Data Mining Applications

Final Project

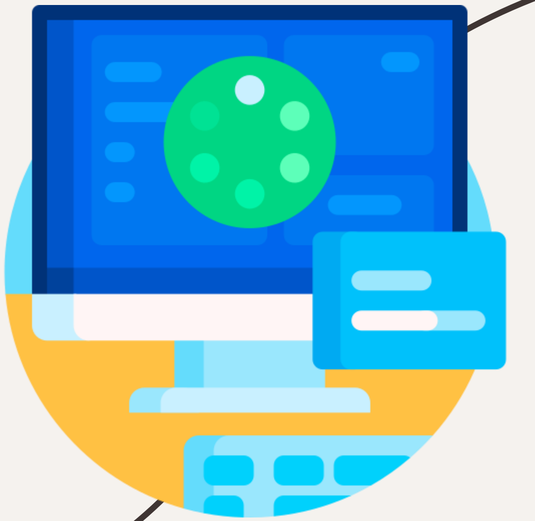
Contents

01

Introduction

Background

- A complex modern semi-conductor manufacturing process is normally under consistent surveillance via the monitoring of signals/variables collected from sensors and or process measurement points.
- Not all of these signals are equally valuable in a specific monitoring system.



Background

- Consider each type of signal as a feature, then feature selection may be applied to identify the most relevant signals.
- The Process Engineers may use these signals to determine key factors contributing to yield excursions downstream in the process.
- Enable an increase in process throughput, decreased time to learning and reduce the per unit production costs.

02

Literature Review

Literature Review

Reference 1

Xu, Z., Shen, D., Kou, Y., and Nie, T., 2022, “A Synthetic Minority Oversampling Technique Based on Gaussian Mixture Model Filtering for Imbalanced Data Classification,” IEEE Transactions on Neural Networks and Learning Systems, Early Access, 1-14.

Reference 2

Wazery, Y. M., Saber, E., Houssein, E. H., Ali, A. A., and Amer, E., 2021, “An Efficient Slime Mould Algorithm Combined With K-Nearest Neighbor for Medical Classification Tasks,” IEEE Access, Vol. 9, 113666-113682.

Reference 3

Cervantes, J., Garcia-Lamont, F., Rodríguez-Mazahua, L., and Lopez, A., 2020, “A comprehensive survey on support vector machine classification: Applications, challenges and trends,” Neurocomputing, Vol. 408, No. 30, 189-215.

Reference 4

Demir, S., and Sahin, E. K., 2022, “Comparison of tree-based machine learning algorithms for predicting liquefaction potential using canonical correlation forest, rotation forest, and random forest based on CPT data,” Soil Dynamics and Earthquake Engineering, Vol. 154, 107-130.

Reference 1

"A Synthetic Minority Oversampling Technique Based on Gaussian Mixture Model Filtering for Imbalanced Data Classification."

Objective :

- In the imbalanced data classification, minority samples are far less than majority samples, which makes it difficult for minority to be effectively learned by classifiers.
- A synthetic minority oversampling technique (SMOTE) improves the sensitivity of classifiers to minority by synthesizing minority samples without repetition.
- Propose a synthetic minority oversampling technique based on Gaussian mixture model filtering (GMF-SMOTE).

Conclusion :

- The GMF-SMOTE performs better than the traditional oversampling algorithms on 20 UCI datasets.
- The population averages of sensitivity and specificity indexes of random forest (RF) on the UCI datasets synthesized by GMF-SMOTE are 97.49% and 97.02%, respectively.

Reference 2

"An Efficient Slime Mould Algorithm Combined With K-Nearest Neighbor for Medical Classification Tasks"

Objective :

- The integration of machine learning in computer-based diagnostic systems facilitates the early detection of diseases, enabling more productive treatments and prolonged survival rates.
- This paper proposes ISMA, an improved version of the slime mould algorithm (SMA) hybridized with the opposition-based learning (OBL) strategy based on the k-nearest neighbor (kNN) classifier for the classification approach.

Conclusion :

- Combined the Opposition-Based learning (OBL) and the slime mould algorithm (SMA) based on k-nearest neighbor (kNN) called ISMA–kNN for reducing the feature selection (FS) and classification purpose.
- On most of the data sets, the ISMA–kNN classification approach has been achieved the lowest number of feature selection with the highest classification accuracy within a reasonable period.

Reference 3

"A comprehensive survey on support vector machine classification: Applications, challenges and trends."

Objective :

- SVM algorithms have gained recognition in research and applications in several scientific and engineering areas.
- This paper provides a brief introduction of SVMs, describes many applications and summarizes challenges and trends.

Conclusion :

- The training of an SVM basically consists in solving a QP problem, this task is a high computational burden when the number of instances is large.
- When the data sets are very large or imbalanced, the accuracy of SVM is poor.

Reference 4

"Comparison of tree-based machine learning algorithms for predicting liquefaction potential using canonical correlation forest, rotation forest, and random forest based on CPT data."

Objective :

- This research investigates and compares the performance of three tree-based Machine Learning (ML) methods, Canonical Correlation Forest (CCF), Rotation Forest (RotFor), and Random Forest (RF).

Conclusion :

- The mean values of liquefied events for Dataset [A] and [B] are 0.5885 (133/226 types) and 0.7154 (181/253 types), respectively.
- The RotFor method achieved better prediction results than CCF and RF algorithms considering Dataset [B].

Dataset	Appr. Train %	Yes/No	Train, %	Yes/No	Test, %	*OA*	*Kappa*	*P*	*R*	*F*
[A]	**40%**	**46/46**	50	46/46	50	**0.8913**	**0.7826**	**0.9736**	**0.8043**	**0.8809**
[B]	29%	36/36		36/36		0.7917	0.5833	0.7692	0.8333	0.8000
[A]	**49%**	**55/55**	60	37/37	40	**0.9054**	**0.8108**	**0.8750**	**0.9459**	**0.9091**
[B]	34%	43/43		29/29		0.7931	0.5862	0.8148	0.7586	0.7857
[A]	**58%**	**65/65**	70	28/28	30	**0.9107**	**0.8214**	**0.8966**	**0.9286**	**0.9123**
[B]	40%	50/50		22/22		0.9091	0.8181	0.9474	0.9545	0.9130
Train	*Test*									
[A]	[B]	100% 181/72 [B]		100% 133/93 [A]		0.8221	0.5614	0.8736	0.8784	0.8760
[B]	[A]					0.8097	0.5836	0.7678	0.9699	0.8571

03

Dataset/Preprocessing

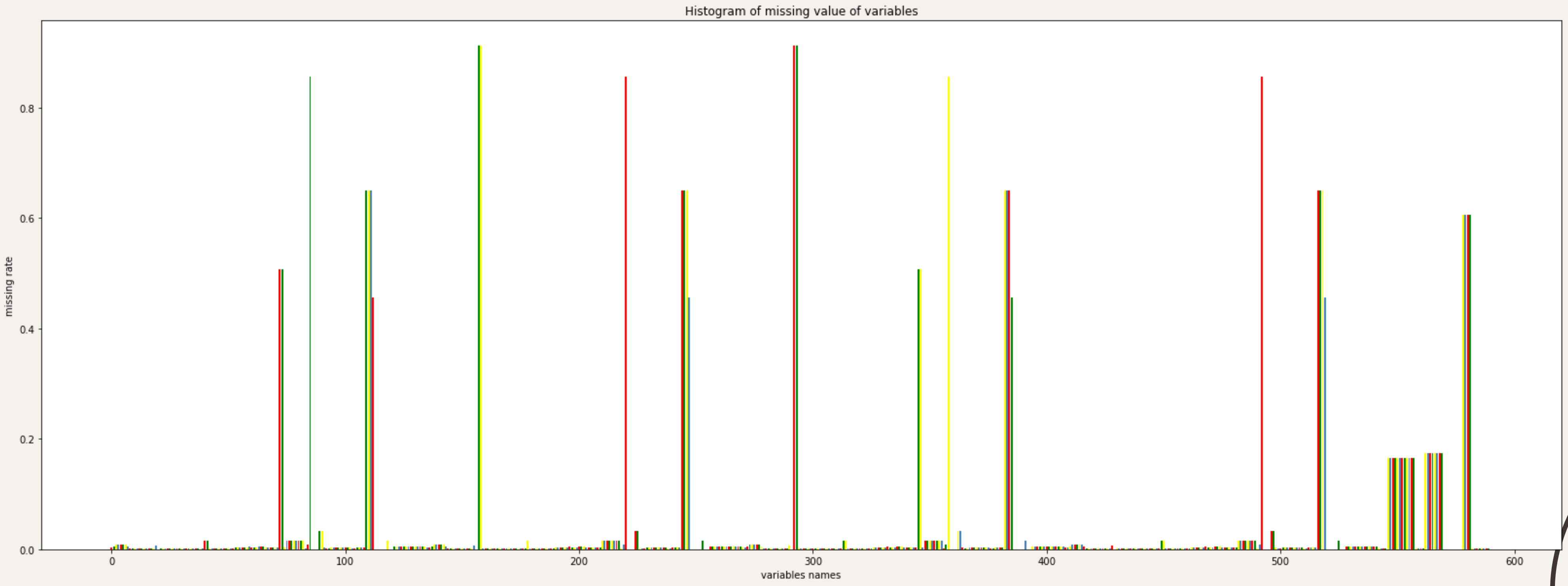
Original Dataset

- SECOM Data Set

	Time	0	1	2	3	4	5	6	7	8	...	581	582	583	584	585	586	587	588	589	Pass/Fail
0	2008-07-19 11:55:00	3030.93	2564.00	2187.733	1411.127	1.360	100.0	97.613	0.124	1.500	...	NaN	0.500	0.012	0.004	2.363	NaN	NaN	NaN	NaN	-1
1	2008-07-19 12:32:00	3095.78	2465.14	2230.422	1463.661	0.829	100.0	102.343	0.125	1.497	...	208.204	0.502	0.022	0.005	4.445	0.010	0.020	0.006	208.204	-1
2	2008-07-19 13:17:00	2932.61	2559.94	2186.411	1698.017	1.510	100.0	95.488	0.124	1.444	...	82.860	0.496	0.016	0.004	3.175	0.058	0.048	0.015	82.860	1
3	2008-07-19 14:43:00	2988.72	2479.90	2199.033	909.793	1.320	100.0	104.237	0.122	1.488	...	73.843	0.499	0.010	0.003	2.054	0.020	0.015	0.004	73.843	-1
4	2008-07-19 15:22:00	3032.24	2502.87	2233.367	1326.520	1.533	100.0	100.397	0.123	1.503	...	NaN	0.480	0.477	0.104	99.303	0.020	0.015	0.004	73.843	-1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1562	2008-10-16 15:13:00	2899.41	2464.36	2179.733	3085.378	1.484	100.0	82.247	0.125	1.342	...	203.172	0.499	0.014	0.004	2.867	0.007	0.014	0.005	203.172	-1
1563	2008-10-16 20:49:00	3052.31	2522.55	2198.567	1124.659	0.876	100.0	98.469	0.120	1.433	...	NaN	0.497	0.013	0.004	2.624	0.007	0.014	0.005	203.172	-1
1564	2008-10-17 05:26:00	2978.81	2379.78	2206.300	1110.497	0.824	100.0	99.412	0.121	NaN	...	43.523	0.499	0.015	0.004	3.059	0.020	0.009	0.003	43.523	-1
1565	2008-10-17 06:01:00	2894.92	2532.01	2177.033	1183.729	1.573	100.0	98.798	0.121	1.462	...	93.494	0.500	0.018	0.004	3.566	0.026	0.025	0.007	93.494	-1
1566	2008-10-17 06:07:00	2944.92	2450.76	2195.444	2914.179	1.598	100.0	85.101	0.123	NaN	...	137.784	0.499	0.018	0.004	3.627	0.012	0.016	0.004	137.784	-1

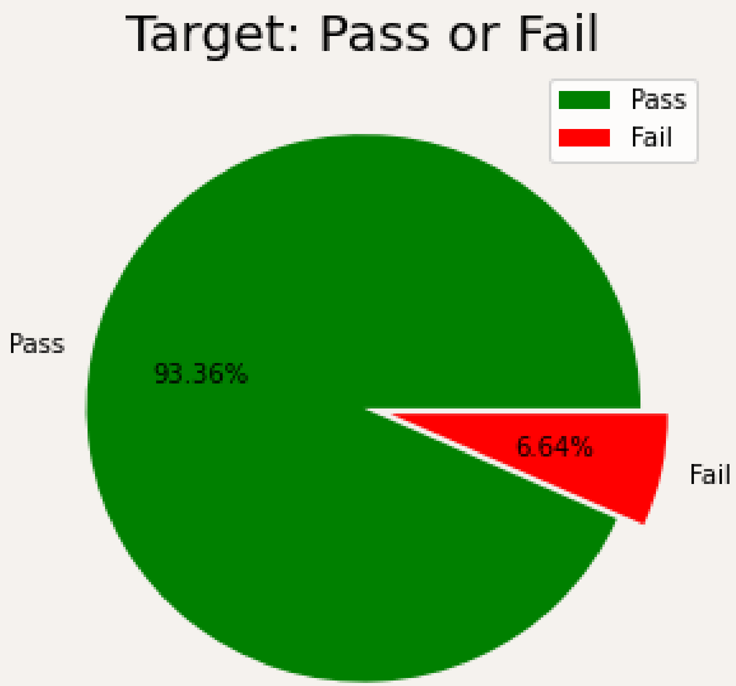
1567 rows × 592 columns

Check missing value



- Remove the feature with more than 90% of null values
- Remove 4 feature

Label Pie Chart

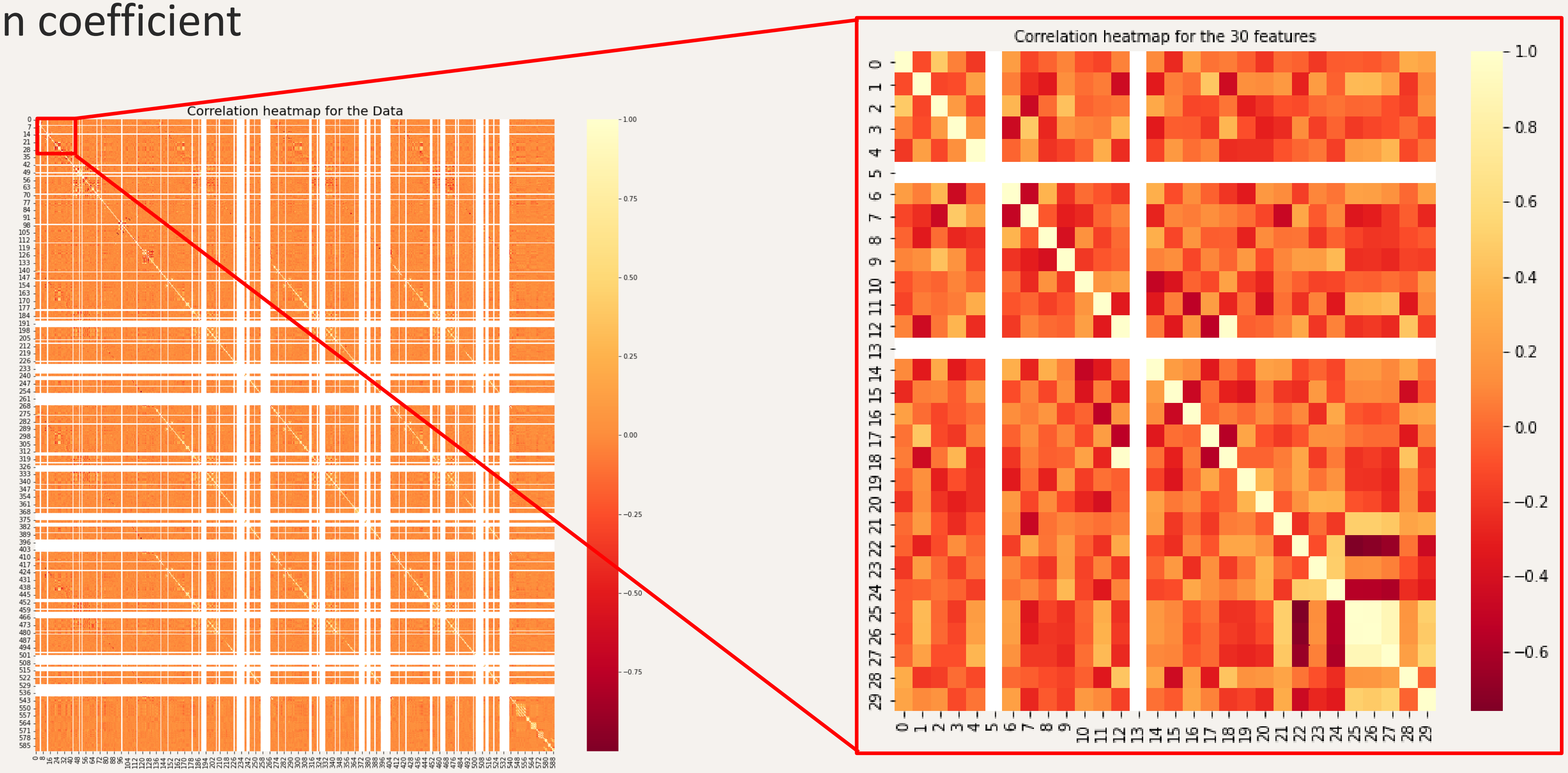


Classification

- -1: 1463
- 1: 104

The Correlation between Feature

- Using Pearson correlation coefficient

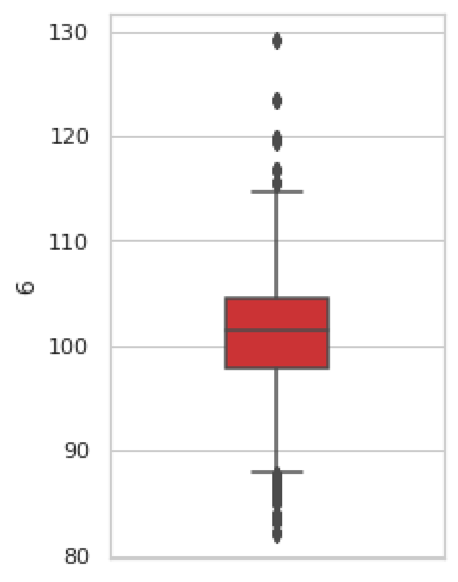


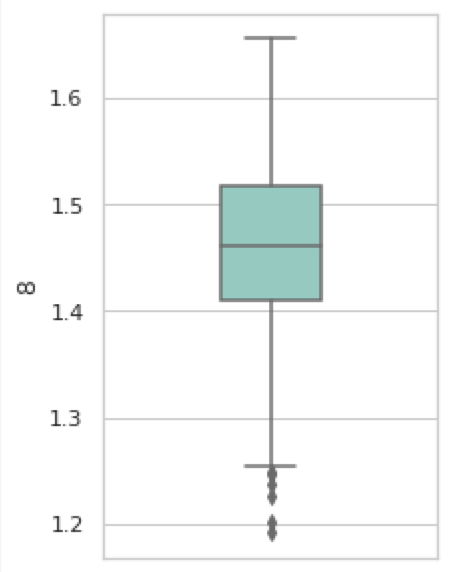
Imputation

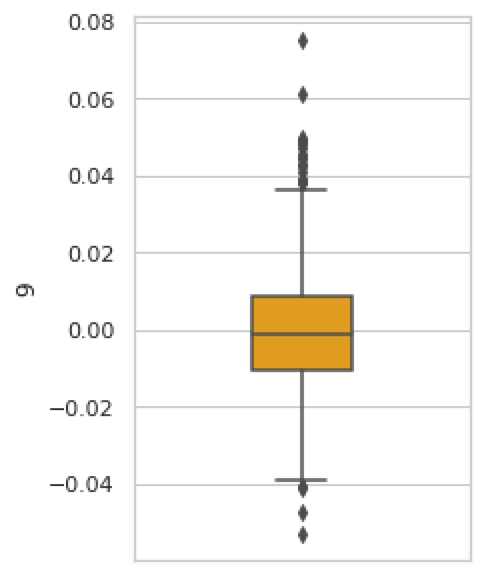
- KNN imputation
- neighbors=3

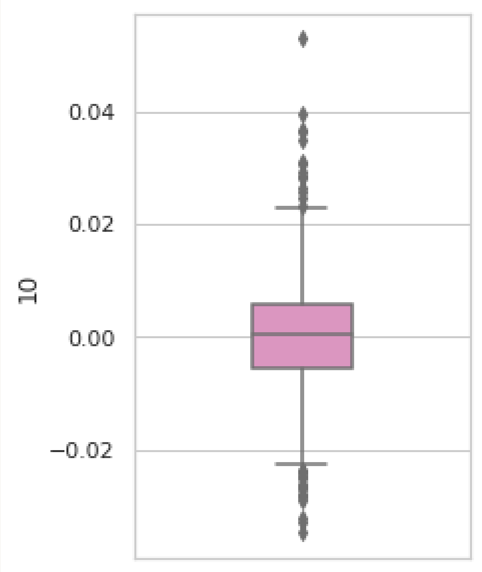
Addressing Outliers

- If the value is greater than **Q3+1.5*IQR** or less than **Q1-1.5*IQR**, it is considered an outlier.
- Outliers are identified and replaced by **median** value of the corresponding feature.









Check Multi-collinearity problem

- Checking for **correlated independent features** using correlation matrix. The threshold is selected as 0.80.
- If two features are correlated by coefficient>0.9, one of the correlated feature is removed.
- Number of features removed = 328

	feature1	feature2	correlation
0	11	147	0.903
1	12	282	0.905
2	17	420	0.907
3	18	18	0.981
4	21	153	0.892
...	...	...	...
323	583	584	0.831
324	584	585	0.996
325	585	583	0.831
326	587	588	0.852
327	588	587	0.852

328 rows × 3 columns

Check Multi-collinearity problem

- Checking for **Variance Inflation Factor (VIF)** of each feature. Features with VIF>5 are removed.
- The **Variance Inflation Factor (VIF)** is a numerical value that represents the degree of collinearity between observations of an independent variable.
- A VIF greater than 5 is considered multi-collinearity.
- Number of features with VIF > 5 = 3

	features	VIF
0	0	1.151
1	1	1.133
2	2	2.199
3	3	2.541
4	4	1.319
...	...	...
250	578	1.536
251	581	1.468
252	582	1.216
253	586	1.453
254	589	1.452

255 rows × 2 columns

Feature selection

- Features with very **low variance** will not have predictive power. Thus, features with very low variance are detected and dropped.
- Variance Threshold is calculated as (0.8*(1-0.8)).
- Number of features removed: 189

	Name	Var
0	0	3838.65
1	1	3550.195
2	2	661.589
3	3	112323.602
4	4	0.111
...	...	...
250	578	0.0
251	581	1354.308
252	582	0.0
253	586	0.0
254	589	2057.881

255 rows × 2 columns

Feature selection

- Using XGBoost to further select the best features, features with feature importance smaller than 0.01 are detected and dropped
- Number of features removed : 15

	Name	FI
18	59	0.029
57	500	0.029
56	499	0.027
15	41	0.027
31	129	0.025
...	...	...
19	63	0.006
50	484	0.005
53	487	0.004
63	570	0.002
48	482	0.001

66 rows × 2 columns

Normalization

- Bring all values into the range [0,1]

$$\frac{X - X_{min}}{X_{max} - X_{min}}$$

Final Dataset

	0	1	2	14	16	22	23	28	32	39	...	486	488	489	499	500	510	547	548	562	581
0	0.555	0.699	0.425	0.404	0.586	0.700	0.457	0.234	0.050	0.207	...	0.844	0.053	0.000	0.000	0.000	0.601	0.180	0.372	0.293	0.416
1	0.735	0.408	0.723	0.560	0.234	0.395	0.593	0.469	0.401	0.378	...	0.131	0.195	0.000	0.000	0.000	0.438	0.672	0.284	0.556	0.390
2	0.282	0.687	0.416	0.515	0.258	0.490	0.456	0.396	0.366	0.518	...	0.747	0.192	0.328	0.000	0.000	0.438	0.759	0.285	0.692	0.430
3	0.438	0.452	0.504	0.521	0.428	0.438	0.339	0.161	0.402	0.074	...	0.105	0.000	0.442	0.000	0.712	0.438	0.512	0.111	0.749	0.383
4	0.559	0.519	0.743	0.589	0.719	0.425	0.471	0.155	0.732	0.457	...	0.000	0.750	0.000	0.293	0.000	0.438	0.341	0.640	0.205	0.818
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1562	0.190	0.406	0.370	0.674	0.675	0.399	0.513	0.621	0.153	0.457	...	0.183	0.318	0.249	0.000	0.000	0.495	0.410	0.483	0.556	0.390
1563	0.615	0.577	0.501	0.490	0.670	0.076	0.549	0.717	0.205	0.982	...	0.000	0.273	0.385	0.816	0.875	0.274	0.375	0.175	0.681	0.359
1564	0.410	0.157	0.555	0.599	0.440	0.495	0.442	0.628	0.624	0.313	...	0.171	0.382	0.138	0.457	0.000	0.510	0.011	0.246	0.187	0.226
1565	0.177	0.605	0.351	0.530	0.505	0.466	0.467	0.586	0.320	0.607	...	0.131	0.148	0.160	0.511	0.434	0.730	0.375	0.175	0.556	0.485
1566	0.316	0.366	0.479	0.610	0.513	0.457	0.480	0.729	0.037	0.457	...	0.119	0.238	0.211	0.000	0.000	0.706	0.375	0.175	0.248	0.714

1567 rows × 51 columns

04

Model Compare

Model

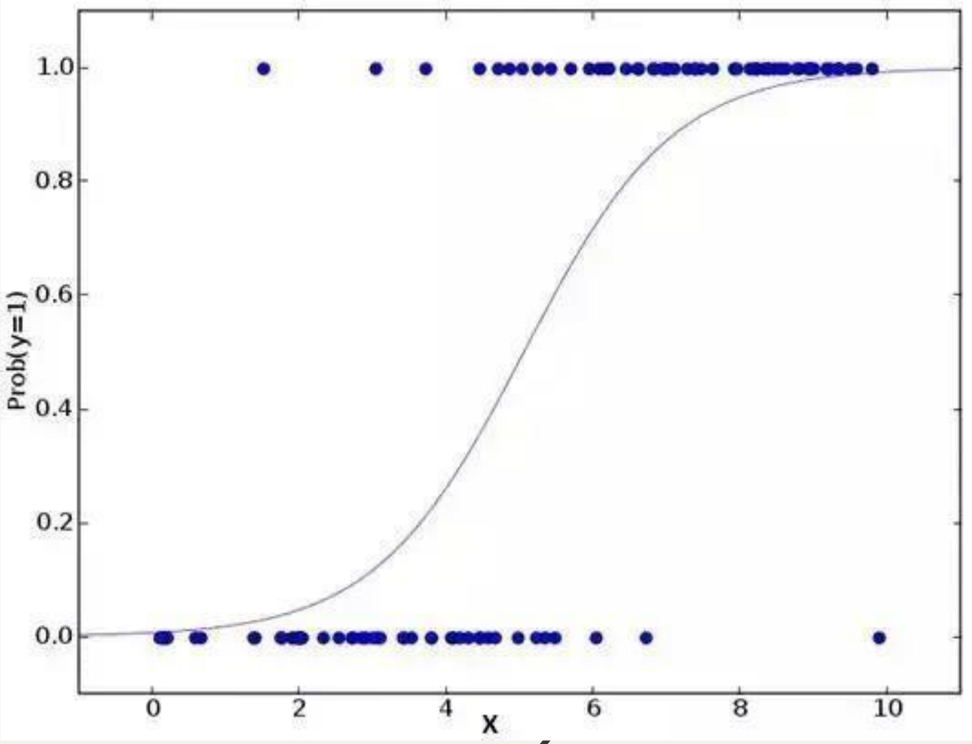
- Logistic regression
- Random Forest
- K Nearest Neighbor (KNN)
- Supper Vector Machine(SVM)
- Rotation Forest

Logistic regression

- Dependent variable is binary (success/ failure or pass fail)
- sigmoid function (logistic function)

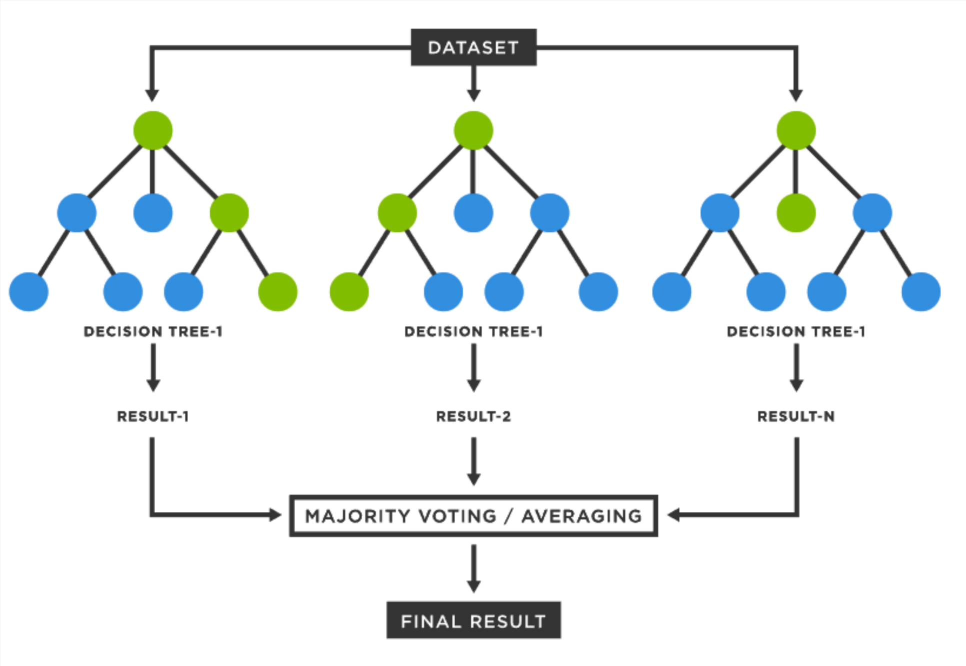
$$P_i = 1 - \left(\frac{1}{1+e_i^z}\right)$$

$$Z_i = \log\left(\frac{P_i}{1-P_i}\right) = \beta_0 + \beta_1 * x_1 + \ldots + \beta_n * x_n$$



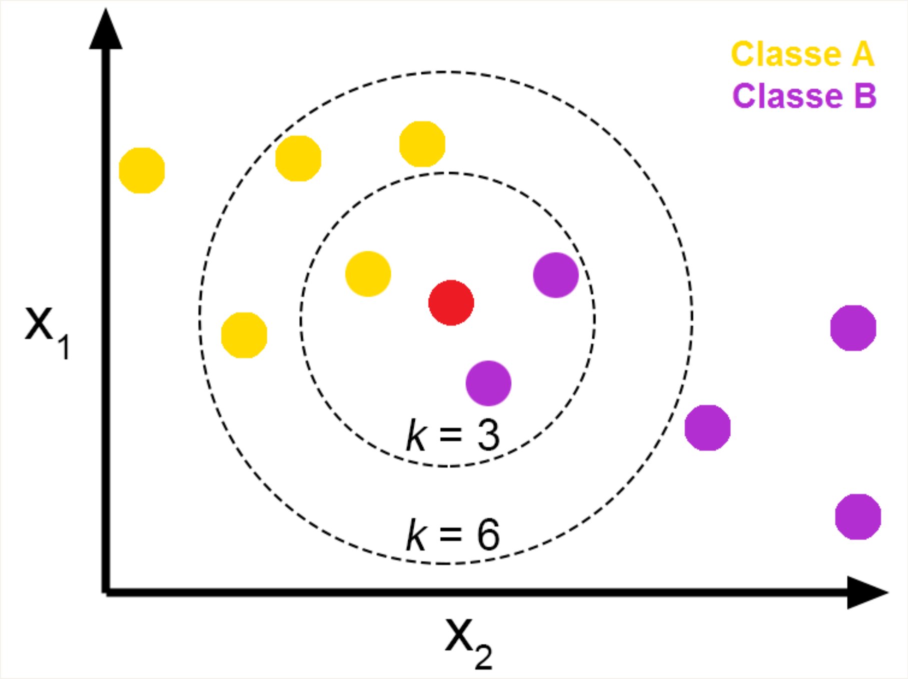
RF (Random Forest)

- Consists of a large number of individual decision trees
- Each individual tree spits out a class prediction
- The class with the most votes becomes the model’s prediction



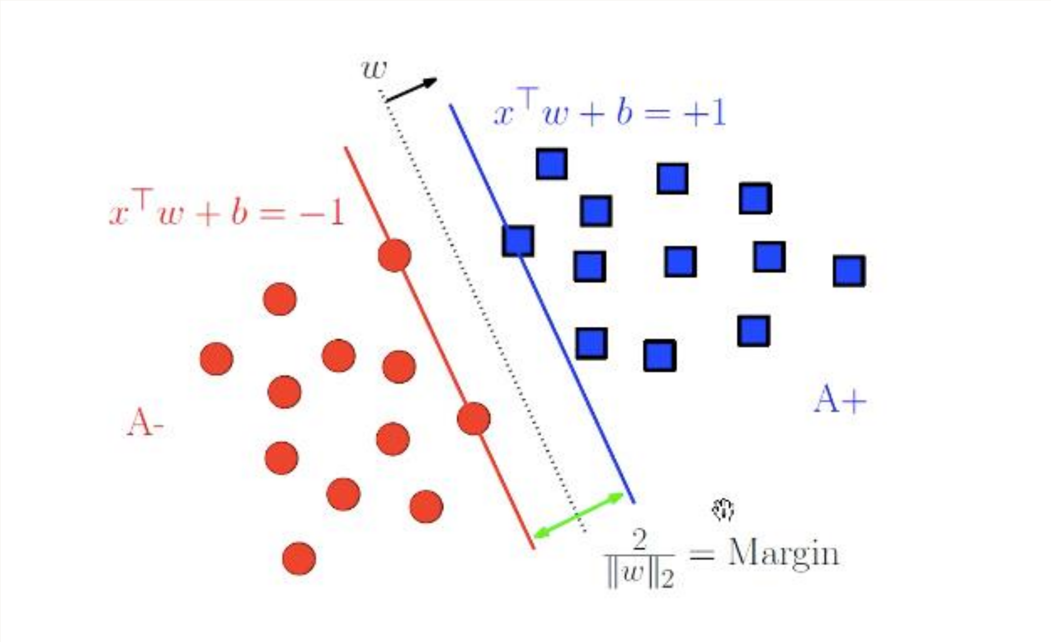
KNN (K Nearest Neighbor)

- According to the distance between each other to classify the data
- Whichever category is closest to it will be classified into that category



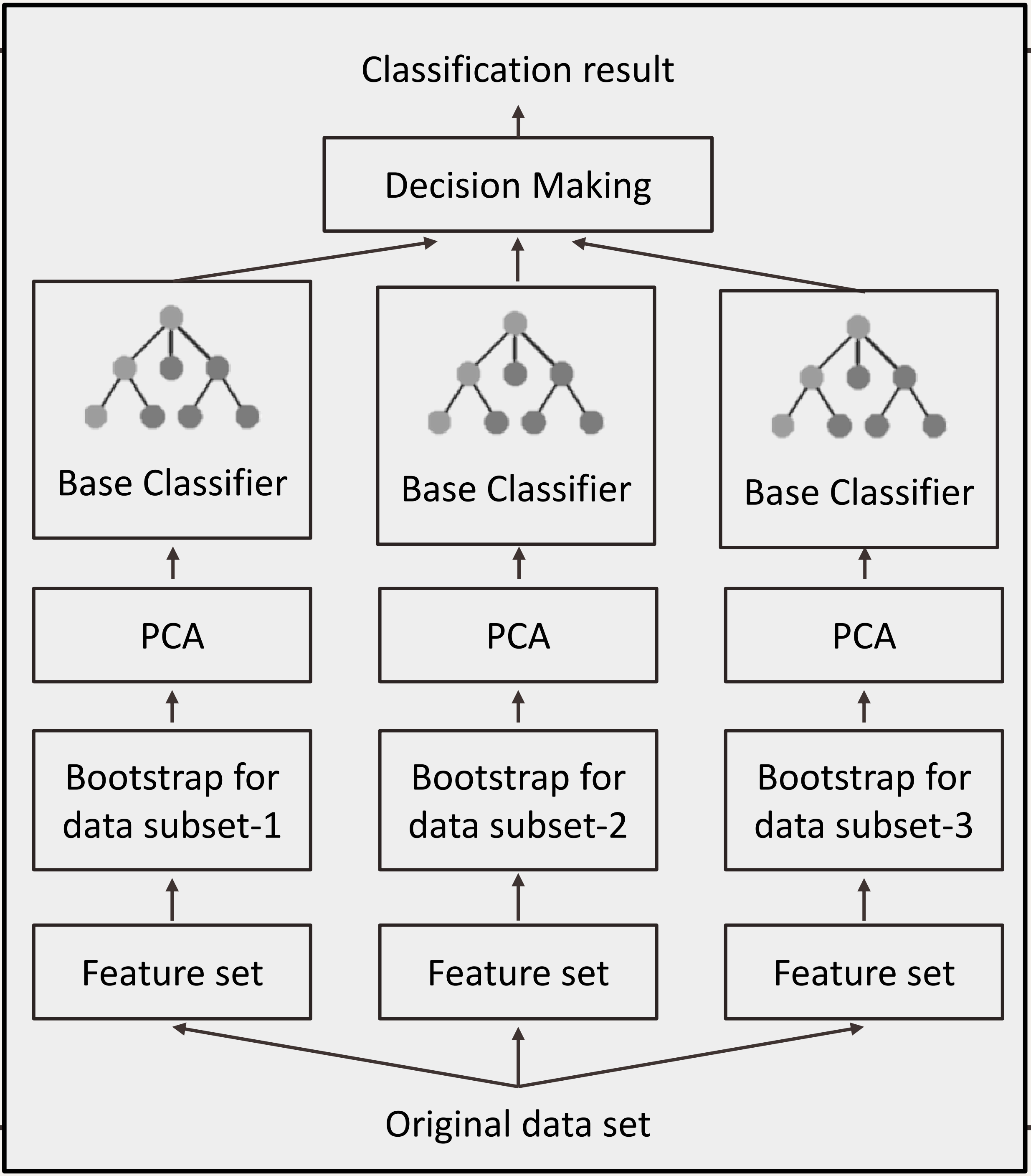
SVM (Support Vector Machine)

- Find a Hyperplane to effectively cut the samples
- The samples on both sides of the Hyperplane should be far away from the Hyperplane.



RotF (Rotation Forest)

- Split feature set into K subsets
- Use splited feature set to bootstrap data subset(K -subset)
- Run Principal Component Analysis on each subset separately
- Use the new feature set to construct a decision tree
- Use majority vote to determine final classification.



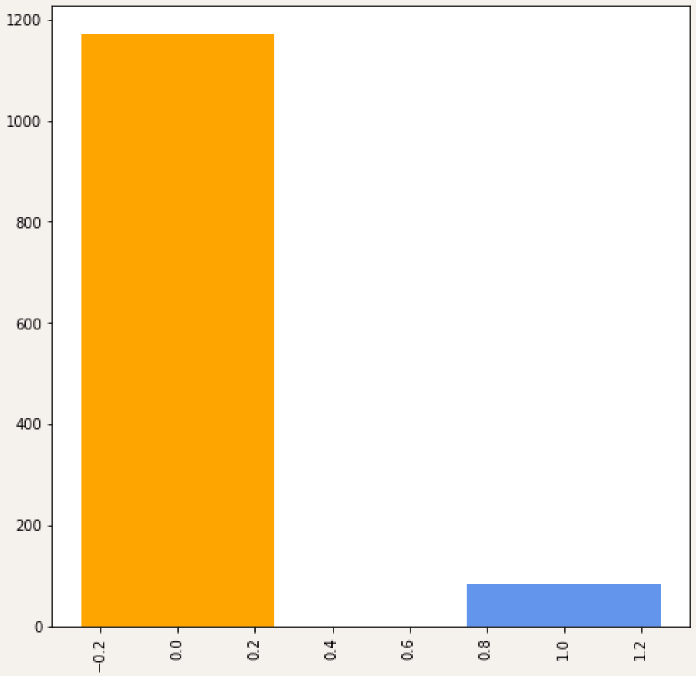
05

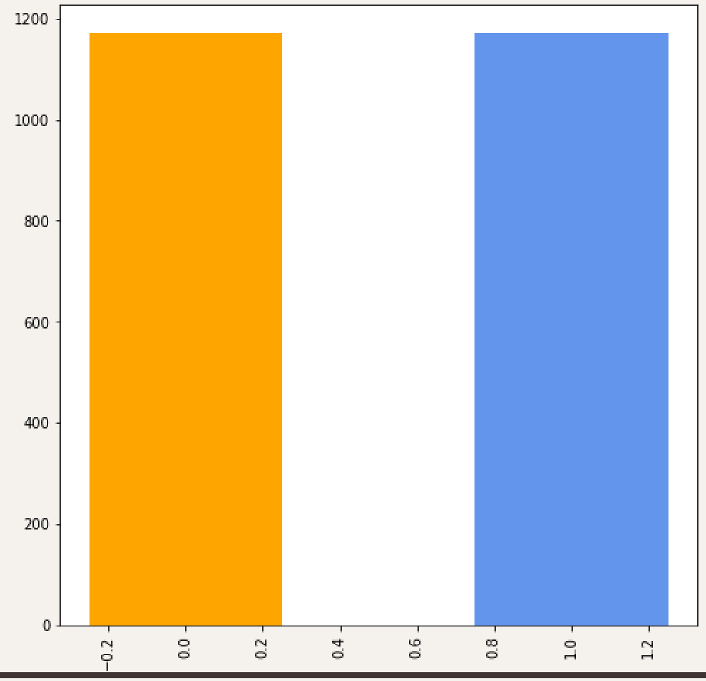
Methodology

SMOTE

The algorithm steps :

- 1.Sampling the nearest neighbor algorithm, calculates the K nearest neighbors of each minority sample.
- 2.Randomly select N samples from K nearest neighbors for random linear interpolation.
- 3.Construct a new minority sample.
- 4.Synthesize the new sample with the original data to generate a new training set.





Split training data and testing data

- Using a train-test split of 80%-20%.
- The split is stratified to maintain the same dependent class distribution for train and test data.

Oversampling

- Because the data is highly imbalance , we need to do data oversampling.
- Using Synthesized Minority Oversampling Technique(SMOTE) to up sampling the minority class data of training data

```
before:
0    1170
1      83
Name: Pass/Fail, dtype: int64
After Oversampling

0    1170
1    1170
Name: label, dtype: int64
```

Logistic Regression

C: [0.0001,0.001,0.1, 1, 100, 1000,10000]

Max iteration : 1, 10, 100, 500,1000

Class weight: balanced, None

solver: liblinear, sag, lbfgs, newton-cg

Final parameter:

C=1, max iteration =500, class weight=none ,

solver=newton-cg

Training Data

```
AUC :  0.8102564102564103
confusion_matrix
[[905 265]
 [179 991]]
              precision    recall  f1-score   support

           0       0.83      0.77      0.80      1170
           1       0.79      0.85      0.82      1170

    accuracy                           0.81      2340
   macro avg       0.81      0.81      0.81      2340
weighted avg       0.81      0.81      0.81      2340
```

Testing Data

```
AUC :  0.6271737363887535
confusion_matrix
[[228  65]
 [ 11  10]]
              precision    recall  f1-score   support

           0       0.95      0.78      0.86       293
           1       0.13      0.48      0.21        21

    accuracy                           0.76       314
   macro avg       0.54      0.63      0.53       314
weighted avg       0.90      0.76      0.81       314
```

Random Forest

N estimators : 500,700

Max features: log2,sqrt,auto

Max depth:20,30,40,50

Min samples leaf:5,10,20,30,50

Final parameter:

N estimators=500, max features= log2 , max depth=30,min samples leaf=5,

Training Data

```
AUC :  0.9918803418803419
confusion_matrix
[[1170    0]
 [  19 1151]]
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      1170
           1       1.00      0.98      0.99      1170

    accuracy                           0.99      2340
   macro avg       0.99      0.99      0.99      2340
weighted avg       0.99      0.99      0.99      2340
```

Testing Data

```
AUC :  0.5424995936941329
confusion_matrix
[[290   3]
 [ 19   2]]
              precision    recall  f1-score   support

           0       0.94      0.99      0.96       293
           1       0.40      0.10      0.15        21

    accuracy                           0.93       314
   macro avg       0.67      0.54      0.56       314
weighted avg       0.90      0.93      0.91       314
```

KNN

N neighbors : 1,2,⋯,50

Algorithm : ball tree ,kd tree ,brute

Leaf size : 5,10,15,20,30

Final parameter:

N neighbors = 2, Algorithm =ball tree ,leaf size = 5

Training Data

AUC : 0.9952991452991453
confusion_matrix
[[1170   0]
 [11 1159]]

	precision	recall	f1-score	support
0	0.99	1.00	1.00	1170
1	1.00	0.99	1.00	1170
accuracy			1.00	2340
macro avg	1.00	1.00	1.00	2340
weighted avg	1.00	1.00	1.00	2340

Testing Data

AUC : 0.6305054444986185
confusion_matrix
[[216 77]
 [10 11]]

	precision	recall	f1-score	support
0	0.96	0.74	0.83	293
1	0.12	0.52	0.20	21
accuracy			0.72	314
macro avg	0.54	0.63	0.52	314
weighted avg	0.90	0.72	0.79	314

Support Vector Machine

C:0.01,0.1, 1, 10, 100

class weight: balanced, None

kernel: linear , rbf ,sigmoid ,poly

gamma: auto ,scale

Final parameter:

C=10 , class weight=balanced, kernel=linear ,

gamma=scale

Training Data

```
AUC :  0.8170940170940171
confusion_matrix
[[ 874  296]
 [ 132 1038]]
              precision    recall  f1-score   support

           0       0.87      0.75      0.80      1170
           1       0.78      0.89      0.83      1170

    accuracy                           0.82      2340
   macro avg       0.82      0.82      0.82      2340
weighted avg       0.82      0.82      0.82      2340
```

Testing Data

```
AUC :  0.6611409068746953
confusion_matrix
[[220  73]
 [  9  12]]
              precision    recall  f1-score   support

           0       0.96      0.75      0.84       293
           1       0.14      0.57      0.23        21

    accuracy                           0.74       314
   macro avg       0.55      0.66      0.53       314
weighted avg       0.91      0.74      0.80       314
```

Rotation Forest

N estimators : 500,700

Max features: log2,sqrt,auto

Max depth: 20,30,40,50

Min samples leaf: 5,10,20,30,50

Final parameter:

n estimators=700 , max features= auto ,max depth=50,min samples leaf=50

Training Data

```
AUC :  0.9662393162393162
confusion_matrix
[[1153   17]
 [  62 1108]]
              precision    recall  f1-score   support

           0       0.95      0.99      0.97      1170
           1       0.98      0.95      0.97      1170

    accuracy                           0.97      2340
   macro avg       0.97      0.97      0.97      2340
weighted avg       0.97      0.97      0.97      2340
```

Testing Data

```
AUC :  0.5730537948967983
confusion_matrix
[[280  13]
 [ 17   4]]
              precision    recall  f1-score   support

           0       0.94      0.96      0.95       293
           1       0.24      0.19      0.21        21

    accuracy                           0.90       314
   macro avg       0.59      0.57      0.58       314
weighted avg       0.90      0.90      0.90       314
```

Evaluation Index

$$Sensitivity = TP/(TP + FN)$$

$$Specificity = TN/(FP + TN)$$

$$G - mean = \sqrt{Sensitivity + Specificity}$$

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TN + FN)(TN + FP)(TP + FN)(TP + FP)}}$$

		True Condition	
	Total Population (T)	**Positive**	**Negative**
Predicted outcome	**Positive**	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

Compare the result of the methodology

Stratified 5-Fold cross validation

Testing Data(mean)	Accuracy	Sensitivity	Specificity	G-mean	MCC	AUC
Logistic Regression	0.736	0.948	0.108	0.316	0.099	0.587
Random Forest	0.918	0.935	0.142	0.277	0.046	0.514
KNN	0.693	0.942	0.087	0.285	0.055	0.550
Support Vector Machine	0.718	0.950	0.106	0.315	0.104	0.595
Rotation Forest	0.899	0.940	0.181	0.388	0.107	0.548

06
Conclusion/References

Conclusion

- Through the oversampling method(SMOTE), we can solve the problem of training data set is imbalance.
- Accuracy is not an appropriate metric when evaluating imbalanced datasets.
- Compared with other models, rotation forest is the most effective model to solve this imbalanced secom dataset.

References

- Xu, Z., Shen, D., Kou, Y., and Nie, T., 2022, “A Synthetic Minority Oversampling Technique Based on Gaussian Mixture Model Filtering for Imbalanced Data Classification,” IEEE Transactions on Neural Networks and Learning Systems, Early Access, 1-14.
- Wazery, Y. M., Saber, E., Houssein, E. H., Ali, A. A., and Amer, E., 2021, “An Efficient Slime Mould Algorithm Combined With K-Nearest Neighbor for Medical Classification Tasks,” IEEE Access, Vol. 9, 113666-113682.
- Cervantes, J., Garcia-Lamont, F., Rodríguez-Mazahua, L., and Lopez, A., 2020, “A comprehensive survey on support vector machine classification: Applications, challenges and trends,” Neurocomputing, Vol. 408, No. 30, 189-215.
- Demir, S., and Sahin, E. K., 2022, “Comparison of tree-based machine learning algorithms for predicting liquefaction potential using canonical correlation forest, rotation forest, and random forest based on CPT data,” Soil Dynamics and Earthquake Engineering, Vol. 154, 107-130.

Thanks