to reduce abnormal yield loss

Problems Semiconductor Manufacturing Faces

01

Size and dimension of integrated circuits (IC) keeps shrinking to accommodate nano-generation

- ➔ increase difficulty in controlling quality

02

Poor data quality

- ➔ process measurements are often insufficient, incomplete or unreliable

03

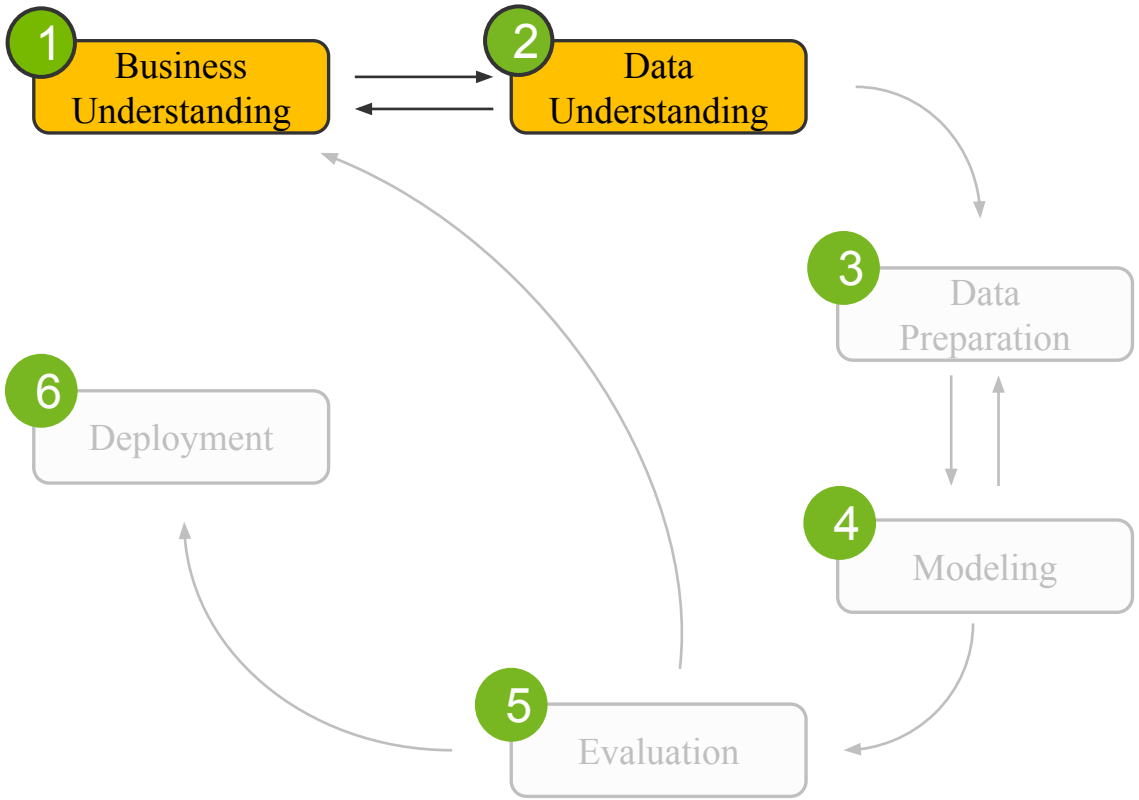
Large data quantity

- ➔ hard to extract useful information
- ➔ hard to monitor and maintain with the increasing complexity and manufacturing processes
- ➔ difficult to detect correlation between each feature
- ➔ increase in errors

02

Data Understanding

- Data Importing
- Data Description
- Data Exploration
- Data Quality



Data Loading & Data Information

```
[ ] data_url="https://archive.ics.uci.edu/ml/machine-learning-databases/secom/secom.data"
```

```
[ ] label_url="https://archive.ics.uci.edu/ml/machine-learning-databases/secom/secom_labels.data"
```

```
[ ] secom_data = pd.read_csv(data_url,sep=' ',header=None)
```

```
[ ] secom_data.head()
```

	0	1	2	3	4	5	6	7	8	9	...

```
[ ] secom_labels = pd.read_csv(label_url, sep = " ",header=None)
```

```
[ ] secom_labels.head()
```

	0	1
0	-1	19/07/2008 11:55:00
1	-1	19/07/2008 12:32:00

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
import seaborn as sns
import missingno as msno

from sklearn.model_selection import train_test_split
from sklearn.impute import KNNImputer
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import f1_score, matthews_corrcoef
from yellowbrick.classifier import ConfusionMatrix, ClassPredictionError
from sklearn.preprocessing import Normalizer
from sklearn.feature_selection import VarianceThreshold

import plotly as py
import plotly.express as px
```

- The SECOM dataset consists of 2 files with:
  - 1 data file containing **1567 observations** taken from a wafer fabrication production line, each observation with **590 sensor measurements**
  - 1 label file containing **timestamp** & **passed/failed classification**
- Large number of observations with many different sensor measurements together with other problems such as missing values, imbalance between passed & failed test results, etc, make it a challenge to analyze this dataset.

Data Imbalance

KEY FINDINGS:

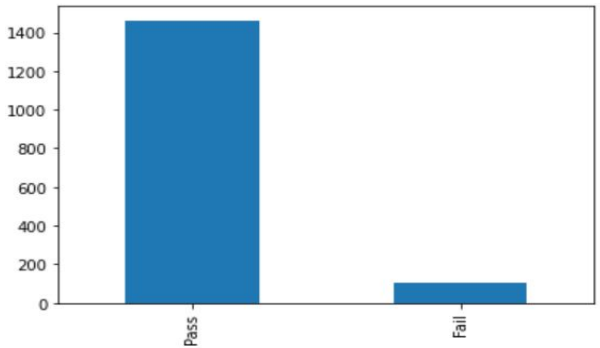
There is an imbalance in dataset due to the product passing or failing the test

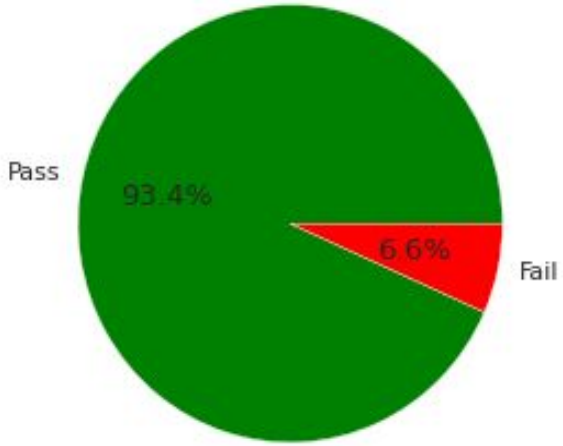
Among 1567 observations, there are

- **104 fail cases** (labeled as 1)
- **1463 pass cases** (labeled as -1)
- **1:14** ratio between the number of fail & pass result

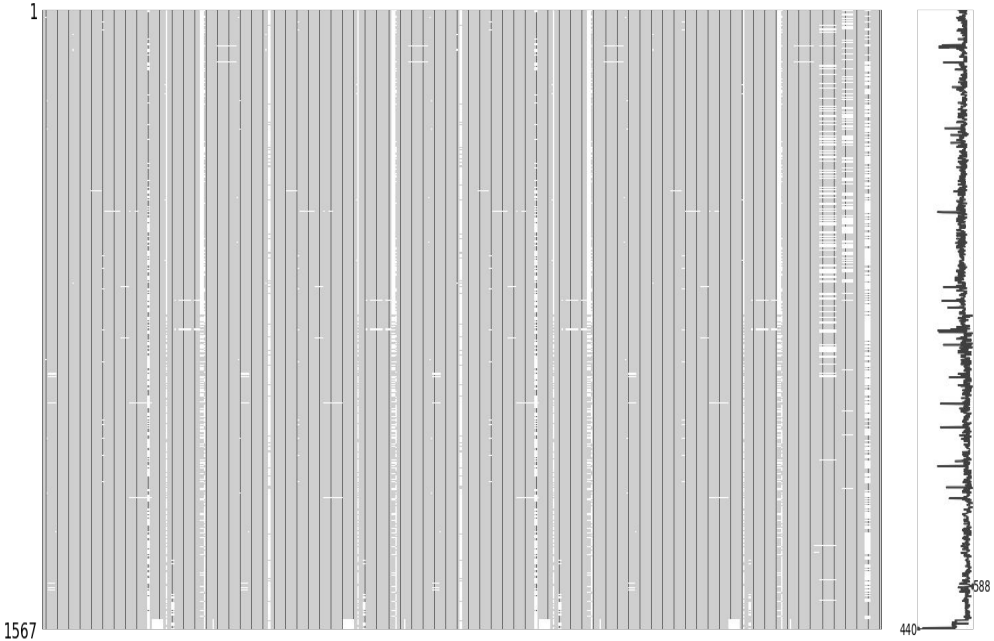
```
[ ]  data.Classification.value_counts()

     Pass    1463
     Fail     104
     Name: Classification, dtype: int64
```

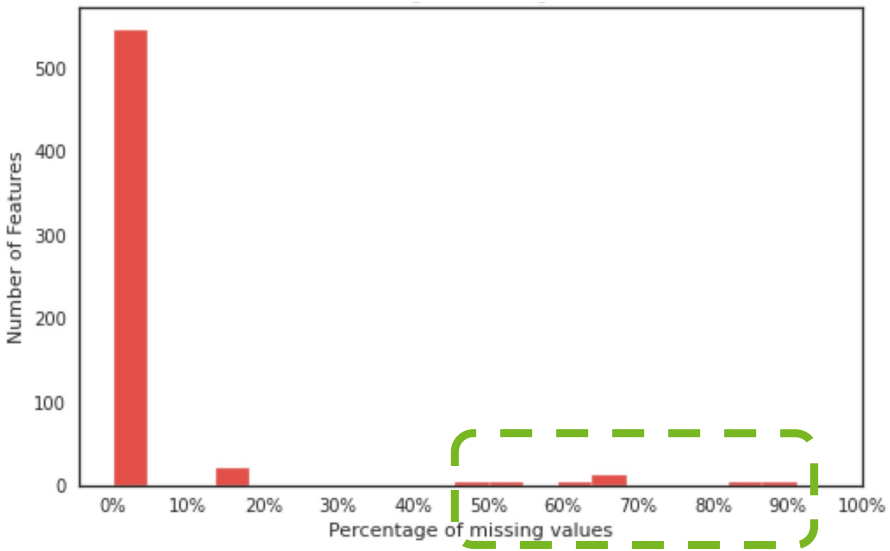
```
fig1 = plt.figure("Figure 1")
data.Classification.value_counts().plot(kind = 'bar')
plt.show()
```





Missing Value Analysis





Missing Value Analysis

```
missing55 = missing[missing["percent_missing"] >= 55]
missing55.describe()
```

	percent_missing
count	24.000000

KEY FINDINGS

- **41951** missing values
- **24** features with more than **55%** of values missing

RESULT

- Features with large amount of missing values might be problematic

(POSSIBLE) SOLUTION

- Remove features with more than 55% missing values

Duplicate Value Analysis

```
## Storing features with Duplicated Values
dupval = []
for col in secom_2.columns:
    if(secom_2[col].unique().size == 1):
        if(len(dupval) == 0):
            dupval =  secom_2[col]
        else:
            dupval = pd.concat([dupval, secom_2[col]], axis=1)
dupval.pop("Classification")
dupval.head()
```

```
const_cols = nunique[nunique == 1].index
const_cols.shape
```

```
(116,)
```

KEY FINDINGS

- **No observation or feature** is duplicated
- **116 features** have only **1 unique value (low variance)**

RESULT

- Features with only 1 unique value do not contribute any meaningful value to models that will be built.

(POSSIBLE) SOLUTION

- Remove features with 1 unique value

Master
Project Management
and Data Science

Exploring Time

```
from datetime import datetime
data['year'] = pd.DatetimeIndex(data['Timestamp']).year
data['month'] = pd.DatetimeIndex(data['Timestamp']).month
data['date'] = pd.DatetimeIndex(data['Timestamp']).day
data['week_day'] = pd.DatetimeIndex(data['Timestamp']).weekday
data['start_time'] = pd.DatetimeIndex(data['Timestamp']).time
data['hour'] = pd.DatetimeIndex(data['Timestamp']).hour
data['min'] = pd.DatetimeIndex(data['Timestamp']).minute
data=data.drop('Timestamp',axis=1)
```

```
data.year.unique()
```

```
array([2008])
```

KEY FINDINGS

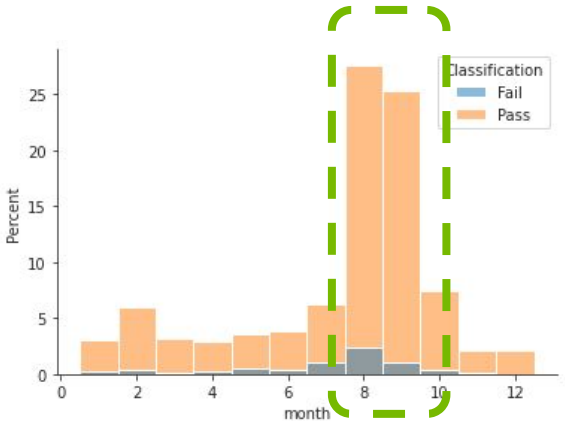
- Data belongs to year **2008**
- The busiest months are **August** & **September**
- **Friday** as a slightly higher value than most days
- The first **10 days** of every month is the busiest
- The ratio of pass & fail result **remain the same** across different time (month, weekday or date)

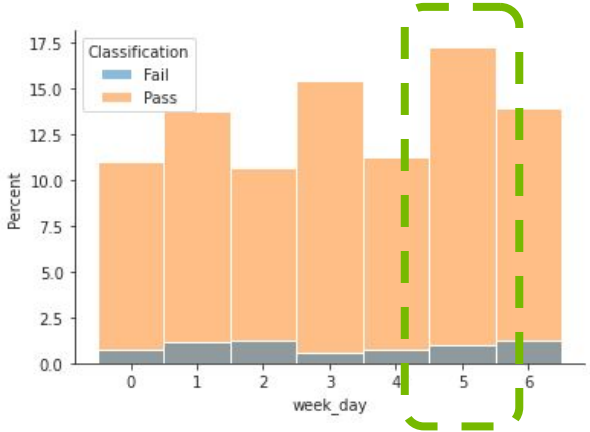
RESULT

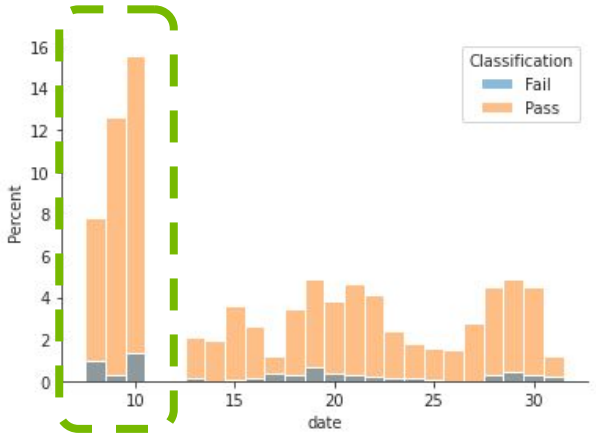
- This helps to understand patterns in terms of time, however there is **no further insight** to pass & fail classification.

(POSSIBLE) SOLUTION

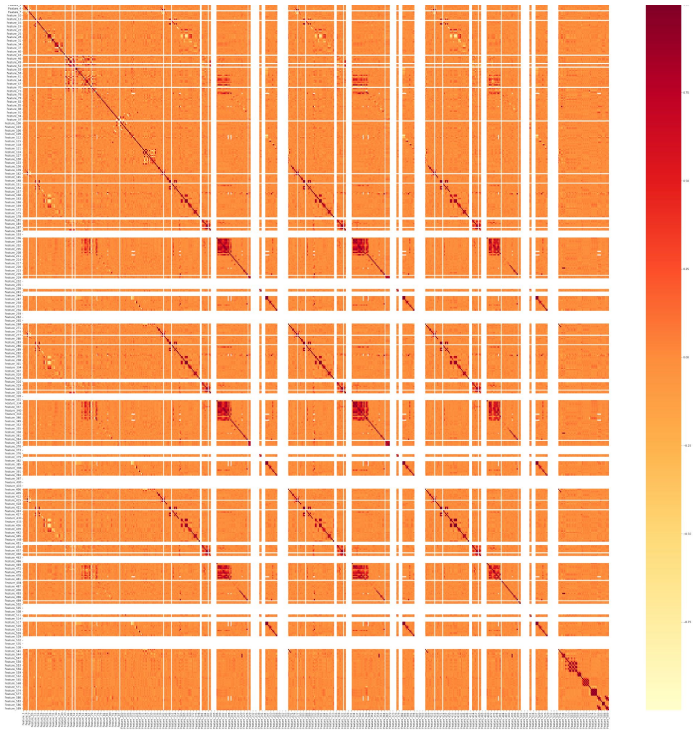
- Remove Timestamp in the next phase





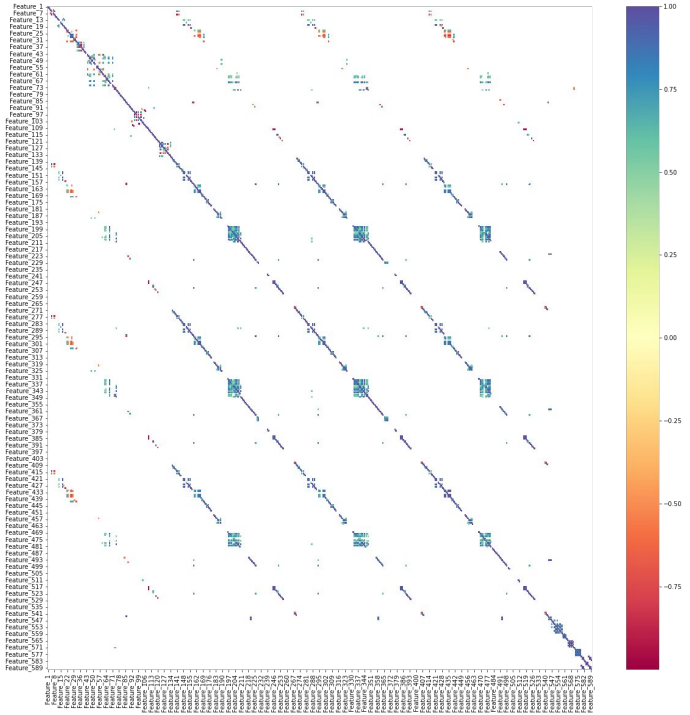


Correlation Heatmap

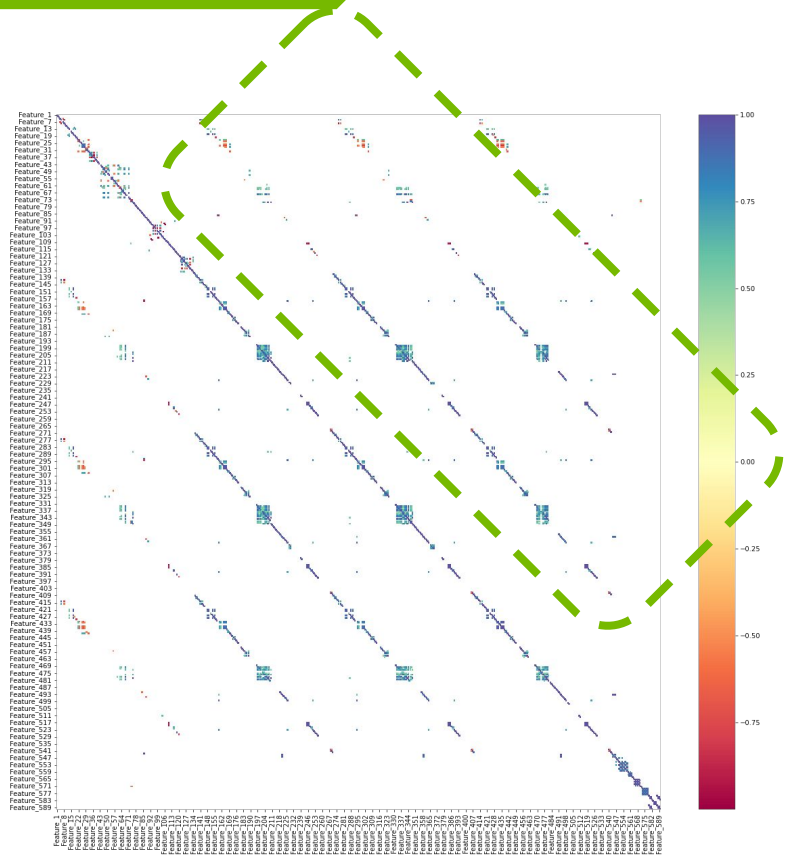


Heatmap

Only pairs with correlation factor >0.5 are colored



In order to understand patterns clearly…



KEY FINDINGS:

- **Several diagonal** pattern beyond the main diagonal that are **highly correlated.**
- **220** features can be removed if one feature of every correlated pair is dropped at a threshold of 0.90

(POSSIBLE) SOLUTIONS:

- **One feature** of each correlated pair can be removed
- Threshold => can be determined as 0.90
  => in order to clean highly correlated features.

Examples of Highly Correlated Features

Feature_28	Feature_26	0.98
Feature_37	Feature_35	1.0
Feature_51	Feature_47	0.9
Feature_55	Feature_54	0.94
Feature_61	Feature_44	0.9
Feature_71	Feature_67	0.9
Feature_97	Feature_95	0.96
Feature_102	Feature_99	0.91
Feature_105	Feature_100	0.99
Feature_106	Feature_93	0.99
Feature_107	Feature_94	0.99

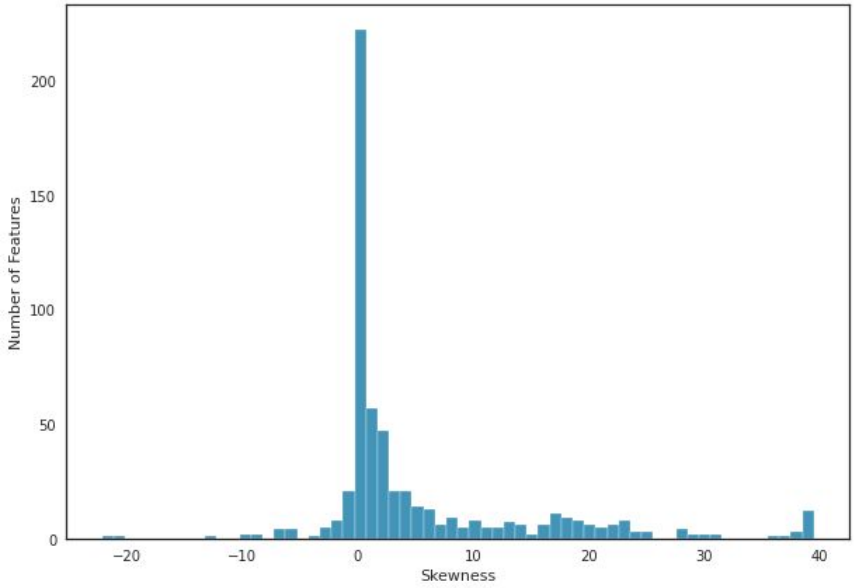
Exploring Skewness

KEY FINDINGS

- 268 features fall in the normal range
- 291 features are positively skewed
- 31 features are negatively skewed

RESULT

- Skewness indicates the likelihood of outliers in the dataset



Top 10 Skewed Features

	Skewness
Feature_391	39.562779
Feature_253	39.560364
Feature_288	39.521337
Feature_153	39.519611
Feature_210	39.509493
Feature_75	39.509493
Feature_479	39.509493
Feature_343	39.509493
Feature_348	39.509493
Feature_207	39.509493

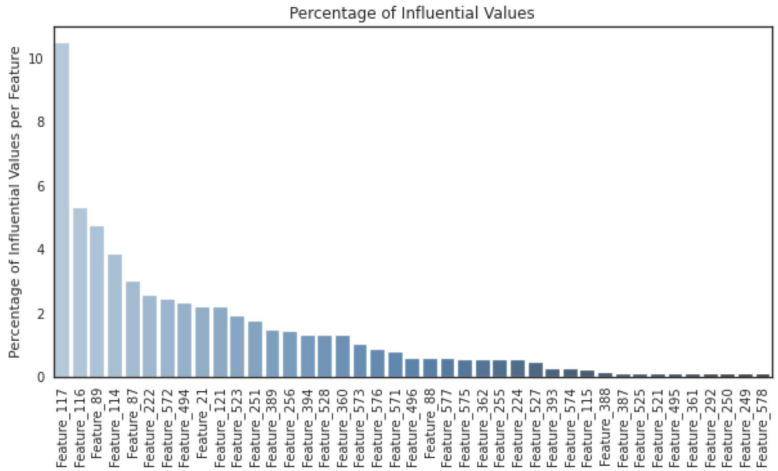
Outlier Detection (Z Score)

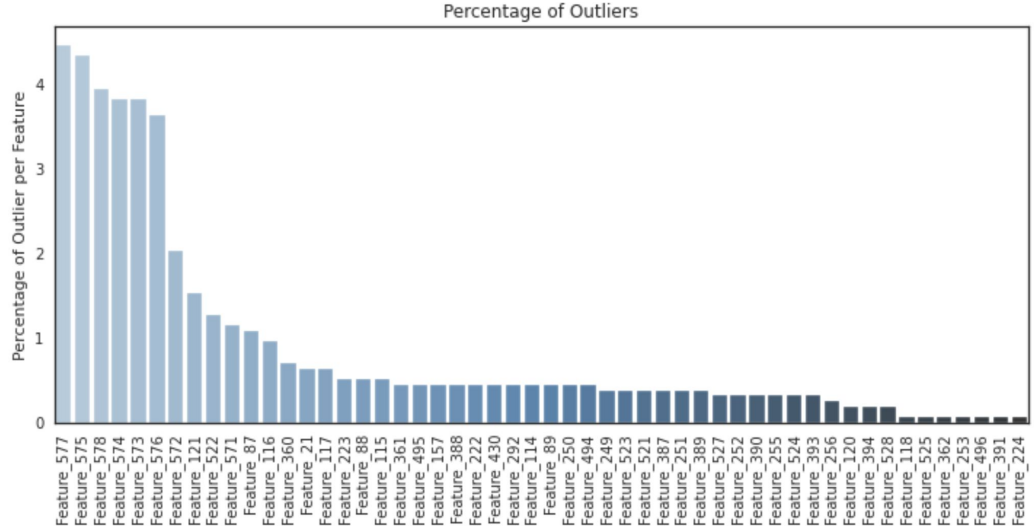
KEY FINDINGS

- **52 features** have more than one outlier
- **41 features** have more than one influential value

RESULT

- Outliers skew the normality of the data which may impact our models and results

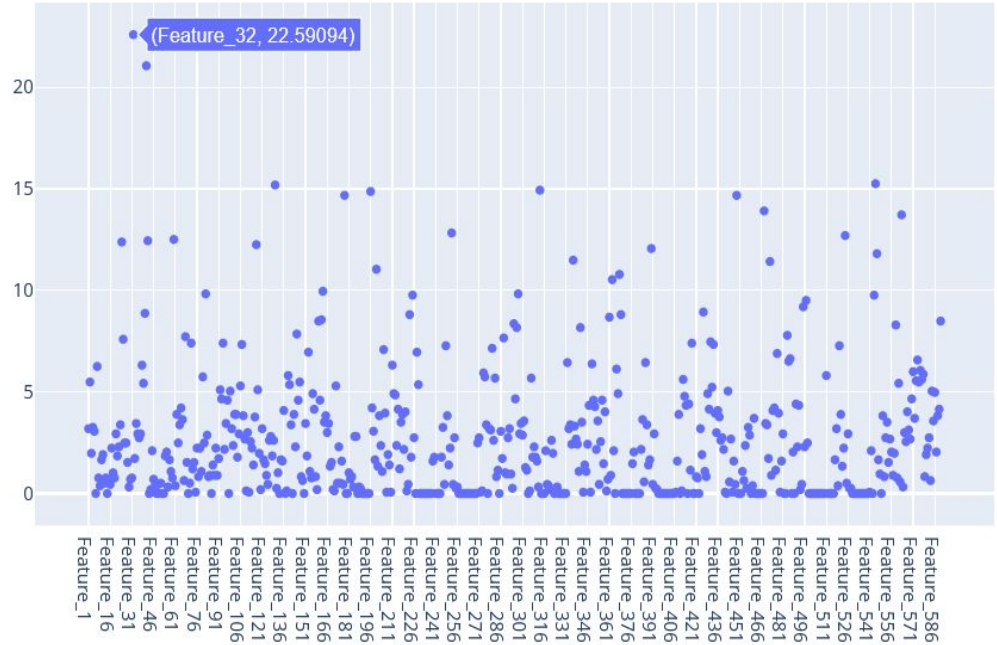




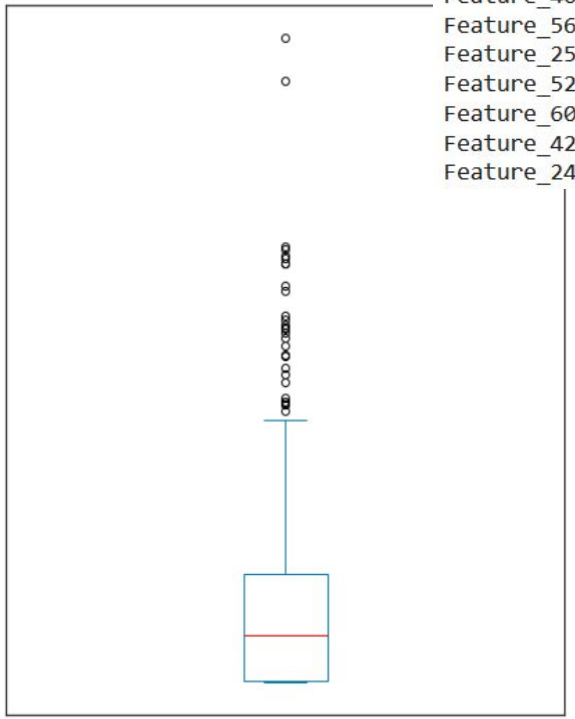
Outlier Detection (IQR)

(POSSIBLE) SOLUTIONS

- Flag outliers
- Remove outliers
- Replace outliers with suitable values
- Quantile transformation



Feature_32	22.590938
Feature_41	21.059349
Feature_545	15.252074
Feature_130	15.188258
Feature_313	14.932993
Feature_196	14.869177
Feature_178	14.677728
Feature_449	14.677728
Feature_468	13.911934
Feature_563	13.720485
Feature_252	12.827058
Feature_524	12.699426
Feature_60	12.507977
Feature_42	12.444161
Feature_24	12.380345

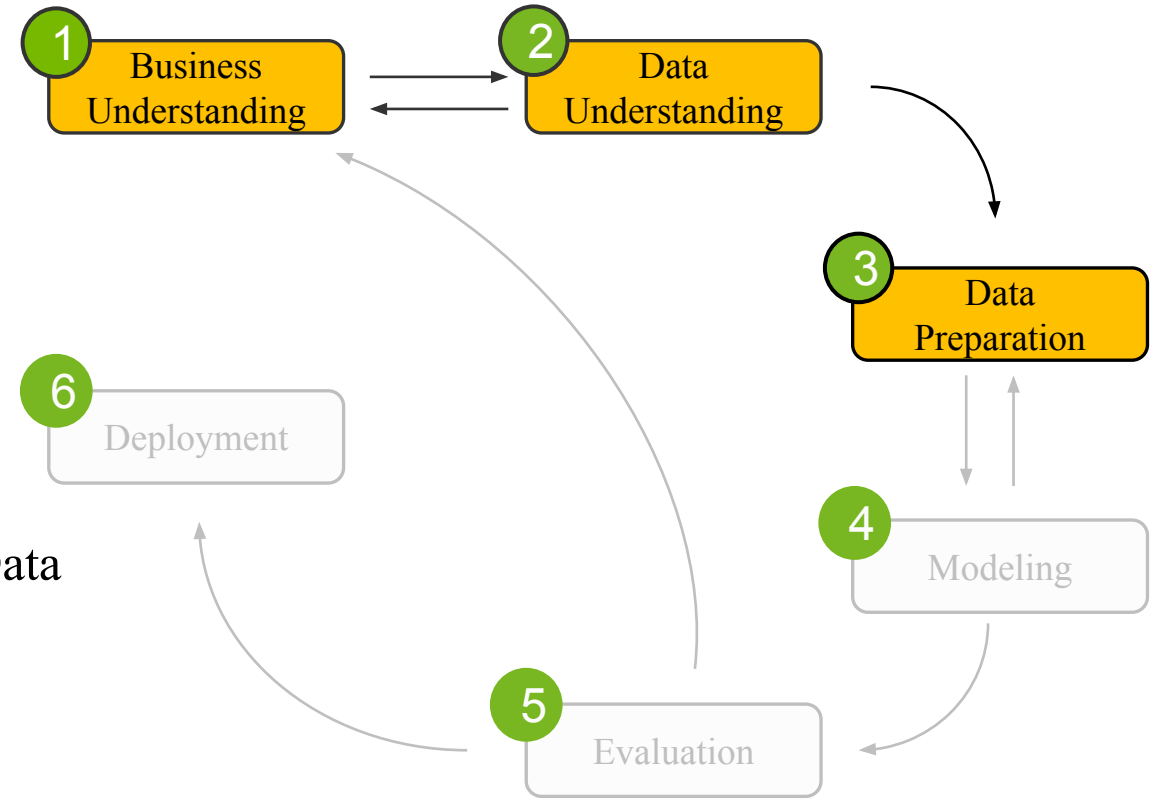


Data Quality

No.	Problems with the dataset	Possible Solutions
1	Imbalance data between pass & fail test results	Treatment for data imbalance in next phase (SMOTE, TOMEK, Random Undersampling, Random Oversampling, etc)
2	Missing values	Define threshold. Flag missing values. Treatment for missing values in next phase (remove missing values or replace missing values with suitable values, etc)
3	Features with only 1 unique value	Remove features
4	Timestamp does not yield meaningful insight to build model	Remove “Timestamp” feature
5	Many highly correlated features	Find threshold for correlation coefficient. One feature of each correlated pair can be removed.
6	Outliers	Flag outliers. Treatment for outliers in next phase (remove outliers, replace with 3s, Quantile Transformer, etc)

03
Data Preparation

- Merge Data
- Separate Training & Test Data
- Dimensionality Reduction



Merging Data

Adding Column Labels

```
secom_data.columns = ["Feature_"+str(column+1) for column in range(len(secom_data.columns))]
secom_data.head()
```

	Feature_1	Feature_2	Feature_3	Feature_4	Feature_5	Feature_6	Feature_7	Feature_8	Feature_9
0	3030.93	2564.00	2187.7333	1411.1265	1.3602	100.0	97.6133	0.1242	1.5005

Reclassifying Data Types

```
secom_labels['Timestamp'] = pd.to_datetime(secom_labels['Timestamp'],errors='raise')
secom_labels['Classification'] = secom_labels["Classification"].astype("category")
```

```
secom_labels.columns = ["Classification","Timestamp"]
secom_labels.head()
```

	Classification	Timestamp
0	-1	2008-07-19 11:55:00

Merged Dataset

```
data= pd.concat([secom_labels,secom_data],axis=1)
data.head()
```

	Classification	Timestamp	Feature_1	Feature_2	Feature_3	Feature_4	Feature_5
0	-1	2008-07-19 11:55:00	3030.93	2564.00	2187.7333	1411.1265	1.3602

Splitting Data into Test and Train

```
df_target = data[['Classification']]
df_data = data.drop(['Classification'], axis=1)
```

```
X_train, X_test, y_train, y_test = train_test_split(df_data, df_target, test_size=0.2, random_state=42, stratify=df_target)
```

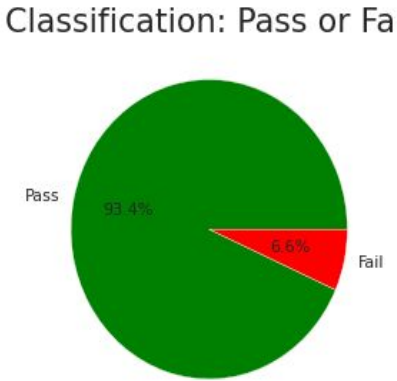
```
X_train = pd.DataFrame(X_train, columns=df_data.columns)
X_test = pd.DataFrame(X_test, columns=df_data.columns)
y_train = pd.DataFrame(y_train, columns=df_target.columns)
y_test = pd.DataFrame(y_test, columns=df_target.columns)
```

```
y_train = y_train.replace(to_replace=[-1, 1], value=[1, 0])
y_test = y_test.replace(to_replace=[-1, 1], value=[1, 0])
```

Splitting Data into Test and Train

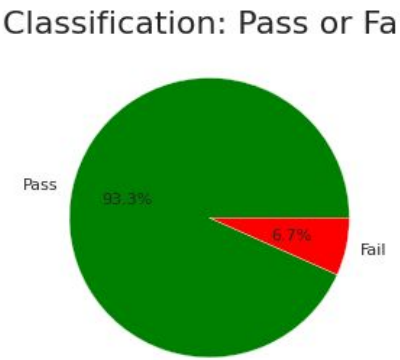
Train Dataset

```
figure(figsize=(6, 4), dpi=80)
pie_data = y_train['Classification'].value_counts()
plt.pie(pie_data, labels=['Pass', 'Fail'], colors=['green', 'red'], autopct="%.1f%%")
plt.title('Classification: Pass or Fail', fontsize=20)
plt.show()
```



Test Dataset

```
figure(figsize=(6, 4), dpi=80)
pie_data = y_test['Classification'].value_counts()
plt.pie(pie_data, labels=['Pass', 'Fail'], colors=['green', 'red'], autopct="%.1f%%")
plt.title('Classification: Pass or Fail', fontsize=20)
plt.show()
```



Dimensionality Reduction

Missing Values

Threshold:
Features with more than 50% missing value

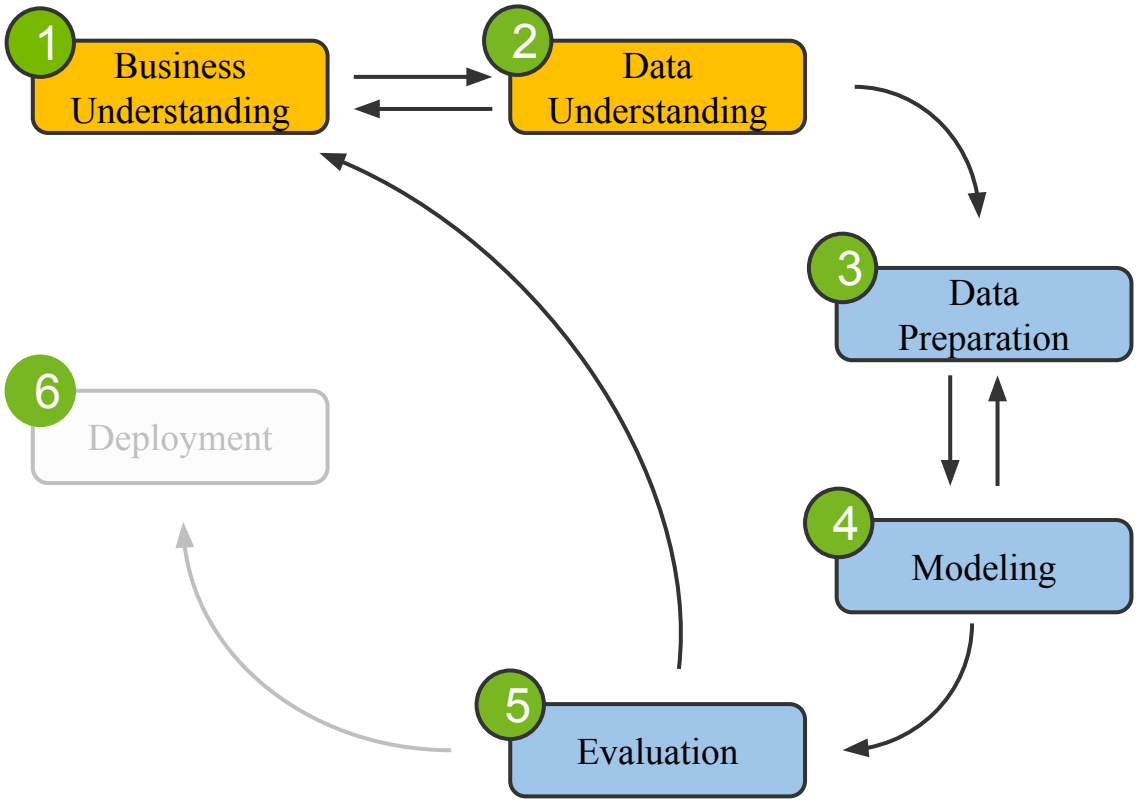
```
def percent(dataframe, threshold):
    columns = dataframe.columns[(dataframe.isna().sum()/dataframe.shape[1])>threshold]
    return columns.tolist()

na_columns = percent(X_train, 0.5)
X_train_na = X_train.drop(na_columns, axis=1)
X_test_na = X_test.drop(na_columns, axis=1)
n_features1 = X_train_na.shape[1]
print(f'Removed = {len(na_columns)} , Left = {n_features1}')
```

```
Removed = 32 , Left = 558
```

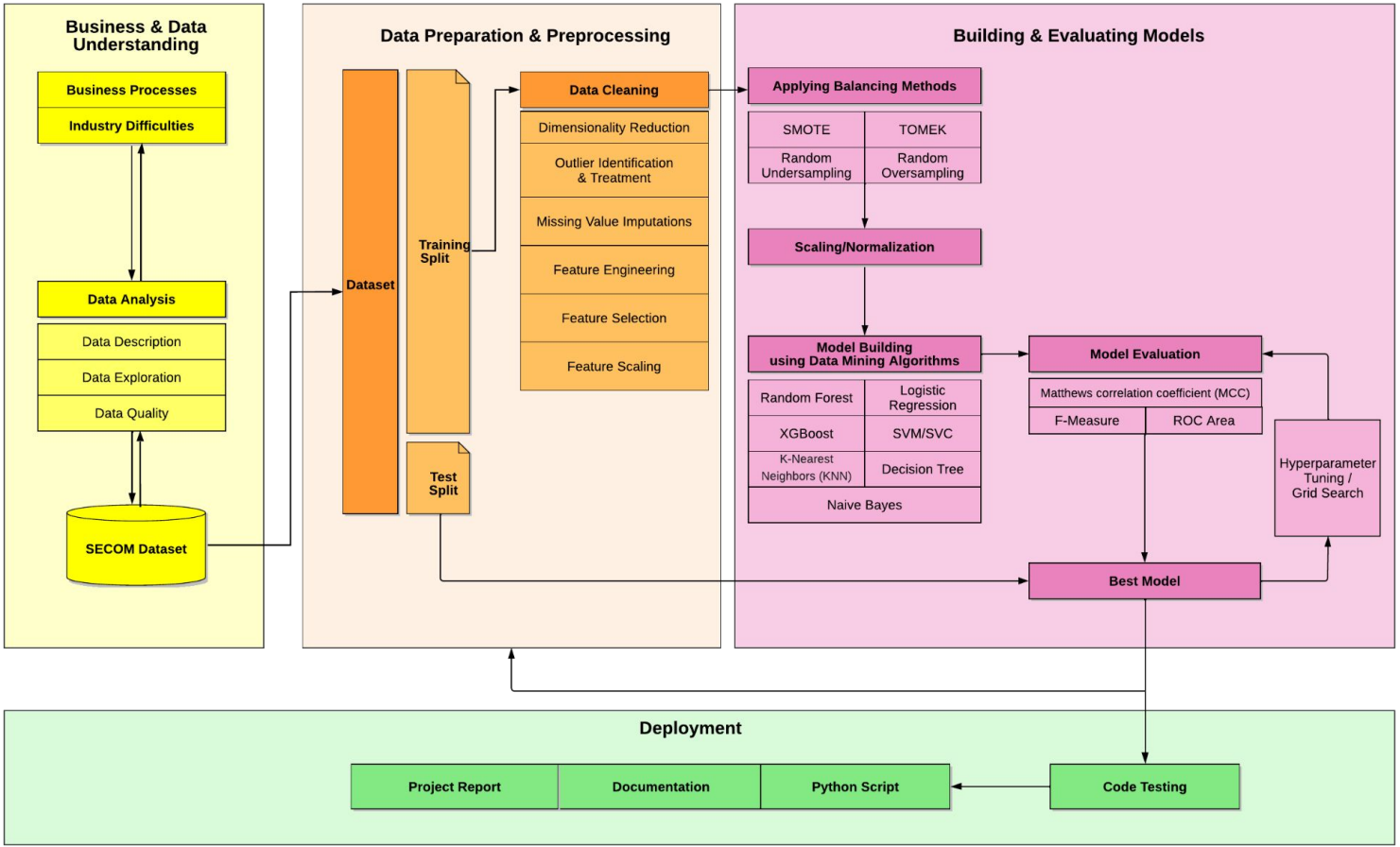
04

Next Steps

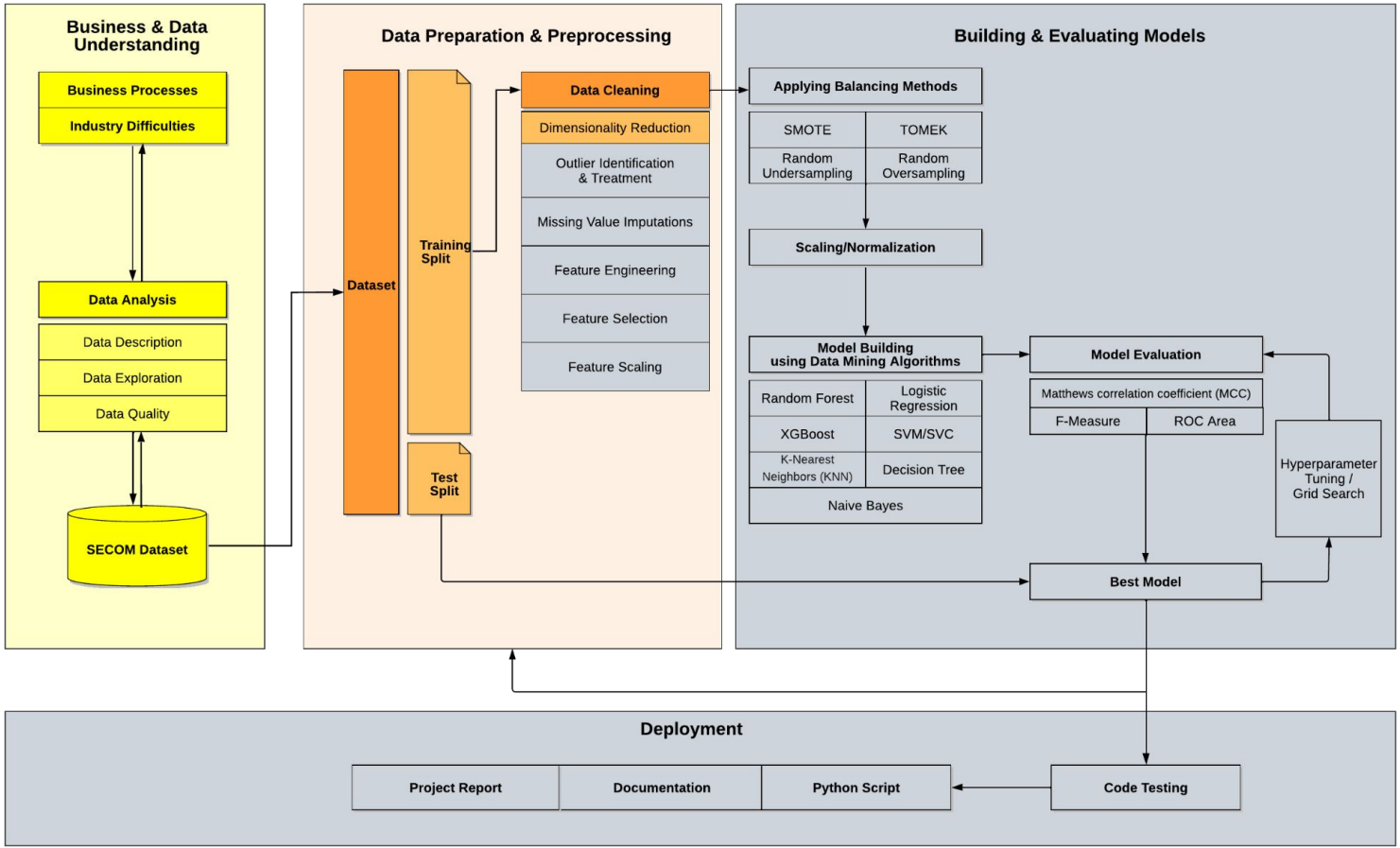
- Data Preparation
- Modeling
- Evaluation



Complete Workflow



Upcoming…


Business & Data Understanding
Business Processes
Industry Difficulties
Data Analysis
Data Description
Data Exploration
Data Quality
SECOM Dataset
Data Preparation & Preprocessing
Dataset
Training Split
Test Split
Data Cleaning
Dimensionality Reduction
Outlier Identification & Treatment
Missing Value Imputations
Feature Engineering
Feature Selection
Feature Scaling
Building & Evaluating Models
Applying Balancing Methods
SMOTE
TOMEK
Random Undersampling
Random Oversampling
Scaling/Normalization
Model Building using Data Mining Algorithms
Random Forest
Logistic Regression
XGBoost
SVM/SVC
K-Nearest Neighbors (KNN)
Decision Tree
Naive Bayes
Model Evaluation
Matthews correlation coefficient (MCC)
F-Measure
ROC Area
Hyperparameter Tuning / Grid Search
Best Model
Deployment
Project Report
Documentation
Python Script
Code Testing

Master
Project Management
and Data Science

Vielen Dank!

Gupta, Himansha
Himansha.Gupta@student.htw-berlin.de

Pomay Polat, Ekin
Ekin.PomayPolat@student.htw-berlin.de

Dsouza, Rashmi Carol
Rashmi.Dsouza@Student.HTW-Berlin.de

Pham, Quynh Dinh Hai
Quynh.Pham@Student.HTW-Berlin.de

www.mpmd.htw-berlin.de

htw