E-Surveillance Alert Classification

Group Members:
Sridhar
Kamakshi
Sarah
Param

Overview

- The Problem
- Exploring the dataset
- Preprocessing of Data
- Feature Selection
- Train Test Split
- Model
    - KNN
    - Random Forest
    - RandomForest with RandomizedSearchCV
- Results
- Discussion

Problem Statement

Prevent break-ins before they occur using IoT security cameras with built-in computer vision capabilities, reducing the need for human intervention. Automated security to safeguard and alert against threats from intrusion or fire using multi-capability sensors such as vibration, motion, smoke, fire, panic switches etc. Ensure the safety of both monetary and intellectual assets with round-the-clock surveillance and controlled access management.

We are tasked with classifying the alert whether it is Critical, Normal, or Testing which is received from the various sensors. Such as vibration, motion, smoke, fire, Panic, shutter(Door sensor).

Features

- 15 features
- All expect 1 object type
- Target feature is Status
- 185687 rows in dataset

Sensor Used

- PIR
- Vibration
- Hooter
- Front Shutter
- Stop Panic
- Arm Disarm keypad
- Panic
- Smoke
- Energy Vault
- Chest Door
- UPS Room Door
- ATM Back Door
- Vibration

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 185687 entries, 0 to 185686
Data columns (total 15 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   DATE                    185687 non-null  object
 1   SOL ID                  185687 non-null  object
 2   SENSOR (As of Portal)   185687 non-null  object
 3   Region                  185687 non-null  object
 4   STATE                   185687 non-null  object
 5   SENSOR NAME (Standard)  185679 non-null  object
 6    EVENT                  185687 non-null  object
 7   EVENT DATE AND TIME     185687 non-null  object
 8   AKNOWLEDGE STATUS       185687 non-null  object
 9   Confirmation            185687 non-null  object
 10  LOG ID                  185687 non-null  int64
 11  2nd AKNOWLEDGE STATUS   185687 non-null  object
 12  Status                  185687 non-null  object
 13  Month                   185687 non-null  object
 14  Reason                  185687 non-null  object
dtypes: int64(1), object(14)
memory usage: 21.3+ MB
```

Target Values

True_Normal: Testing the sensors(Smoke, Panic, fire), Smoke Alert due to AC maintenance / UPS maintenance, Cash loading, Pressing the panic switches unknowingly by bank staffs…

True_Critical: Smoke, Fire, Network Connection error, PIR, Panic(Fire, Theft Attempt, in ATM and Bank)

Breaking the ATM Machine(Vibration sensor will be activated)

Thieves are showing the Weapons to Bank Staff (Panic switch must be pressed by Bank Staff to get the alert)

False_Normal: While opening the bank, make sure to enter the password to change the mode. Otherwise, PIR and Alarm will be generated

Due to rats movement, PIR will be generated in the night time

False_Critcal: No activity, Sensor Malfunctioning (keep on getting the alert). Alert is received from the critical sensor even though there is no activity. Such as Smoke, fire, PIR(Motion detection sensor)

Output Features(cont'd)

Some interesting reasons:

*During renovation work at the bank (smoke and fire sensor will be activated due to dust)

*During sanitization of the branch.

* Cleaning or sweeping the branch.

* Auditing(Auditor will check by burning the papers inside the branch)

* Birthday celebration(candle smoke will generate the alert)


TheQuint.com

Prepocessing

- Dropped the unnecessary columns
  - LOG ID
  - Sensor (as of portal)
  - DATE
  - SENSOR NAAME (standard)
  - Month
  - EVENT DATE AND TIME
  - Reason
- Check if dataset had any missing data
- Used LabelEncoder to change object type features to int
  - Pro: doesn't add more columns
  - cons : has inherent bias

```
p.isna().sum()
```

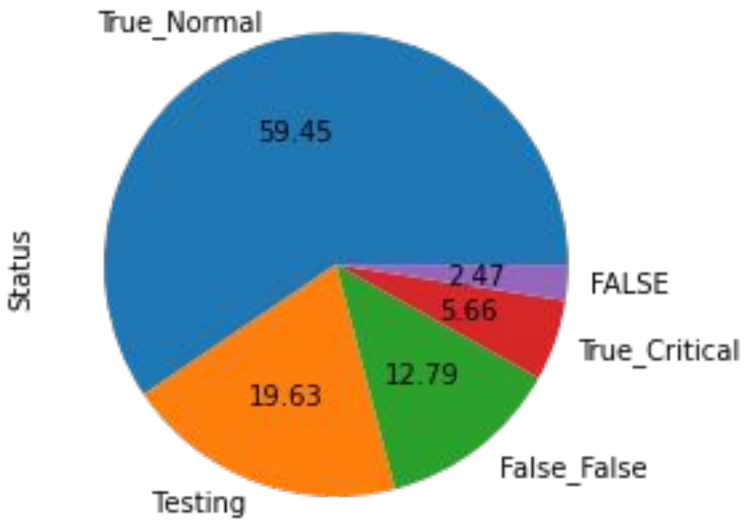
```
SOL ID                   0
Region                   0
STATE                    0
 EVENT                   0
AKNOWLEDGE STATUS        0
Confirmation             0
2nd AKNOWLEDGE STATUS    0
Status                   0
dtype: int64
```

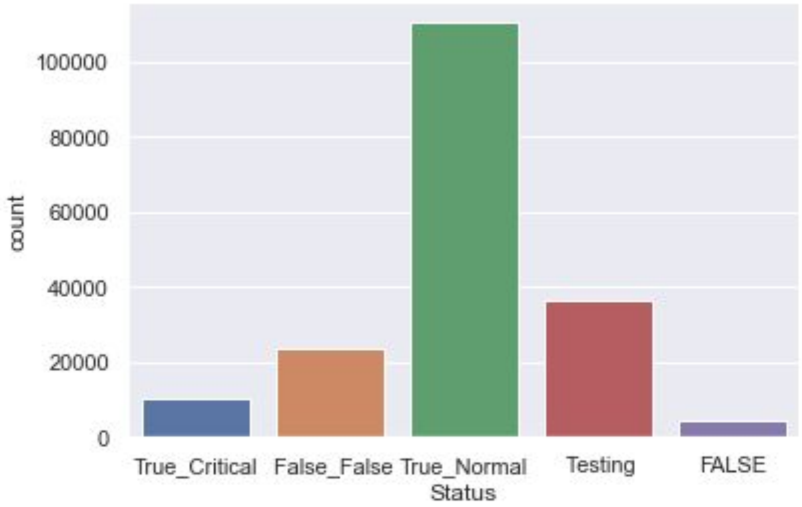
```
from sklearn.preprocessing import LabelEncoder
labelencoder_X=LabelEncoder()
xm=p.apply(LabelEncoder().fit_transform)
xm
```

	SOL ID	Region	STATE	EVENT	AKNOWLEDGE STATUS	Confirmation	2nd AKNOWLEDGE STATUS	Status
0	1255	3	11	6	7872	14	8079	3
1	1166	2	17	8	434	15	442	0
2	1092	2	0	6	7872	14	8079	3
3	1525	3	11	6	7872	14	8079	3
4	11	1	15	6	7872	14	8079	3
...	...	...	...	...	...	...	...	...
185682	1369	3	16	5	8337	19	8587	4
185683	1369	3	16	12	8337	19	8587	4
185684	807	0	13	10	2968	19	3045	1
185685	807	0	13	5	2968	19	3045	1
185686	807	0	13	12	2968	19	3045	1

185687 rows × 8 columns

Checking the data imbalance

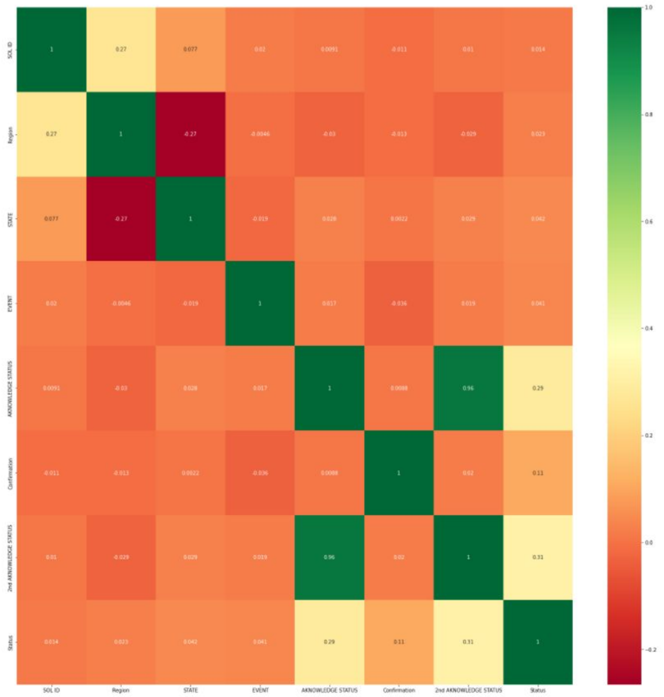


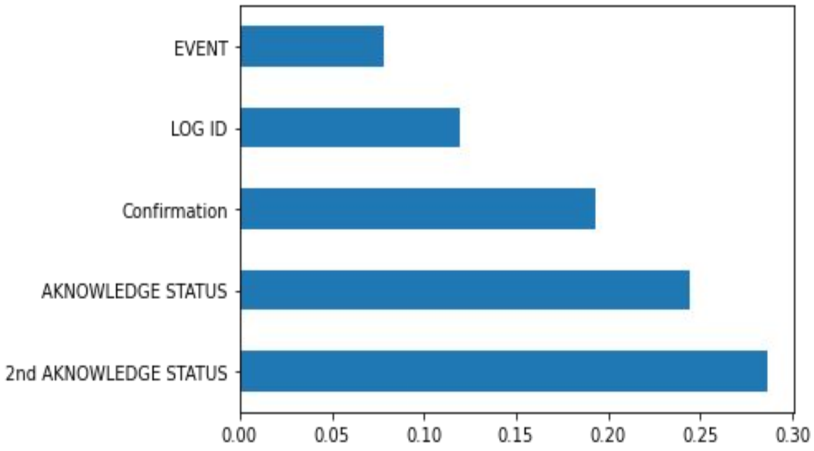


Data has been imbalanced. Majority of the data's are fall under True Normal

Feature Selection

- ❏ Preformed Feature selection using ExtratreesClassifier
  - ❏ Improve accuracy and avoid overfitting
  - ❏ Top 5 features displayed in bar graph





- ❏ Correlation heat map of features
- ❏ Final features selected were:
  - ❏ Region
  - ❏ State
  - ❏ Event
  - ❏ Acknowledge Status
  - ❏ 2nd Acknowledge Status
  - ❏ Confirmation

Performance Metric

When working with imbalanced data, we don't recommend using categorical accuracy as the main evaluation measure. It is not unusual to observe a high evaluation accuracy when testing a classification model trained on very imbalanced data.

Precision: how many selected instances are relevant.
Recall: how many relevant instances are selected.
F1 score: harmonic mean of precision and recall.
Confusion Matrix:
Macro Average:
The macro-average gives every class the same importance, and therefore better reflects how well the model performs.

Train and Test Construction

Train and Test Construction

We build train and test by randomly splitting in the ratio of 70:30 or 80:20 whatever we choose as we have sufficient points to work with.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, random_state=5)

print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(129980, 8)
(55707, 8)
(129980,)
(55707,)
```

Building Machine Learning with Imbalanced dataset

1. KNN

2.RandomForest

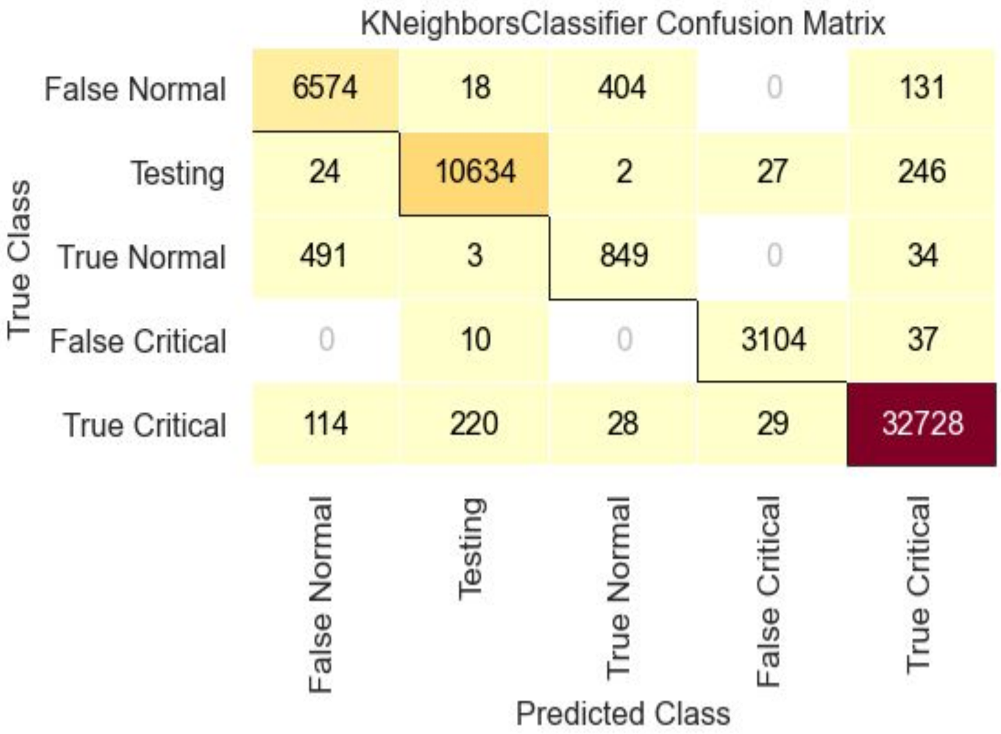
3. Random Forest with RandomizedSearchCV

KNearest Neighborsclassifier

Confusion Matrix

- First we ran a K nearest Neighbor Classifier to

All classes are classified well except True Normal



KNearestNeighborClassifier

- Macro Avg precision and recall gives 0.90 accuracy

```
from sklearn.metrics import classification_report
print(classification_report(y_test,knn_predictions , target_names =data_features ))
```

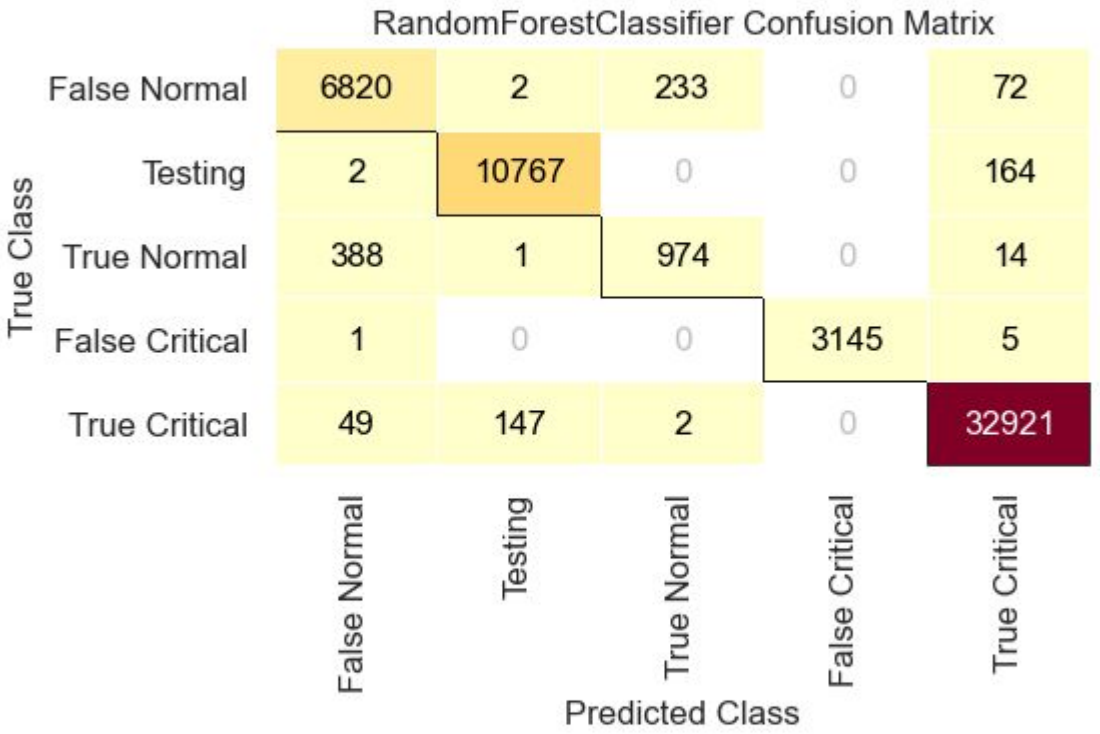
```
               precision    recall  f1-score   support

 True_Critical       0.91      0.92      0.92      7127
   False_False       0.98      0.97      0.97     10933
   True_Normal       0.66      0.62      0.64      1377
       Testing       0.98      0.99      0.98      3151
         FALSE       0.99      0.99      0.99     33119

      accuracy                           0.97     55707
     macro avg       0.90      0.90      0.90     55707
  weighted avg       0.97      0.97      0.97     55707
```

RandomForest Classifier

- All are classified well as compared to True Normal



RandonForest Classifier

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred, target_names =data_features ))
```

```
               precision    recall  f1-score   support

True_Critical       0.94      0.96      0.95      7127
  False_False       0.99      0.99      0.99     10933
  True_Normal       0.80      0.71      0.75      1377
      Testing       1.00      1.00      1.00      3151
        FALSE       0.99      0.99      0.99     33119

     accuracy                           0.98     55707
    macro avg       0.94      0.93      0.94     55707
 weighted avg       0.98      0.98      0.98     55707
```

Random Forest with RandomizedSearchCV

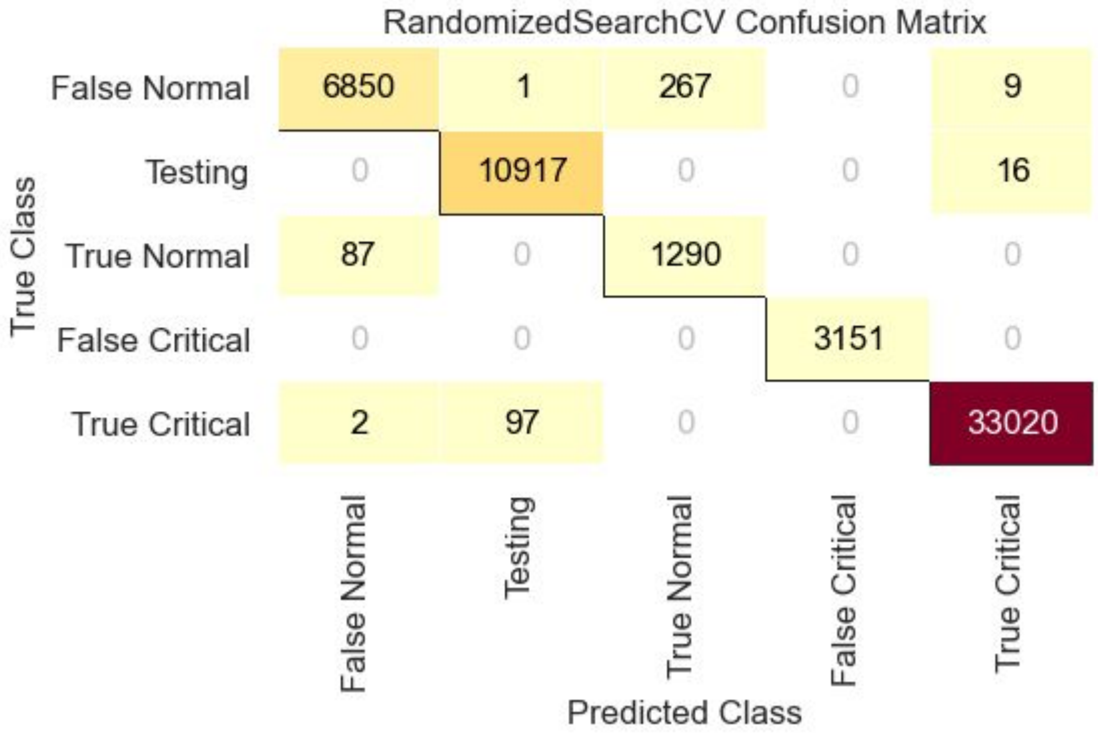
Explore the best parameters

{'n_estimators': 94, 'min_samples_split': 2, 'min_samples_leaf': 1, 'max_features':

0.8999999999999999, 'max_depth': 18, 'bootstrap': True}

```
RandomizedSearchCV(cv=3, estimator=RandomForestClassifier(random_state=50),
                   n_jobs=-1,
                   param_distributions={'bootstrap': [True, False],
                                        'max_depth': [3, 4, 6, 8, 9, 11, 13, 1
4,
                                                      16, 18, 20, None],
                                        'max_features': ['auto', 'sqrt', 0.5,
                                                         0.6, 0.7,
                                                         0.7999999999999999,
                                                         0.8999999999999999],
                                        'min_samples_leaf': [1, 2, 4],
                                        'min_samples_split': [2, 5, 10],
                                        'n_estimators': [10, 31, 52, 73, 94,
                                                         115, 136, 157, 178,
                                                         200]},
                   random_state=50, verbose=2)
```

Random Forest with RandomizedSearchCV Confusion Matrix



- Here, We can see that True Critical is perfectly classified except few errors

Random Forest with RandomizedSearchCV Confusion Matrix

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred1, target_names =data_features ))
```

	precision	recall	f1-score	support
True_Critical	0.93	0.97	0.95	7127
False_False	0.98	1.00	0.99	10933
True_Normal	0.82	0.66	0.73	1377
Testing	1.00	1.00	1.00	3151
FALSE	1.00	0.99	1.00	33119
accuracy			0.98	55707
macro avg	0.95	0.92	0.93	55707
weighted avg	0.98	0.98	0.98	55707

Results

- Improvement from KNN to Random Forest models
- Little improvement when the RandomizedSearchCV was used
- Also seen in the confusion matrix
- Most common mislabel in all models was true_normal being predicted as true_critical

	Accuracy	Error
KNN	: 96.74%	3.264%
Random Forest	: 98.09%	1.914%
Random Forest with RandomizedSearchCV	: 98.28%	1.716%

What is the issue with imbalanced dataset?

Most models trained on imbalanced data will have a bias towards predicting the larger class(es) and, in many cases, may ignore the smaller class(es) altogether.

As a result, the instances belonging to the smaller class(es) are typically misclassified more often than those belonging to the larger class(es)

How to deal with imbalance dataset?

Resample the training set

Oversampling — Duplicating samples from the minority class

Undersampling — Deleting samples from the majority class

Combining Both Random Sampling Techniques

Combining both random sampling methods can occasionally result in overall improved performance in comparison to the methods being performed in isolation.

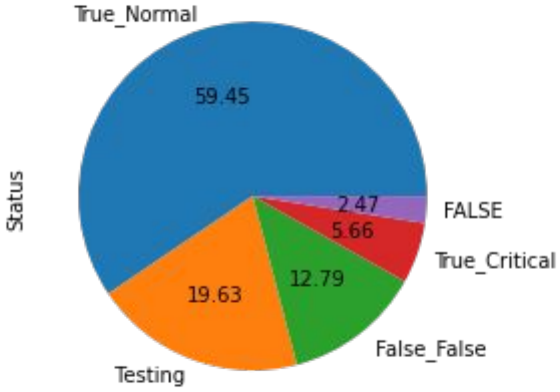
Building the model with balanced data

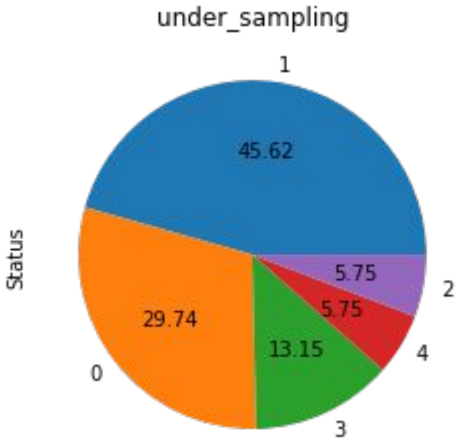
1. RandomForest(Under sampling)
2. RandomForest(Over sampling)
3. RandomForest(Combined sampling)
4. RandomizedSearchCV(Under Sampling)
5. RandomizedSearchCV(Over Sampling)
6. RandomizedSearchCV(Combined Sampling)

Under Sampling

Deleting samples from the majority class

Before Under Sampling **After under sampling**





Before undersampling: Counter({4: 77036, 1: 25630, 0: 16609, 3: 7423, 2: 3282})

After undersampling: Counter({1: 36442, 0: 23755, 3: 10505, 2: 4590, 4: 4590})

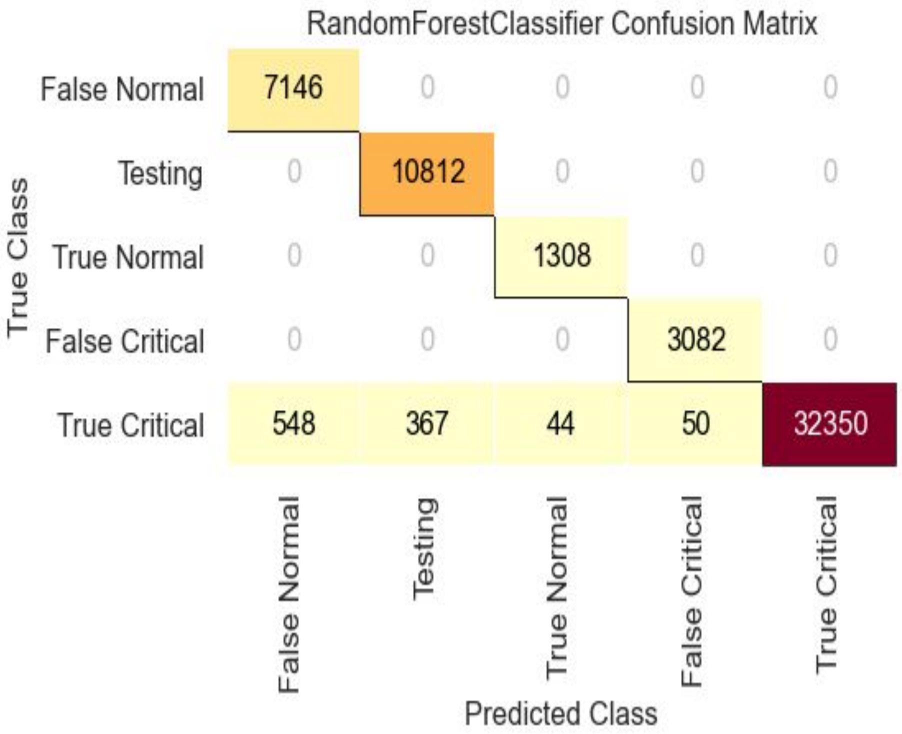
RandomForestClassifier with under sampling

Test Accuracy-----0.9802538280646956

Train Accuracy ----1.0

It is overfitting
it can discard potentially useful information which could be important for building rule classifiers.

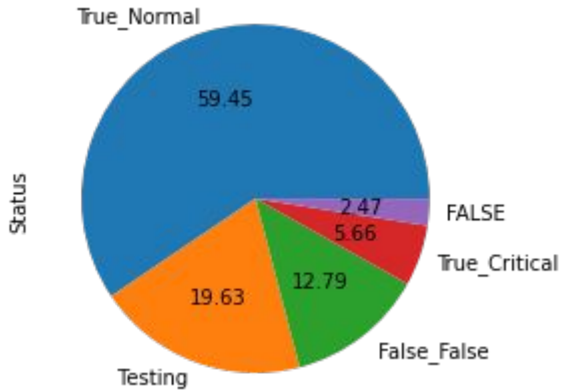
The sample chosen by random under sampling may be a **biased sample**. And it will not be an **accurate representative of the population**. Thereby, resulting in inaccurate results with the actual test data set.



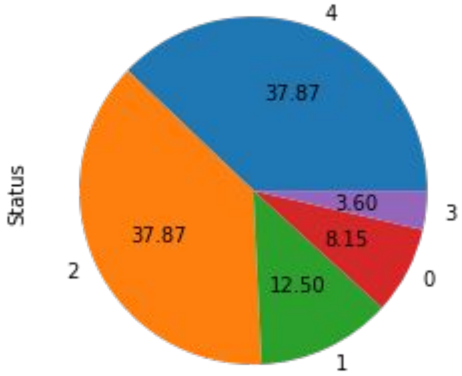
Oversampling

Duplicating samples from the minority class

Before Oversampling



After Oversampling

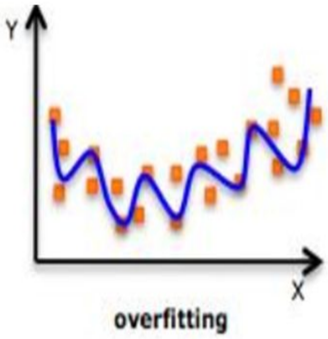


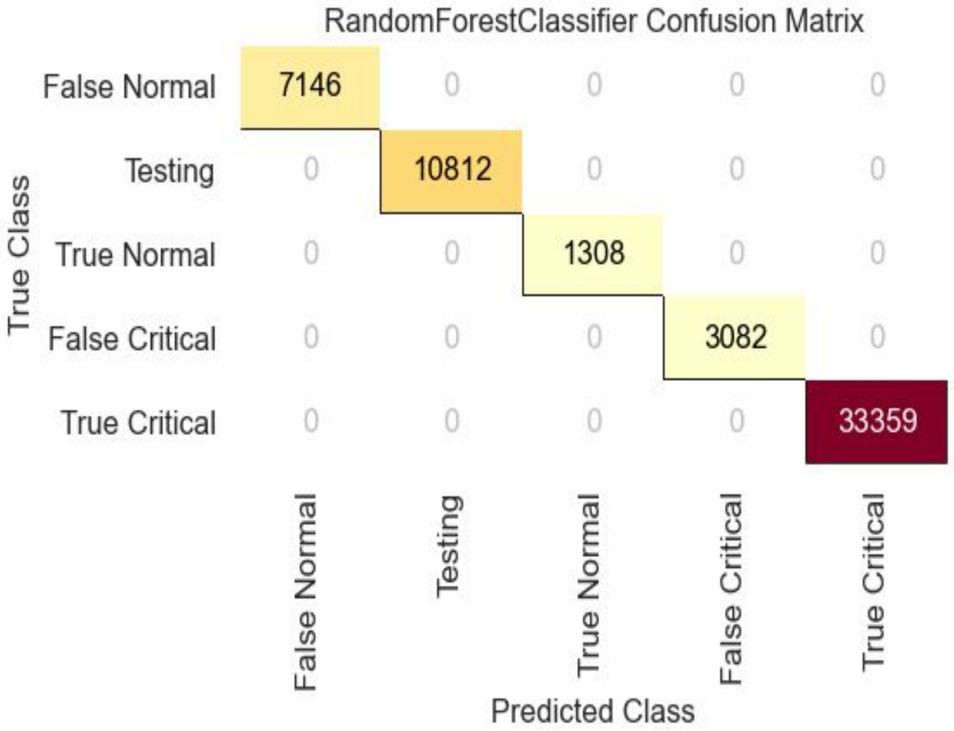
Random Forest Classifier with Oversampling

It increases the likelihood of overfitting since it replicates the minority class events.

Test Accuracy =1.0

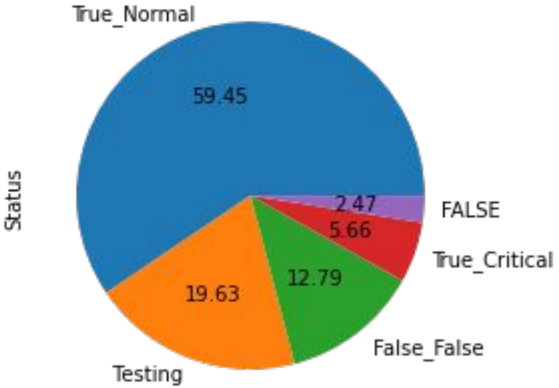
Train Accuracy=1.0



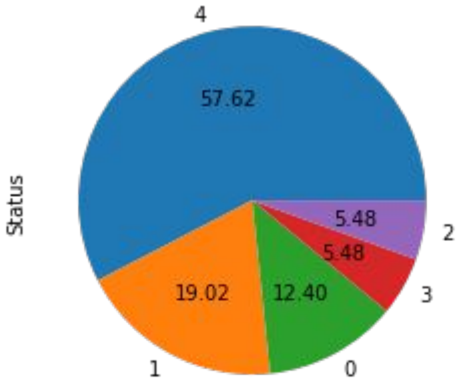


Combined Sampling

Before



After Combined Sampling



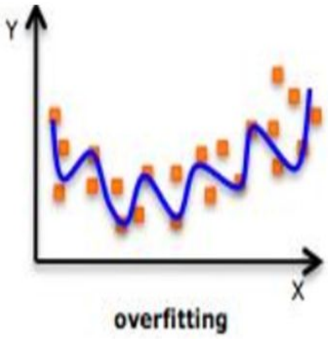
The concept is that we can apply a modest amount of oversampling to the minority class, which improves the bias to the minority class examples, whilst we also perform a modest amount of undersampling on the majority class to reduce the bias on the majority class examples.

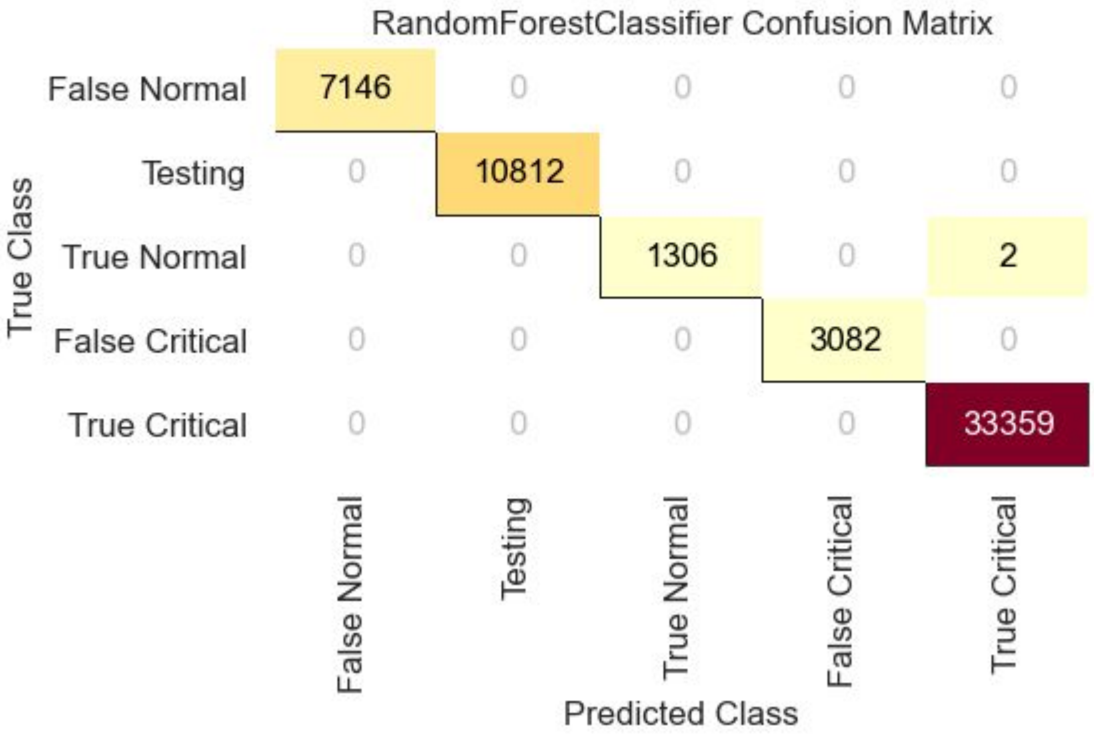
RandomForestClassifier with combined sampling

Test Accuracy=0.9999640978692086

Train Accuracy= 1.0

It is totally overfitting



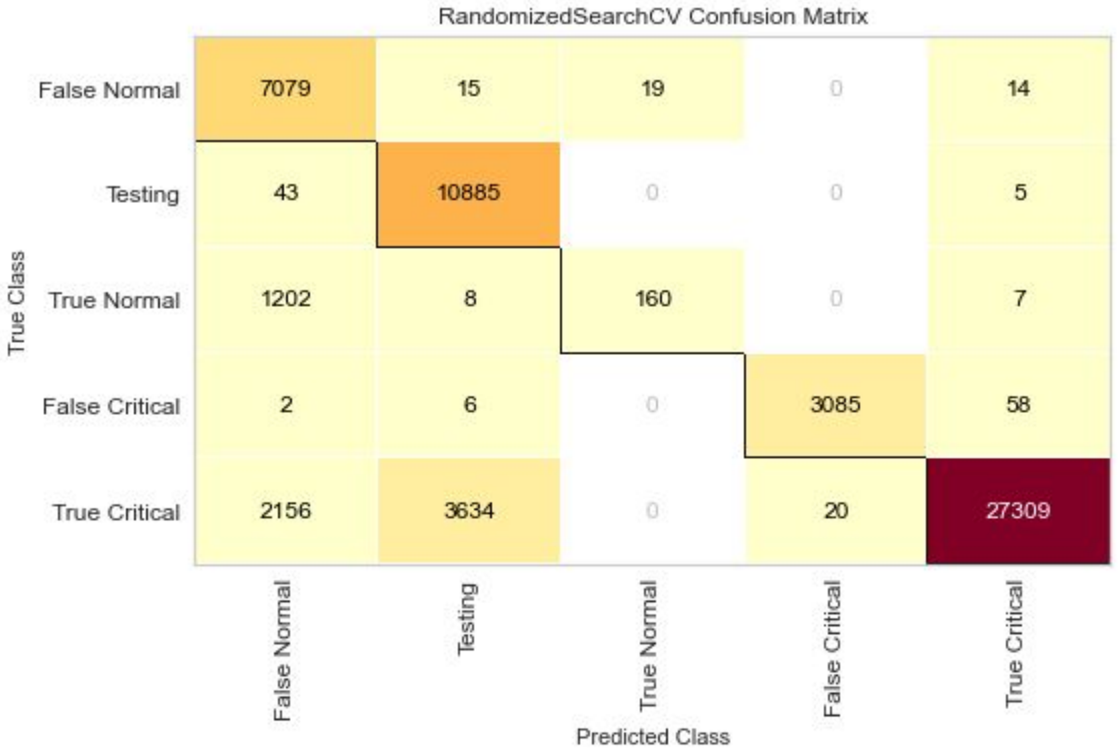


RandomForest Classifier with Undersampling

RandomizedSearchCV

Test Accuracy=0.8709497908700882
Train Accuracy=0.9321123657394658

Misclassification is high for
True Normal and True Critical



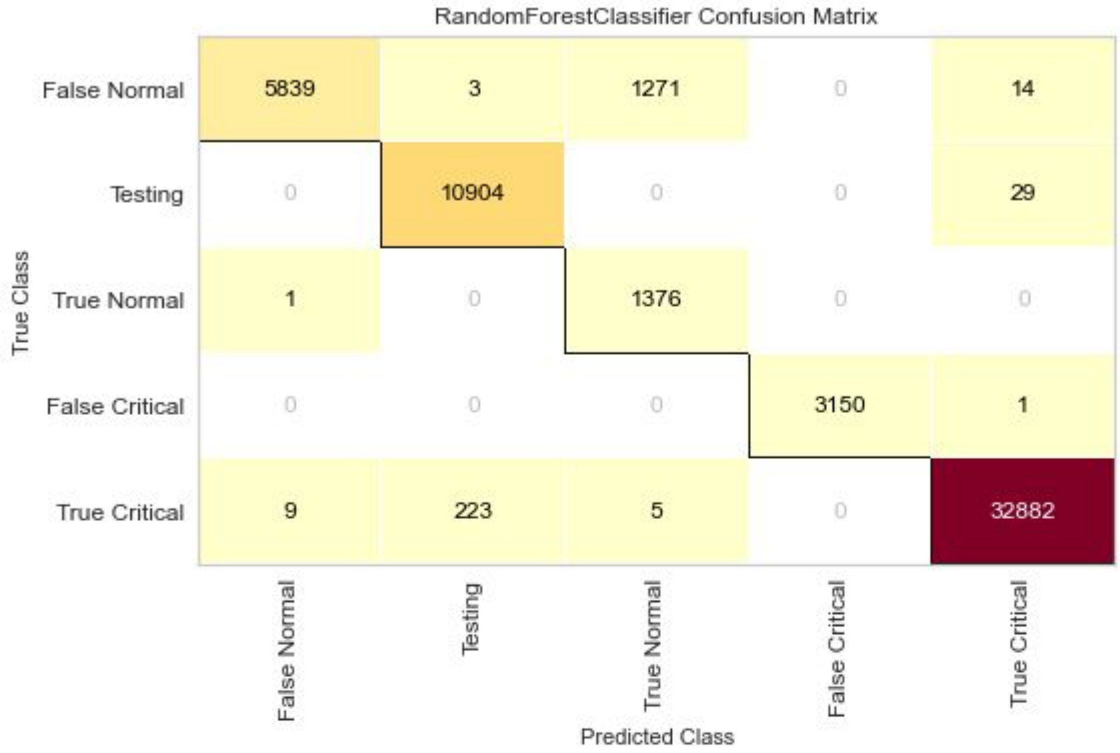
RandomForestClassifier with oversampling

RandomizedSearchCV

Test=0.9720681422442422
Train=0.9823082623193775

All classes classified perfectly except False Normal

Optimal



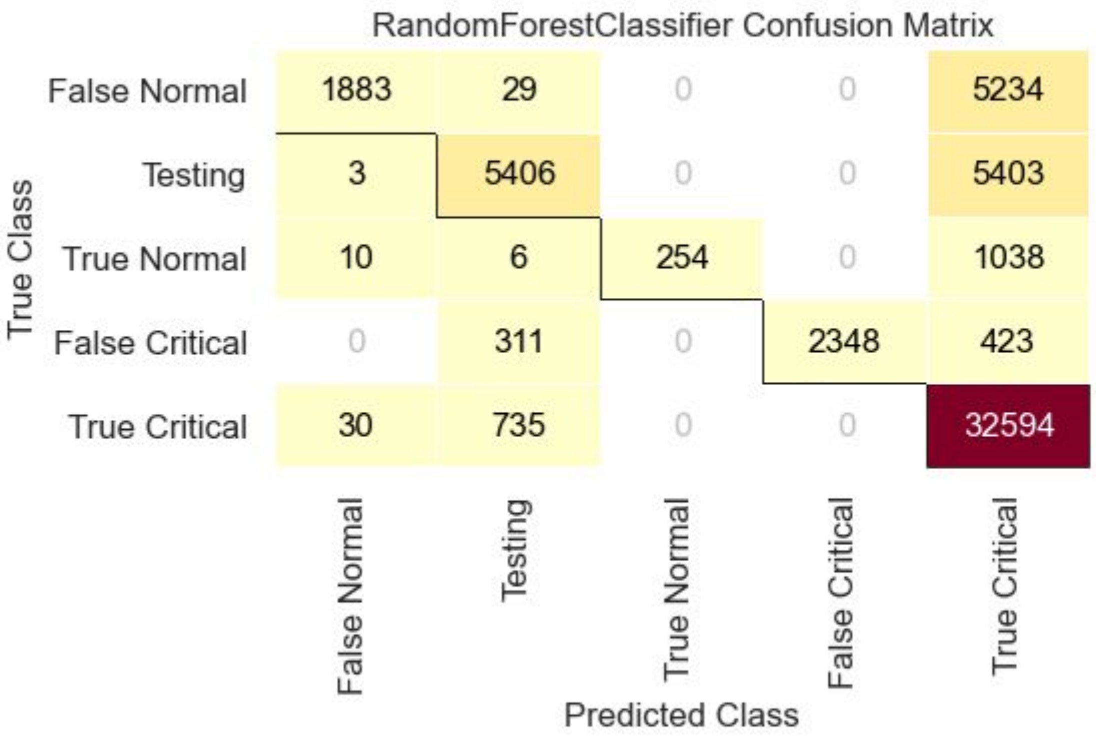
RandomForestClassifier with combined sampling

RandomizedSearchCV

Test-0.7626510133376416

Train-0.7823095792319495

Here, False Normal, True Normal and Testing are heavily misclassified



Summary

Balanced/Imbalanced data	Model	Sampling	Accuracy	Accuracy	Fitting
Imbalanced data	KNN	None	97.8	96.74	balanced
Imbalanced data	RandomForest	None	98.25	98.09	balanced
Imbalanced data	RandomForest with RadomizedSearchCV	None	98.34	98.28	balanced
Balanced data	RandomForest	Under Sampling	100	98	overfitting
Balanced data	RandomForest	OverSampling	100	100	overfitting
Balanced data	RandomForest	Combined Sampling	100	99.9	overfitting
Balanced data	RandomForest with RadomizedSearchCV	Under Sampling	93	87	balanced
Balanced data	RandomForest with RadomizedSearchCV	OverSampling	98	97	balanced
Balanced data	RandomForest with RadomizedSearchCV	Combined Sampling	78	76	balanced

Results

- RandomForest (Oversampling) with RandomizedSearchCV is the optimal model for this dataset.
- Most of the models are overfitting due to resampling the training data.
- Most importantly, Random Forest Classifier worked well even with imbalanced dataset.

Any questions?