Industrial Power Control System Disturbance and Cyber-attack Discrimination

By: Victor Kushnir and Moriya Bitton

Github link:

What is it all about?

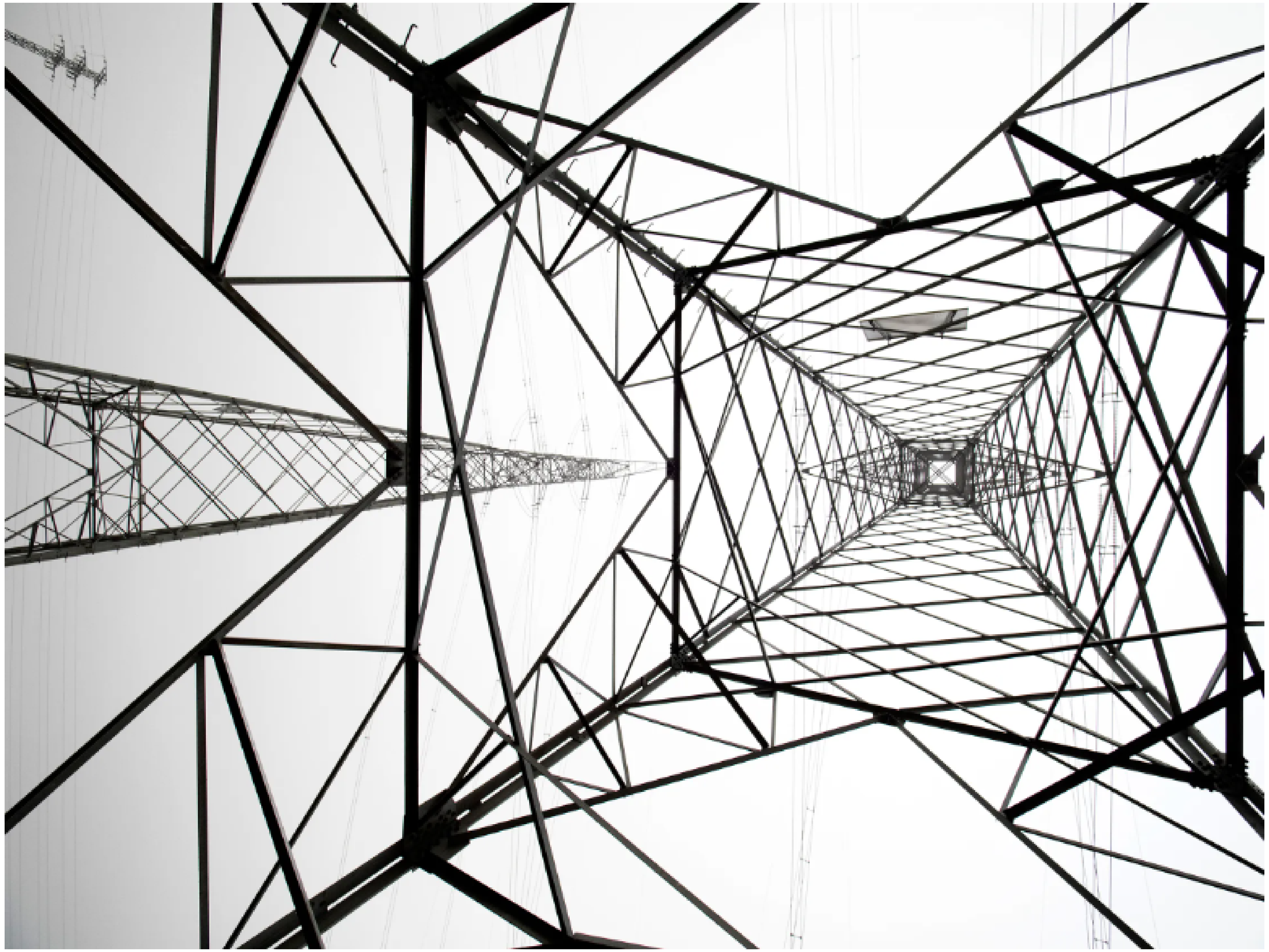
Industrial control systems (ICSs), also known as supervisory control and data acquisition (SCADA) systems, combine distributed computing with physical process monitoring and control. They are comprised of elements providing feedback from the physical world (sensors), elements influencing it (actuators), as well as computers and controller networks that process the feedback data and issue commands to the actuators.

Why is it relevant?

KIM ZETTER SECURITY MAR 3, 2016 7:00 AM

Inside the Cunning, Unprecedented Hack of Ukraine's Power Grid

The hack on Ukraine's power grid was a first-of-its-kind attack that sets an ominous precedent for the security of power grids everywhere.



JOSE A. BERNAT BACET/GETTY IMAGES

https://www.wired.com/2016/03/inside-cunning-unprecedented-hack-ukraines-power-grid/

EXCLUSIVE

Cyberattack Targets Safety System at Saudi Aramco

One report points to Iran, but the evidence is far from conclusive.

By Elias Groll

A flame from a Saudi Aramco oil installation burns brightly during sunset in the Saudi desert on June 23, 2008. (AFP/Marwan Naamani)

DECEMBER 21, 2017, 6:08 PM

https://foreignpolicy.com/2017/12/21/cyber-attack-targets-safety-system-at-saudi-aramco/

COMPUTERWORLD

NEWS

Iran was prime target of SCADA worm

The Stuxnet worm affecting SCADA systems may have been spreading since as early as January

By Robert McMillan

IDG News Service | JUL 23, 2010 8:45 PM PST

Computers in Iran have been hardest hit by a dangerous computer worm that tries to steal information from industrial control systems.

According to data compiled by Symantec, nearly 60 percent of all systems infected by the worm are located in Iran. Indonesia and India have also been hard-hit by the malicious software, known as Stuxnet.

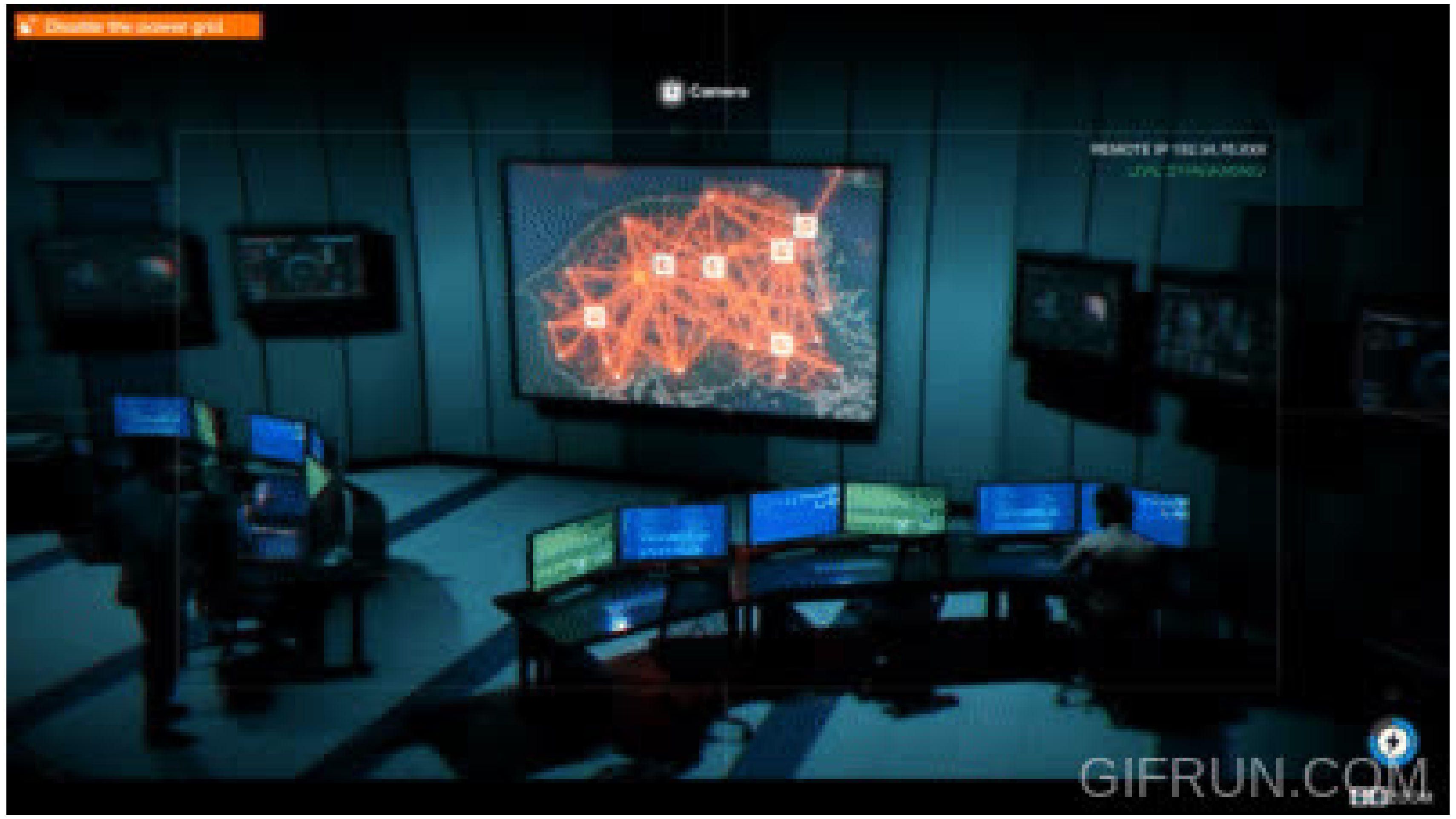
Looking at the dates on digital signatures generated by the worm, the malicious software may have been in circulation since as long ago as January, said Elias Levy, senior technical director with Symantec Security Response.

Stuxnet was discovered last month by VirusBlokAda, a Belarus-based antivirus company that said it found the software on a system belonging to an Iranian customer. The worm seeks out Siemens SCADA (supervisory control and data acquisition) management systems, used in large manufacturing and utility plants, and tries to upload industrial secrets to the Internet.

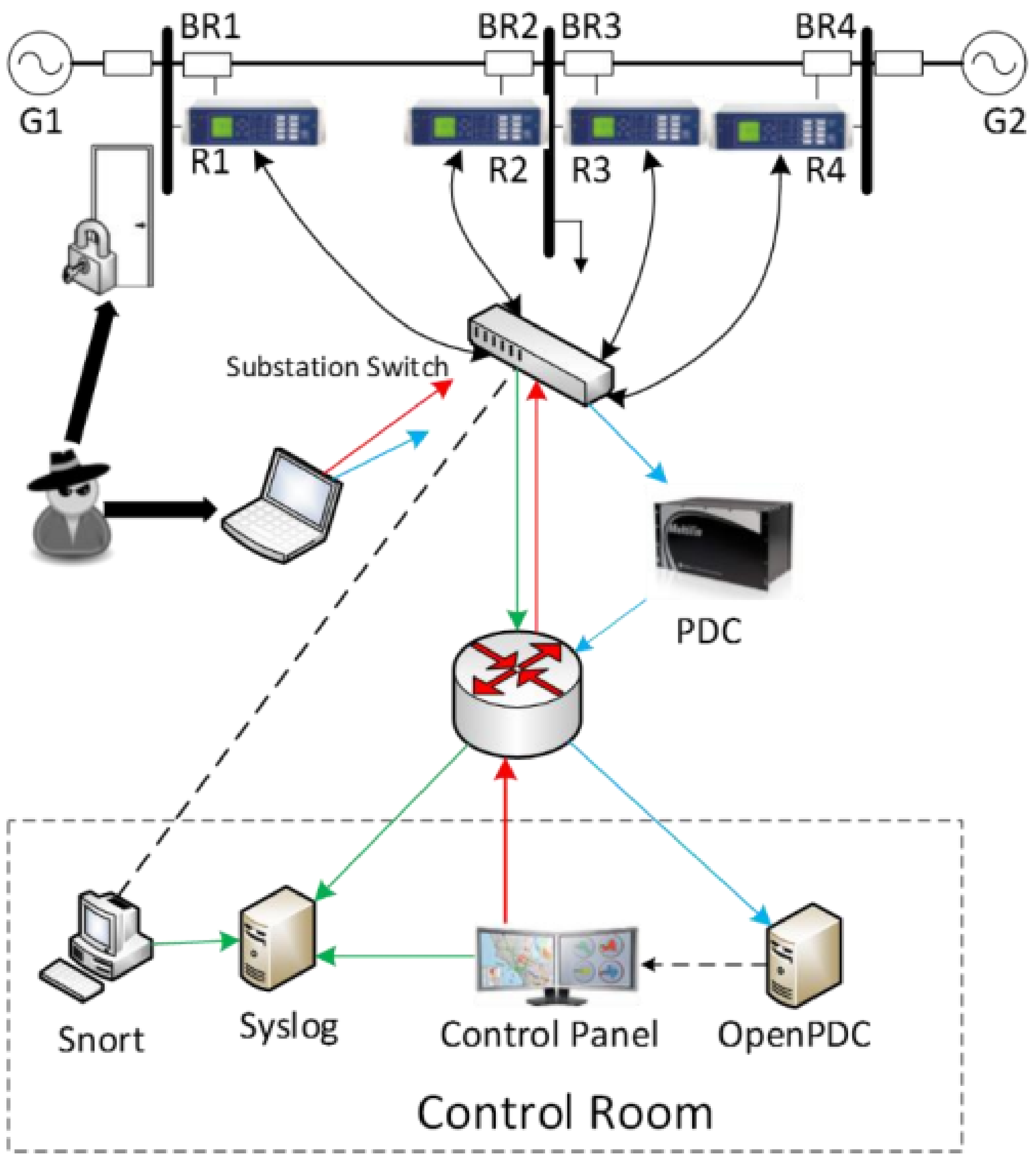
Symantec isn't sure why Iran and the other countries are reporting so many infections. "The most we can say is whoever developed these particular threats was targeting companies in those geographic areas," Levy said.

https://www.computerworld.com/article/2754136/iran-was-prime-target-of-scada-worm.html

WATCH_DOGS



ICS Power Systems



- G1 and G2 are power generators.
- R1 through R4 are Intelligent Electronic Devices (IEDs) that can switch the breakers on or off.
- These breakers are labeled BR1 through BR4.
- We also have two lines, Line One spans from breaker one (BR1) to breaker two (BR2) and Line Two spans from breaker three (BR3) to breaker four (BR4).
- Each IED controls a single breaker.
- BR1 is controlled by R1, BR2 is controlled by R2, and so on.
- The IEDs use a distance protection scheme that trips the breaker on detected faults whether actually valid or faked since they have no internal validation to detect the difference.
- Operators can also manually issue commands to the IEDs R1 through R4 to manually trip the breakers BR1 through BR4.
- Manual override is used when performing maintenance on the lines or other system components.

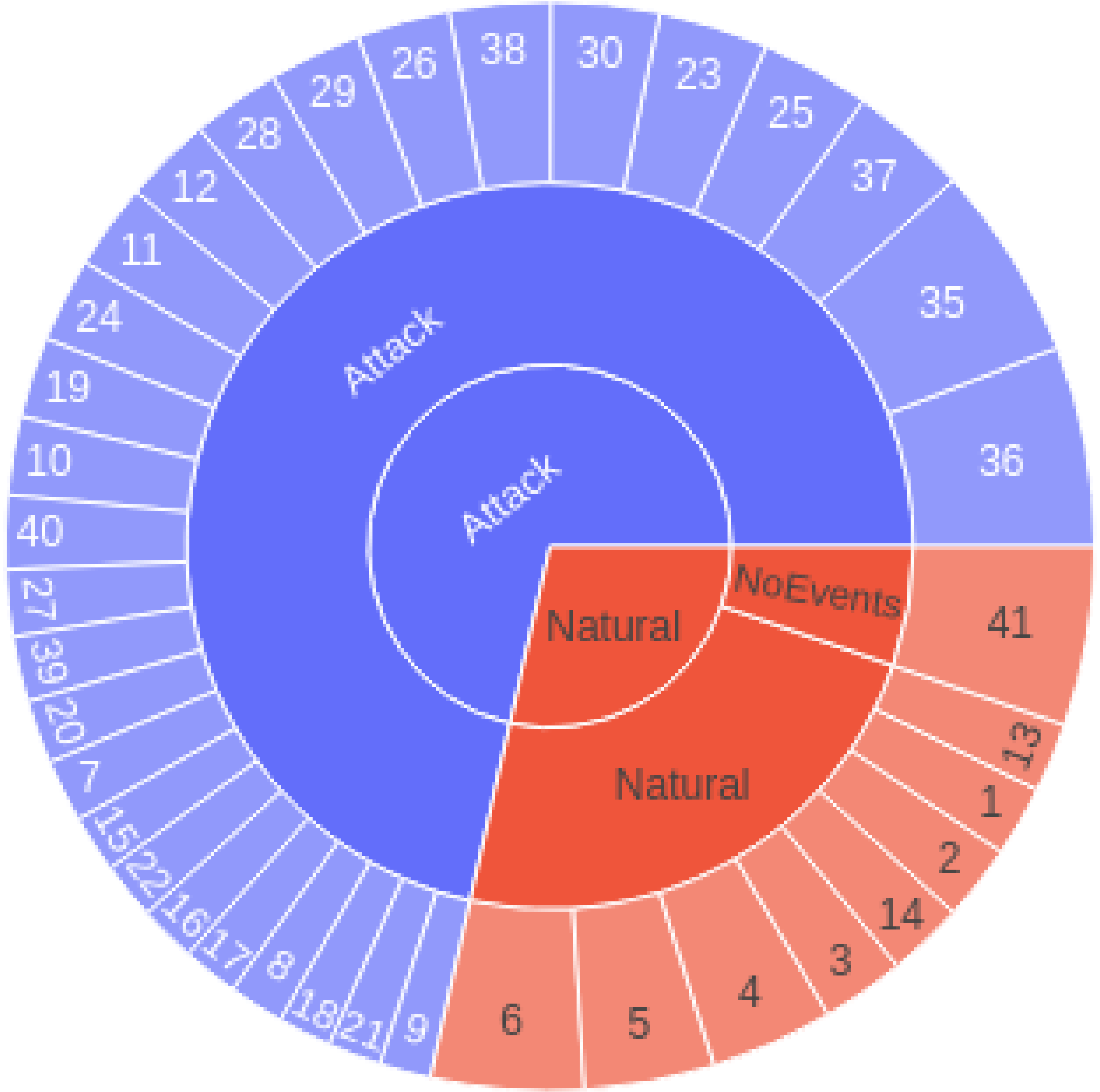
Types of Scenarios

- **Attack Scenarios**
    - **Data Injection**
        - **Attack Sub-type (SLG fault replay)**
            - **Fault from 10-19% on L1 with tripping command**
            - **Fault from 20-79% on L1 with tripping command**
            - **Fault from 80-90% on L1 with tripping command**
            - **Fault from 10-19% on L2 with tripping command**
            - **Fault from 20-79% on L2 with tripping command**
            - **Fault from 80-90% on L2 with tripping command**
    - **Remote Tripping Command Injection**
        - **Attack Sub-type (Command injection against single relay)**
            - **Command Injection to R1**
            - **Command Injection to R2**
            - **Command Injection to R3**
            - **Command Injection to R4**
        - **Attack Sub-type (Command injection against single relay)**
            - **Command Injection to R1 and R2**
            - **Command Injection to R3 and R4**
    - **Relay Setting Change**
        - **Attack Sub-type (Disabling relay function - single relay disabled & fault)**
            - **Fault from 10-19% on L1 with R1 disabled & fault**
            - **Fault from 20-90% on L1 with R1 disabled & fault**
            - **Fault from 10-49% on L1 with R2 disabled & fault**
            - **Fault from 50-79% on L1 with R2 disabled & fault**
            - **Fault from 80-90% on L1 with R2 disabled & fault**
            - **Fault from 10-19% on L2 with R3 disabled & fault**
            - **Fault from 20-49% on L2 with R3 disabled & fault**
            - **Fault from 50-90% on L2 with R3 disabled & fault**
            - **Fault from 10-79% on L2 with R4 disabled & fault**
            - **Fault from 80-90% on L2 with R4 disabled & fault**
        - **Attack Sub-type (Disabling relay function - two relays disabled & fault)**
            - **Fault from 10-49% on L1 with R1 and R2 disabled & fault**
            - **Fault from 50-90% on L1 with R1 and R2 disabled & fault**
            - **Fault from 10-49% on L1 with R3 and R4 disabled & fault**
            - **Fault from 50-90% on L1 with R3 and R4 disabled & fault**
        - **Attack Sub-type (Disabling relay function - two relay disabled & line maintenance)**
            - **L1 maintenance with R1 and R2 disabled**
            - **L1 maintenance with R1 and R2 disabled**

- **Natural Events**
    - **Natural events (SLG faults)**
        - **Fault from 10-19% on L1**
        - **Fault from 20-79% on L1**
        - **Fault from 80-90% on L1**
        - **Fault from 10-19% on L2**
        - **Fault from 20-79% on L2**
        - **Fault from 80-90% on L2**
    - **Natural events (Line maintenance)**
        - **Line L1 maintenance**
        - **Line L2 maintenance**

- **Regular Operation**
    - **Events (Normal operation)**
        - **Normal Operation load changes**

The datasets



• Multiclass - Each of the 37 event scenarios, which included attack events, natural events, and normal operations, was its own class and was predicted independently by the learners,

• Three-class – The 37 event scenarios were grouped into 3 classes: attack events (28 events), natural event (8 events) or “No events” (1 event).

• Binary – The 37 event scenarios were grouped as either an attack (28 events) or normal operations (9 events). The data was drawn from 15 data sets which included thousands of individual samples of measurements throughout the power system for each event type.

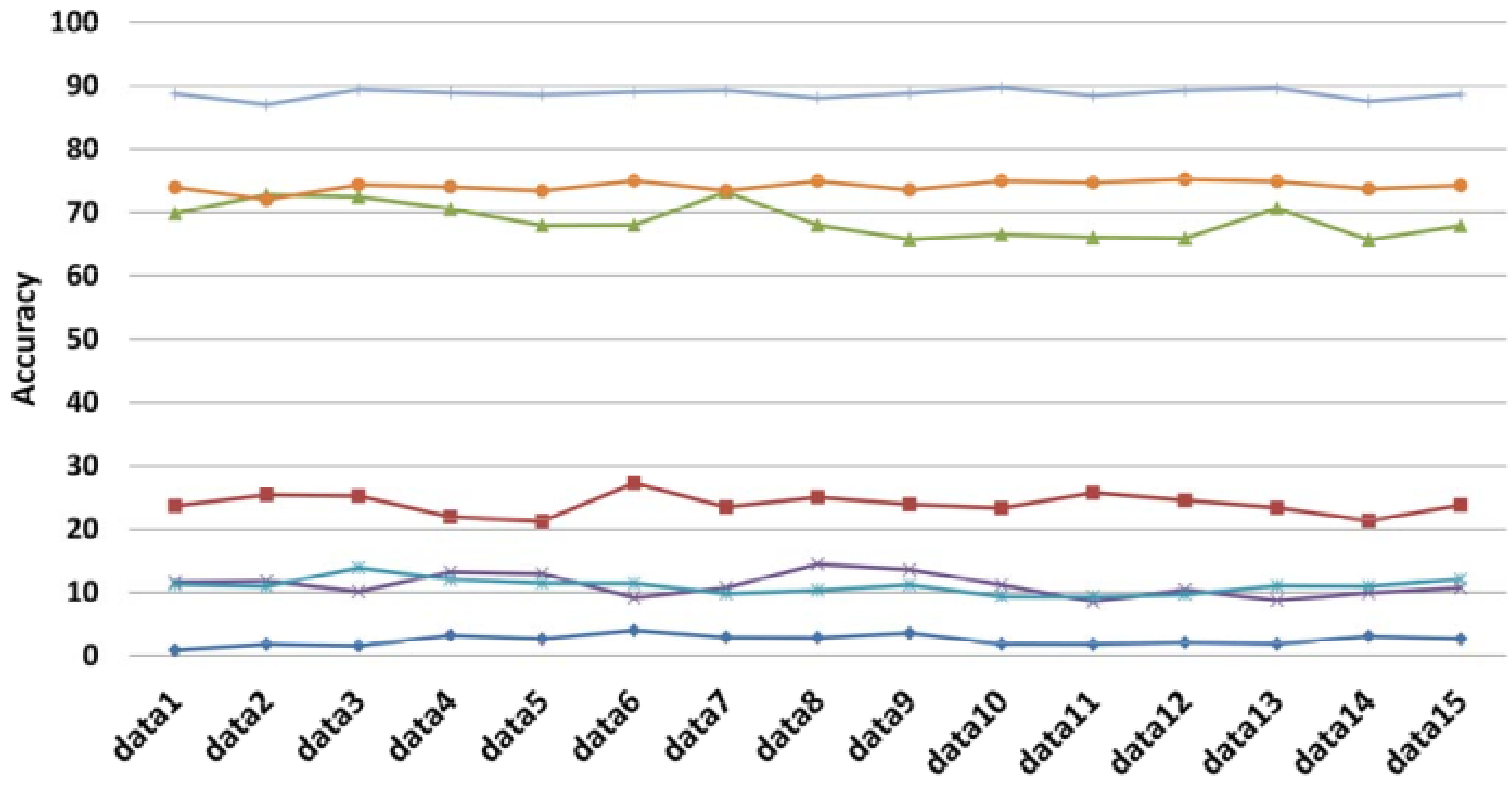
The features

Feature	Description
PA1:VH – PA3:VH	Phase A - C Voltage Phase Angle
PM1: V – PM3: V	Phase A - C Voltage Phase Magnitude
PA4:IH – PA6:IH	Phase A - C Current Phase Angle
PM4: I – PM6: I	Phase A - C Current Phase Magnitude
PA7:VH – PA9:VH	Pos. – Neg. – Zero Voltage Phase Angle
PM7: V – PM9: V	Pos. – Neg. – Zero Voltage Phase Magnitude
PA10:VH - PA12:VH	Pos. – Neg. – Zero Current Phase Angle
PM10: V - PM12: V	Pos. – Neg. – Zero Current Phase Magnitude
F	Frequency for relays
DF	Frequency Delta (dF/dt) for relays
PA:Z	Appearance Impedance for relays
PA:ZH	Appearance Impedance Angle for relays
S	Status Flag for relays

Previous work (The article)

The article proposed that using JRipper with Adaboost on the datasets is the most effective way.

(which is true, but this project should be graded so It can't just remain as it is)



JRipper – Incremental Reduced Error Pruning algorithm that uses a separate-and-conquer methodology, to generate a sophisticated rule set. Were built specifically for WEKA, which is an ML software in JAVA (So implementing the model in python was a real nightmare)

JRipper + Adaboost implementation

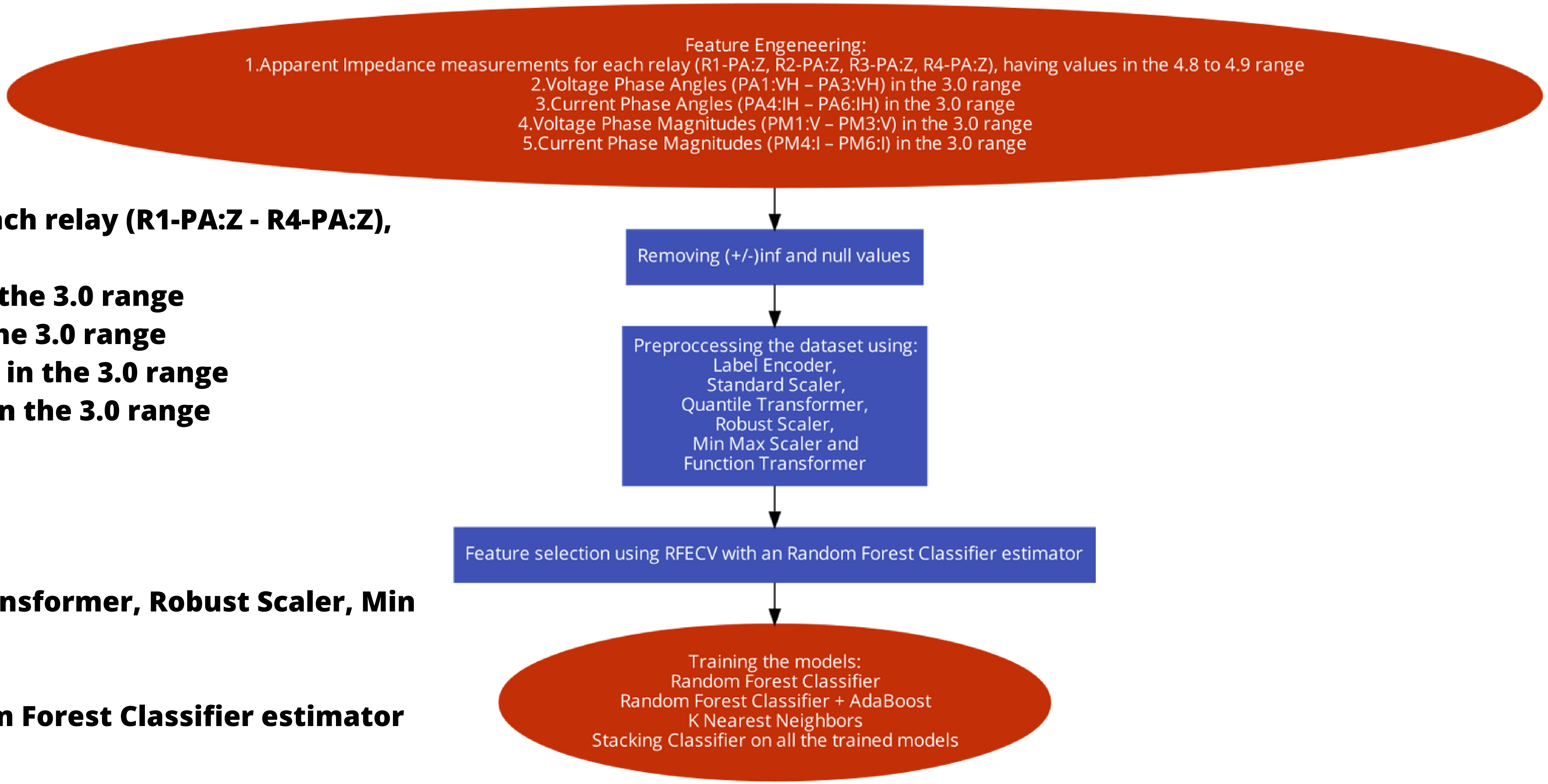
```python
# Jripper + Adaboost
jrp_adaboost = []
loader = Loader(classname="weka.core.converters.ArffLoader")
for filename in dataset_filenames:
    data = loader.load_file(filename)
    data.class_is_last()
    # Split the selected dataset into training and testing sets
    train, test = data.train_test_split(75)

    # Create a new Evaluation object for the selected attributes
    eval = Evaluation(train)

    # Build the classifier on the training datas
    base_cls = Classifier(classname="weka.classifiers.rules.JRip", options=["-F", "3", "-N", "2.0", "-O", "2"])
    cls = Classifier(classname="weka.classifiers.meta.AdaBoostM1", options=["-P", "100", "-S", "1", "-I", "10", "-W", base_cls.classname, "--"])
    cls.build_classifier(train)
    # Make predictions on the test data using the new Evaluation object
    random.seed(1)
    eval.crossvalidate_model(cls, test, 10, Random(1))
    print(eval.percent_correct)
    jrp_adaboost.append(cls)
```

✓ 158m 20.1s

```
88.01611278952669
90.63116370808679
91.5050784856879
92.40384615384616
91.66666666666667
92.14501510574019
90.83094555873926
92.38005644402634
91.01123595505618
93.6265709156194
89.14285714285714
94.92822966507177
90.60721062618596
91.49560117302053
91.75355450236967
```

Every line in the results is the results of the model for each one of the 15 datasets.

Our approach



1.Feature Engeneering:

1.Apparent Impedance measurements for each relay (R1-PA:Z - R4-PA:Z), having values in the 4.8 to 4.9 range

2.Voltage Phase Angles (PA1:VH – PA3:VH) in the 3.0 range

3.Current Phase Angles (PA4:IH – PA6:IH) in the 3.0 range

4.Voltage Phase Magnitudes (PM1:V – PM3:V) in the 3.0 range

5.Current Phase Magnitudes (PM4:I – PM6:I) in the 3.0 range

2.Removing (+/-)inf and null values

3.Preprocessing the dataset using:

Label Encoder, Standard Scaler, Quantile Transformer, Robust Scaler, Min Max Scaler and Function Transformer

4.Feature selection using RFECV with an Random Forest Classifier estimator

5.Training the models:

Random Forest Classifier

Random Forest Classifier + AdaBoost

K Nearest Neighbors

Stacking Classifier on all the trained models

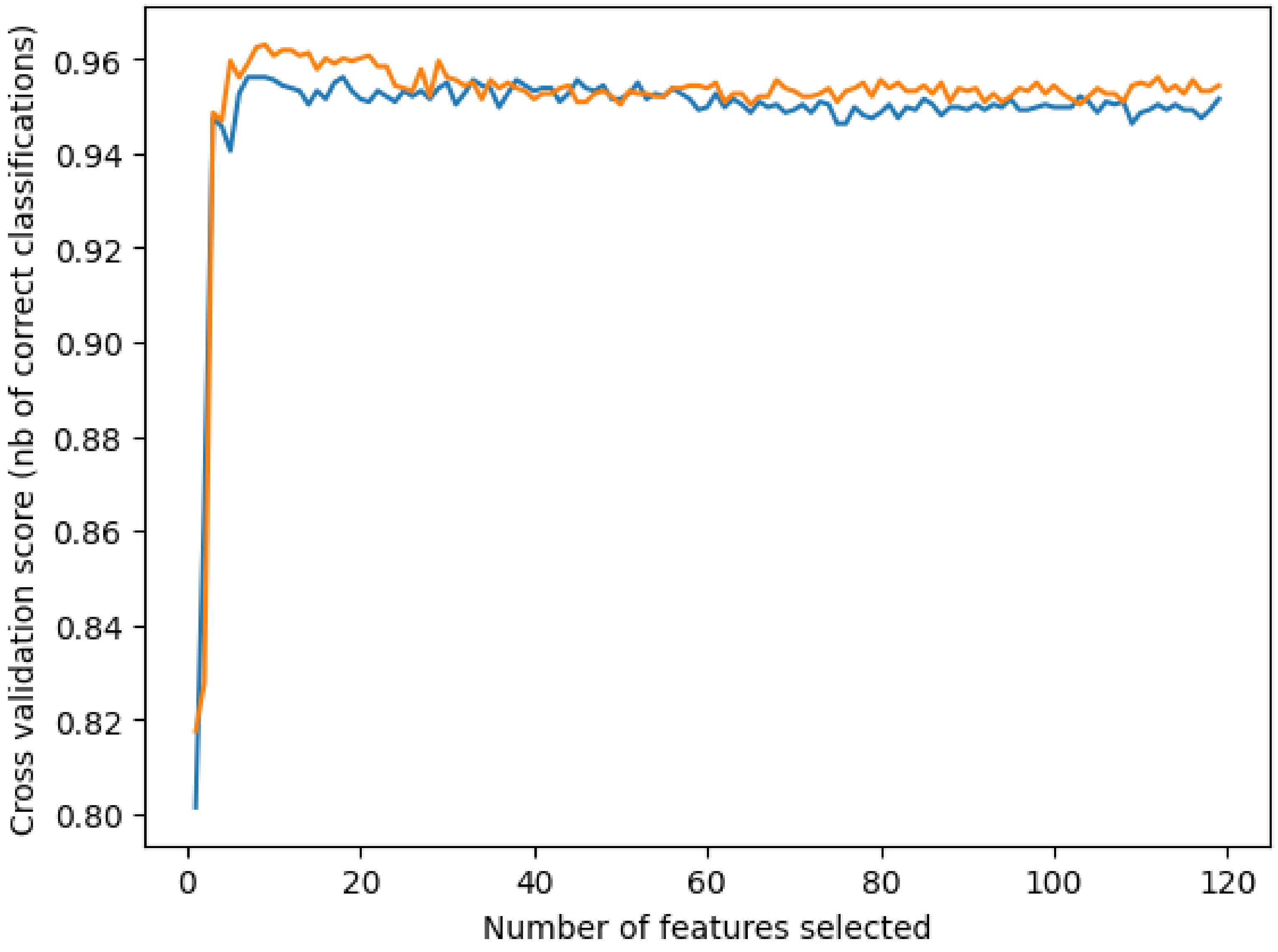
Preprocessing the data

Here we tried to use each technique to the most fitting parameters.

```python
# LabelEncoder encodes labels with a value between 0 and n_classes-1
le = LabelEncoder()
# StandardScaler scales values by subtracting the mean and dividing by the standard deviation
ss = StandardScaler()
# QuantileTransformer transforms features using quantiles information
qt = QuantileTransformer()
# RobustScaler scales values by subtracting the median and dividing by the interquartile range
rs = RobustScaler()
# MinMaxScaler scales values between 0 and 1
mms = MinMaxScaler()
# LogTransformer transforms features by taking the natural logarithm
lt = FunctionTransformer(np.log1p)
# Preprocessing
def vectorize_df(df):
    df_numeric = df.select_dtypes(include=[np.number])
    # Perform label encoder on marked column
    df['marker'] = le.fit_transform(df['marker'])
    for column in df_numeric.columns:
        if column == 'marker':
            continue
        column_data = df_numeric[column]
        # To avoid Input X contains infinity or a value too large for dtype('float64') error we replace them with float.max
        column_data = column_data.replace([np.inf, -np.inf], np.finfo(np.float64).max)
        # Check if the data is normally distributed
        if column_data.skew() < 0.5:
            df_numeric[column] = ss.fit_transform(column_data.values.reshape(-1,1))
        # Check if the data has extreme outliers
        elif column_data.quantile(0.25) < -3 or column_data.quantile(0.75) > 3:
            df_numeric[column] = rs.fit_transform(column_data.values.reshape(-1,1))
        # Check if the data has a Gaussian-like distribution
        elif 0.5 < column_data.skew() < 1:
            df_numeric[column] = lt.fit_transform(column_data.values.reshape(-1,1))
        # Check if the data can be transformed into a Gaussian-like distribution
        elif column_data.skew() > 1:
            df_numeric[column] = qt.fit_transform(column_data.values.reshape(-1,1))
        else:
            df_numeric[column] = mms.fit_transform(column_data.values.reshape(-1,1))
            df[df_numeric.columns] = df_numeric
    return df

df_mult = vectorize_df(df_mult)
```

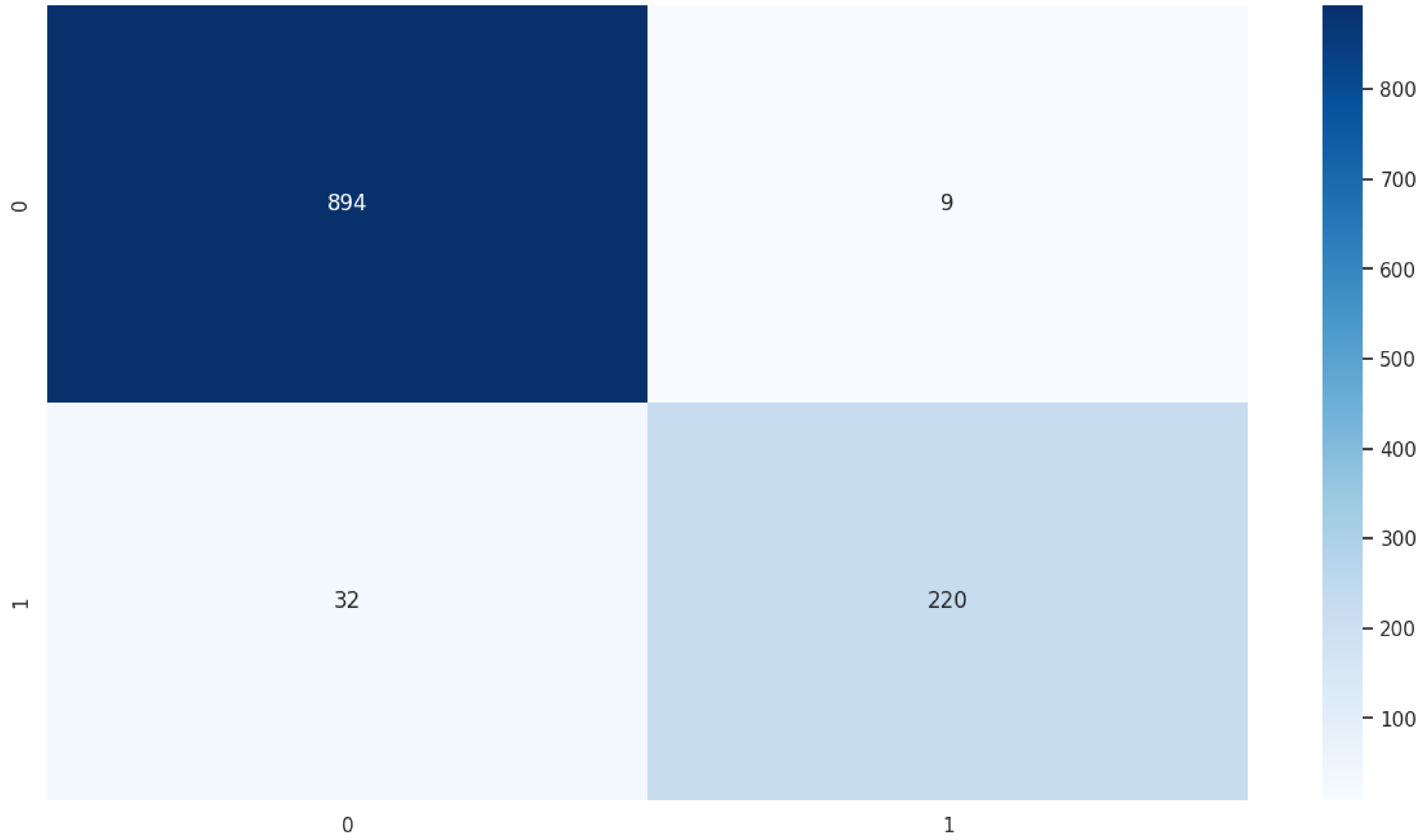
Feature selection

The optimal number of features is between 5 to 25 (from the original 180), depending on the dataset.



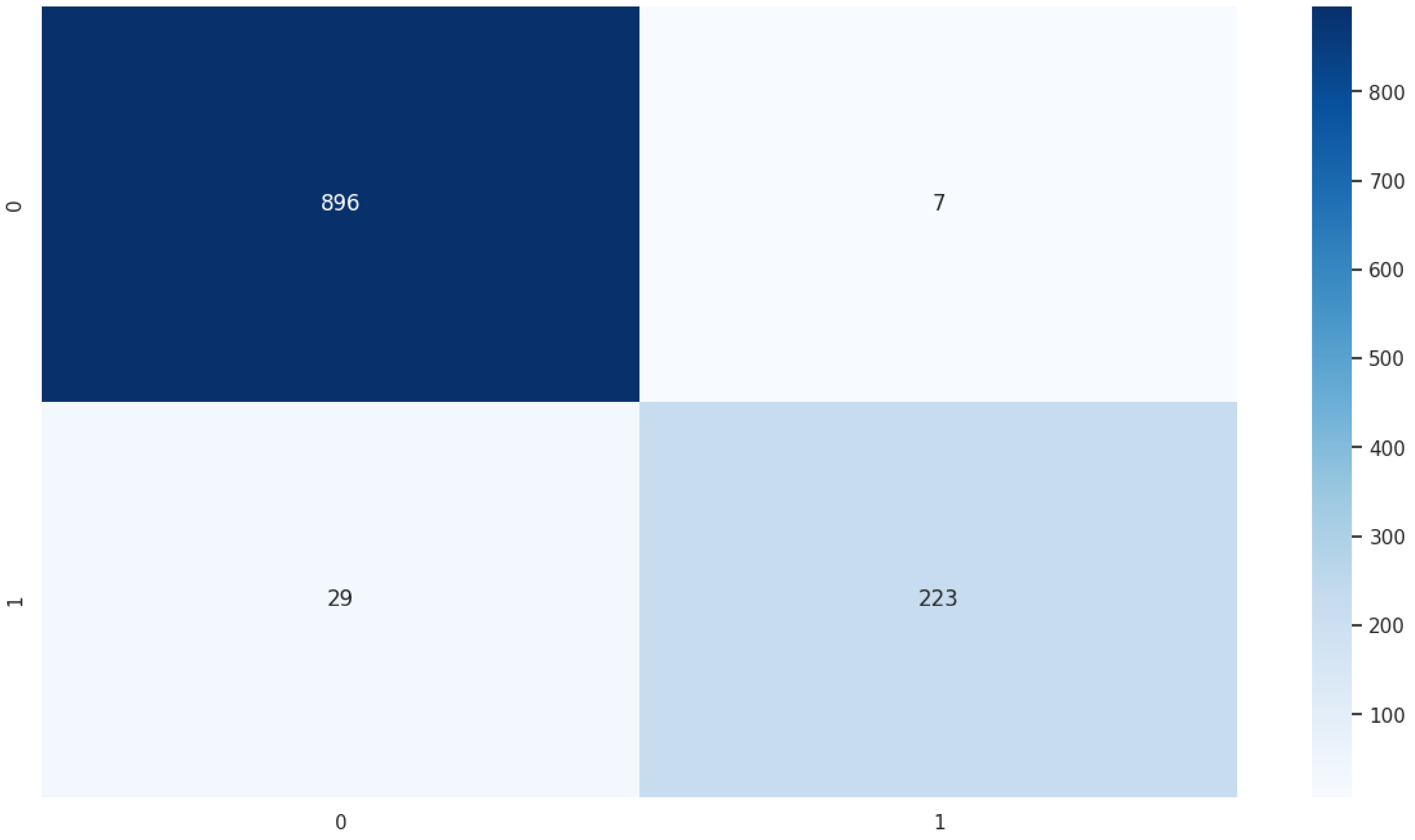
Random Forest Classifier

	precision	recall	f1-score	support
0	0.96544	0.99003	0.97758	903
1	0.96070	0.87302	0.91476	252
accuracy			0.96450	1155
macro avg	0.96307	0.93152	0.94617	1155
weighted avg	0.96441	0.96450	0.96388	1155



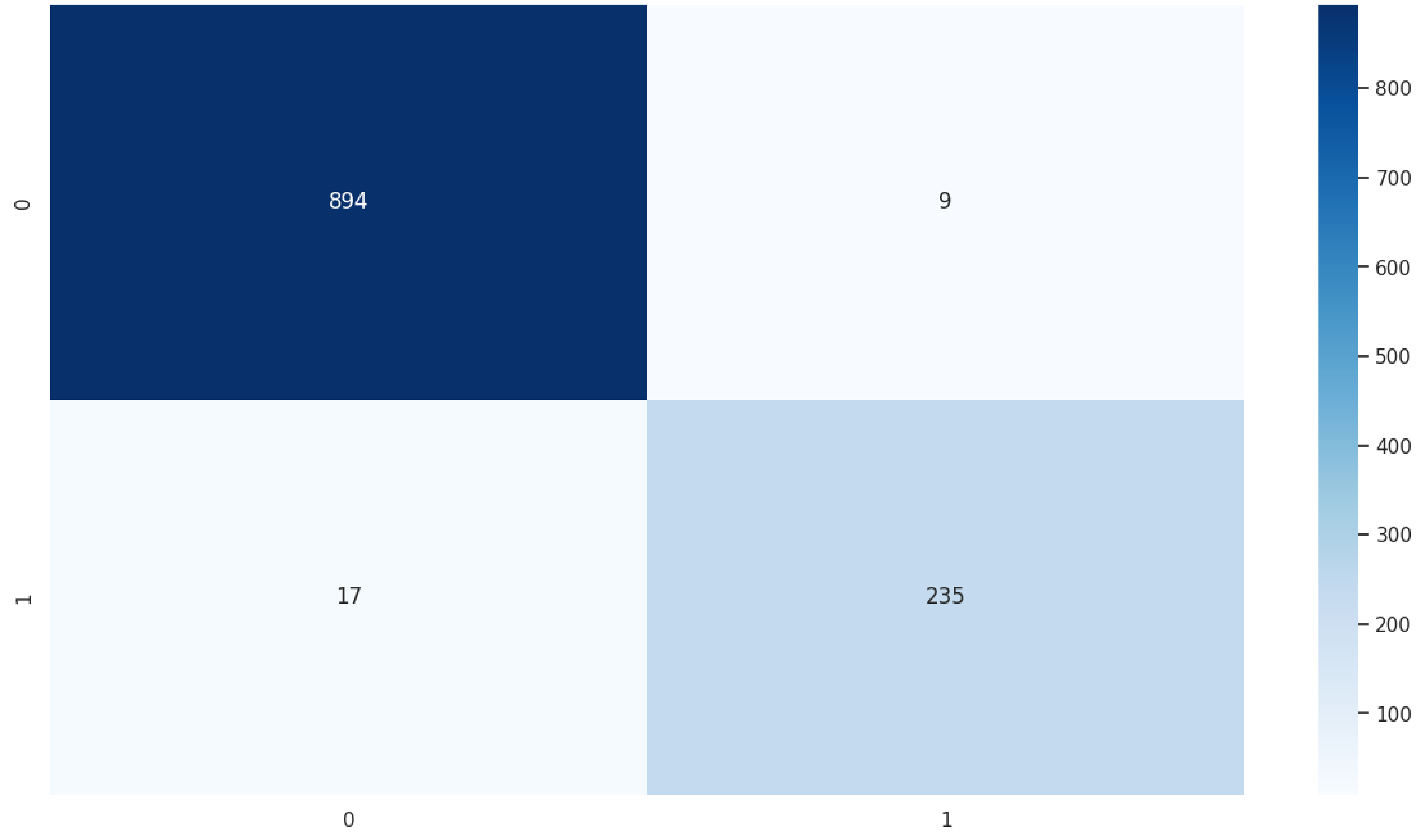
Random Forest Classifier + AdaBoost

	precision	recall	f1-score	support
0	0.96865	0.99225	0.98031	903
1	0.96957	0.88492	0.92531	252
accuracy			0.96883	1155
macro avg	0.96911	0.93858	0.95281	1155
weighted avg	0.96885	0.96883	0.96831	1155



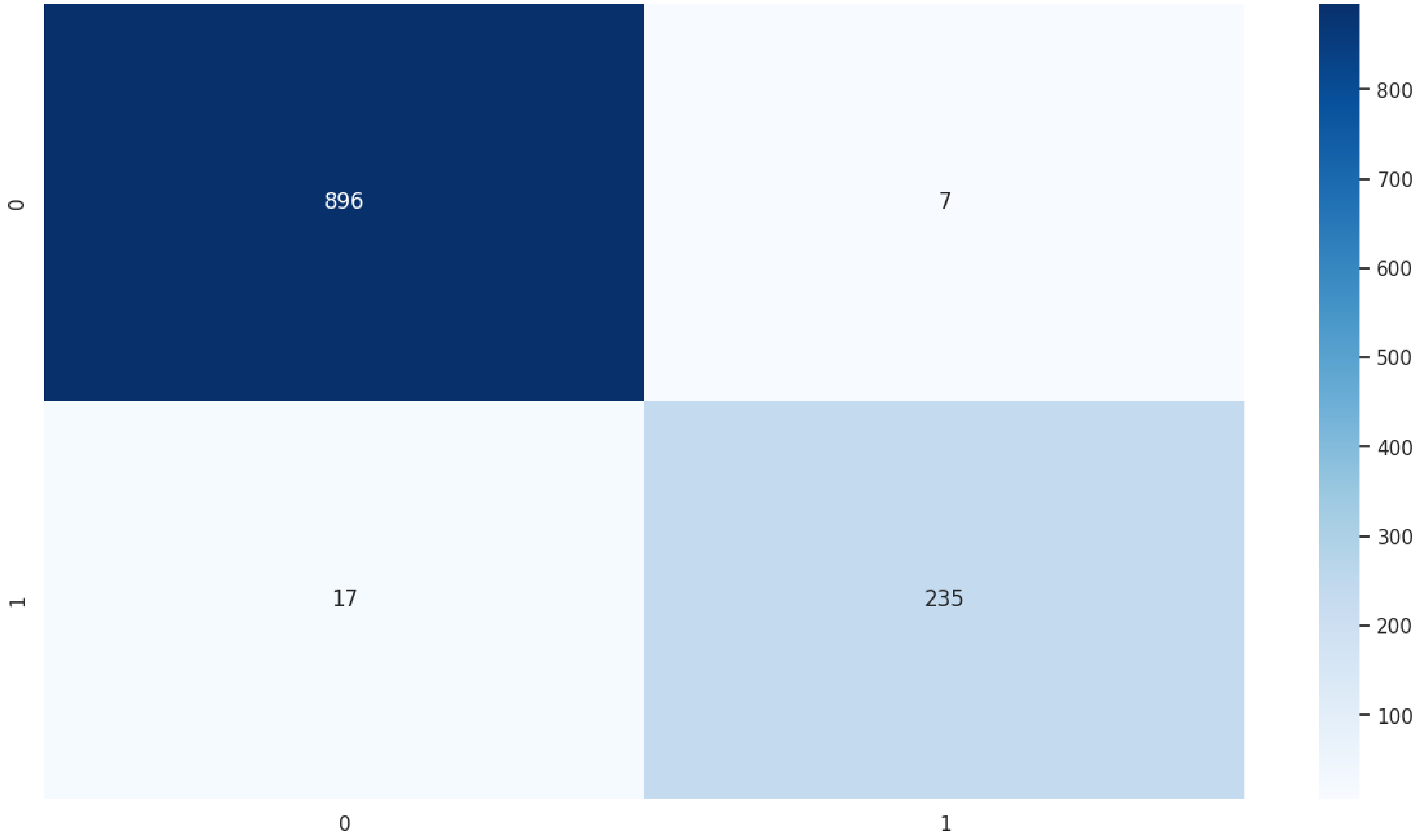
K-Nearest Neighbors

	precision	recall	f1-score	support
0	0.98134	0.99003	0.98567	903
1	0.96311	0.93254	0.94758	252
accuracy			0.97749	1155
macro avg	0.97223	0.96129	0.96662	1155
weighted avg	0.97736	0.97749	0.97736	1155



Stacking Classifier

	precision	recall	f1-score	support
0	0.98138	0.99225	0.98678	903
1	0.97107	0.93254	0.95142	252
accuracy			0.97922	1155
macro avg	0.97623	0.96239	0.96910	1155
weighted avg	0.97913	0.97922	0.97907	1155



Thank you!