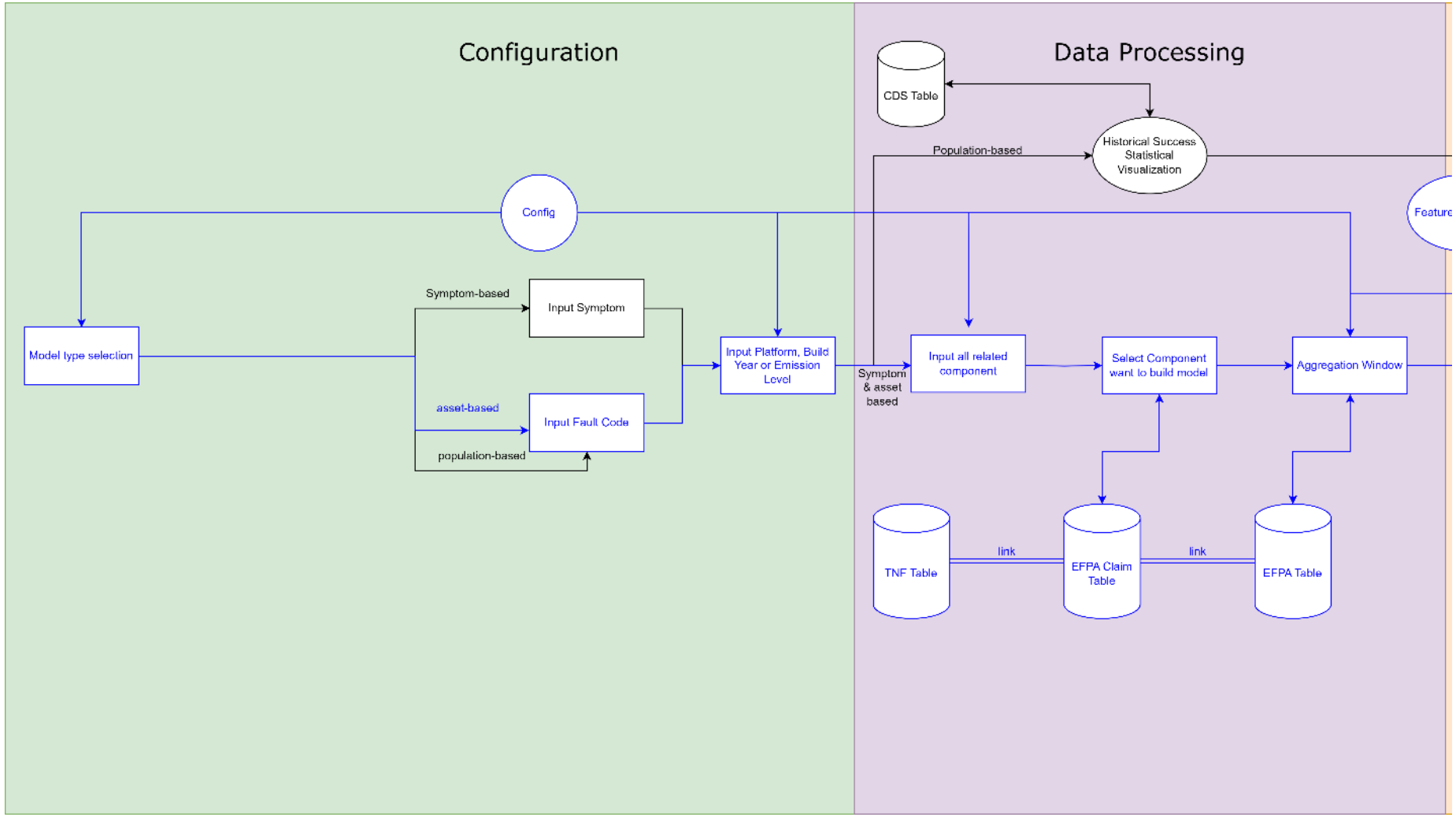
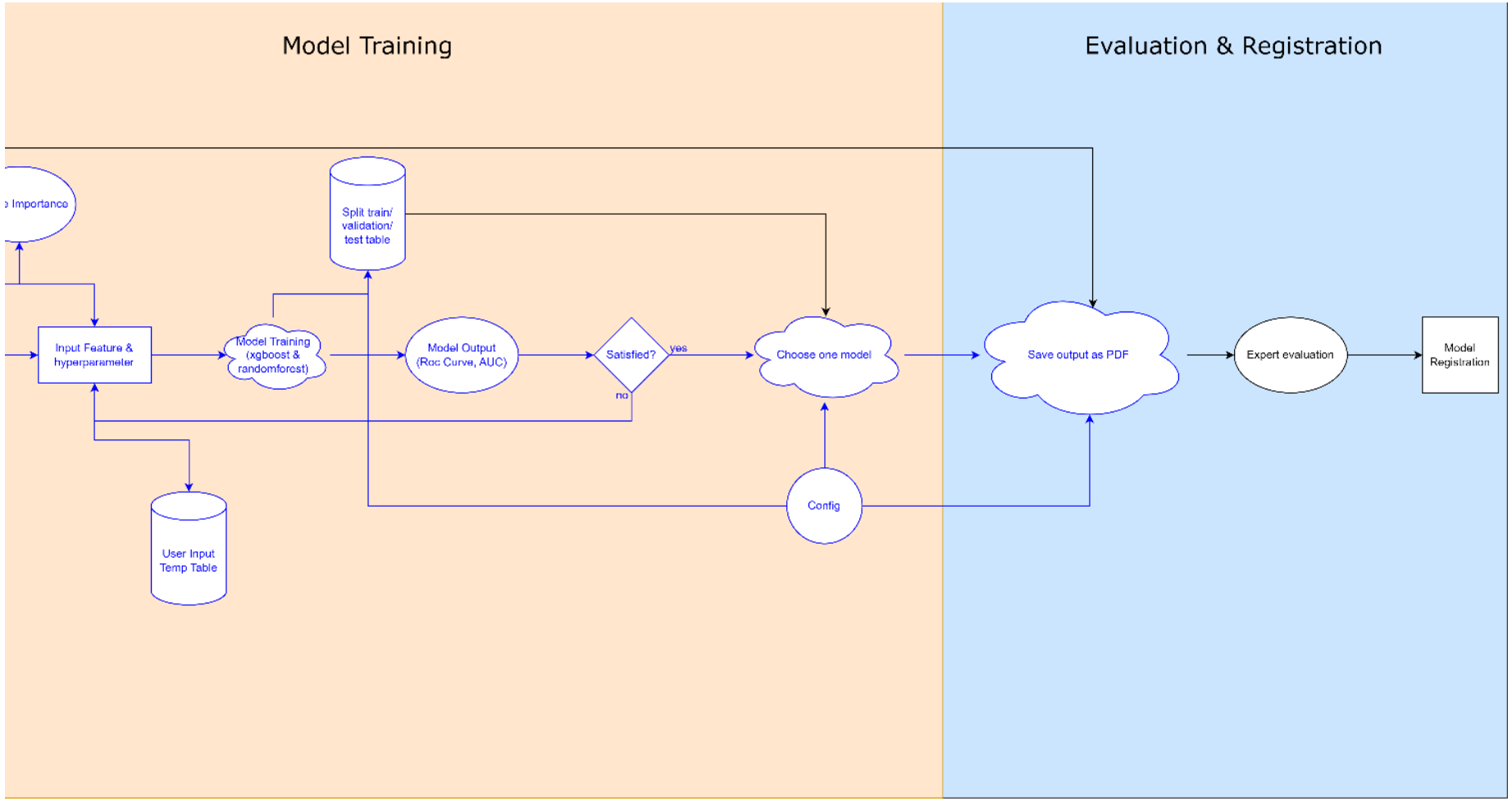
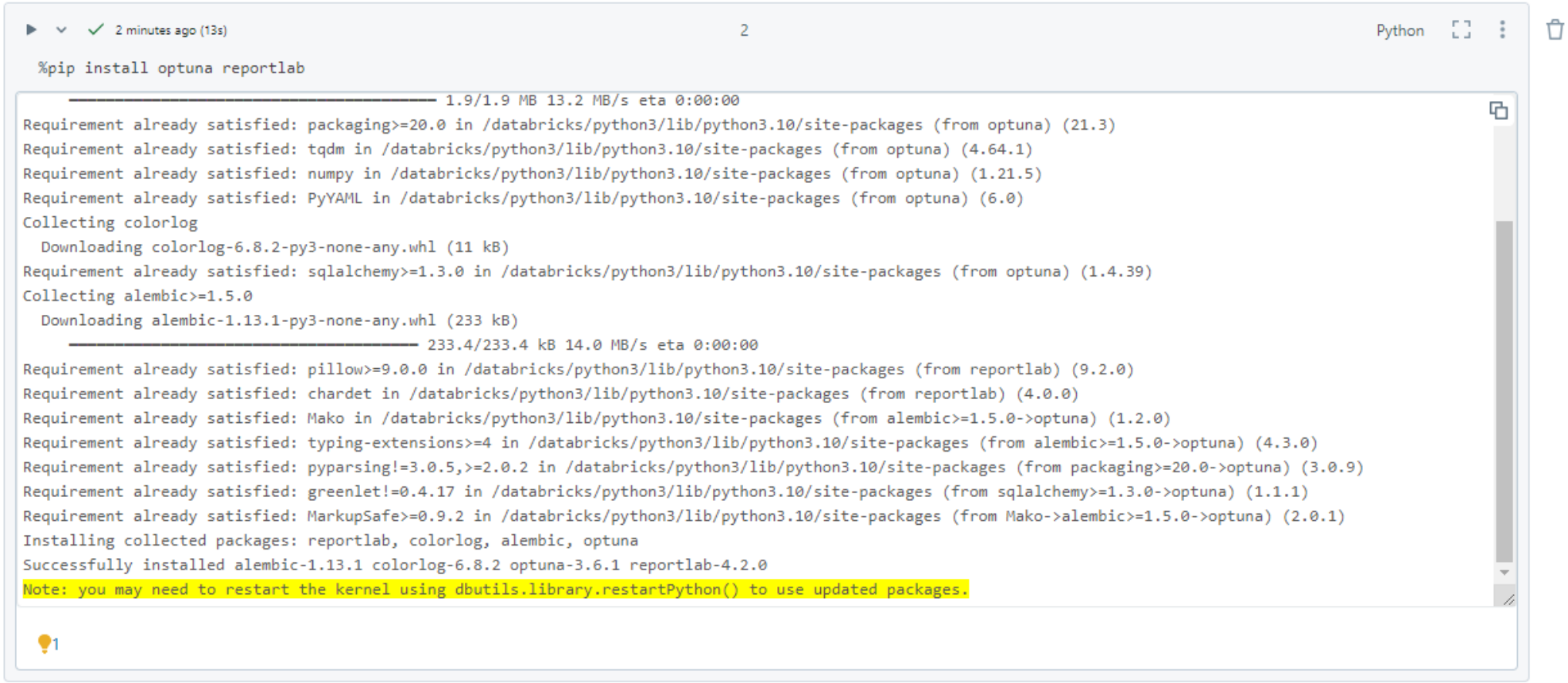
Workflow


Configuration
Data Processing
CDS Table
Historical Success Statistical Visualization
Population-based
Config
Feature
Symptom-based
Input Symptom
Model type selection
Input Platform, Build Year or Emission Level
Symptom & asset based
Input all related component
Select Component want to build model
Aggregation Window
asset-based
Input Fault Code
population-based
TNF Table
link
EFPA Claim Table
link
EFPA Table

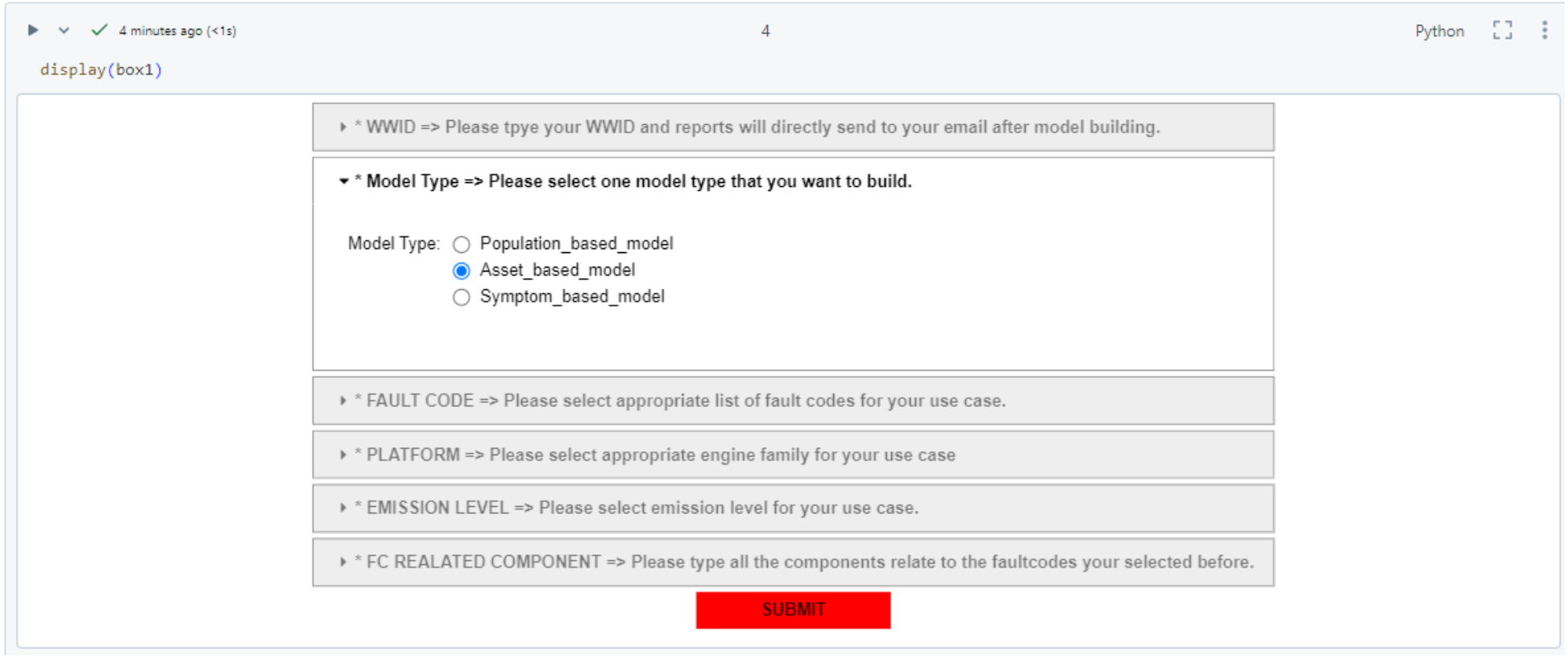
Workflow


Model Training
Evaluation & Registration
Importance
Split train/ validation/ test table
Input Feature & hyperparameter
Model Training (xgboost & randomforest)
Model Output (Roc Curve, AUC)
Satisfied?
yes
no
Choose one model
Save output as PDF
Expert evaluation
Model Registration
User Input Temp Table
Config

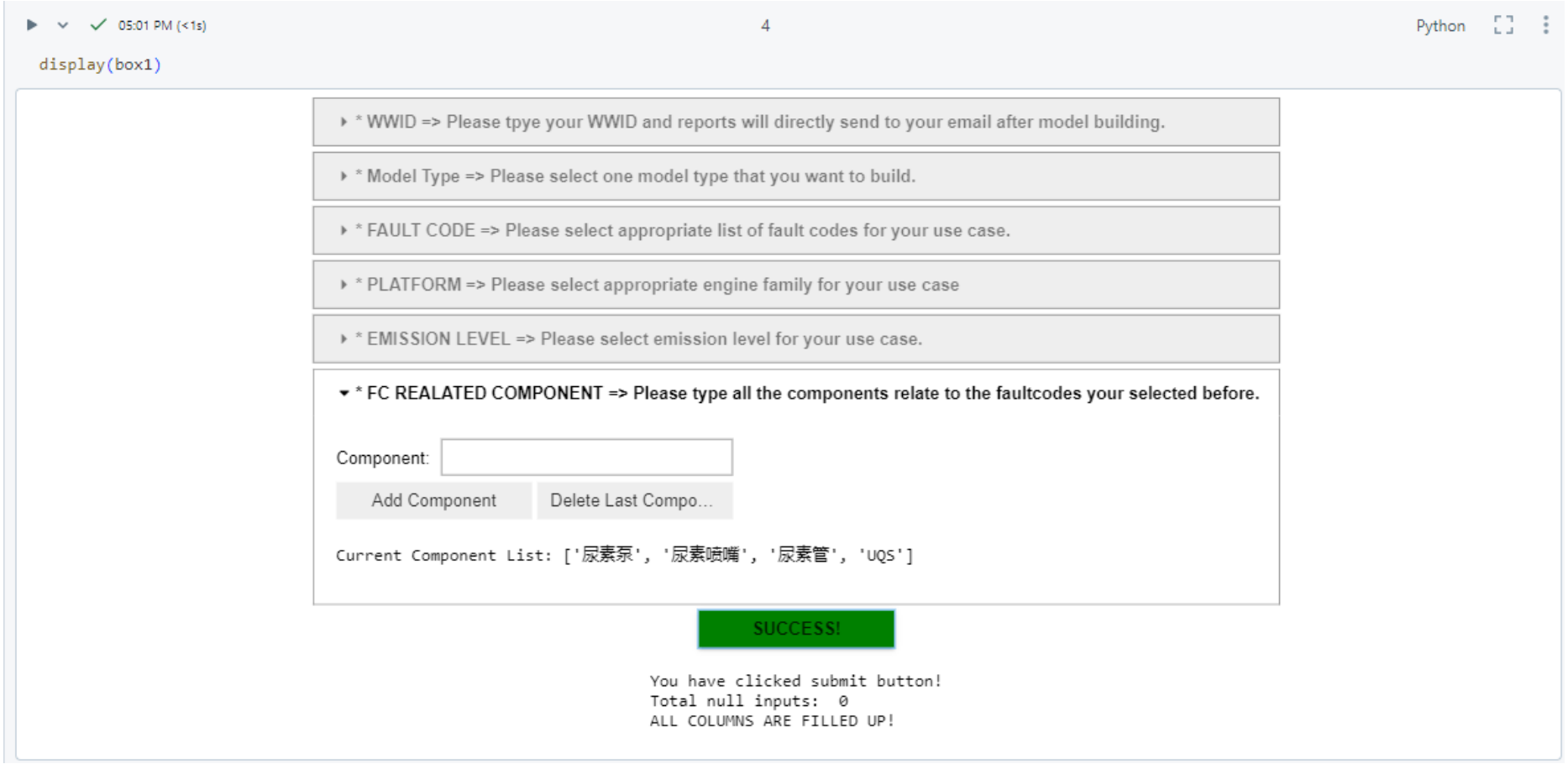
Install Required Packages



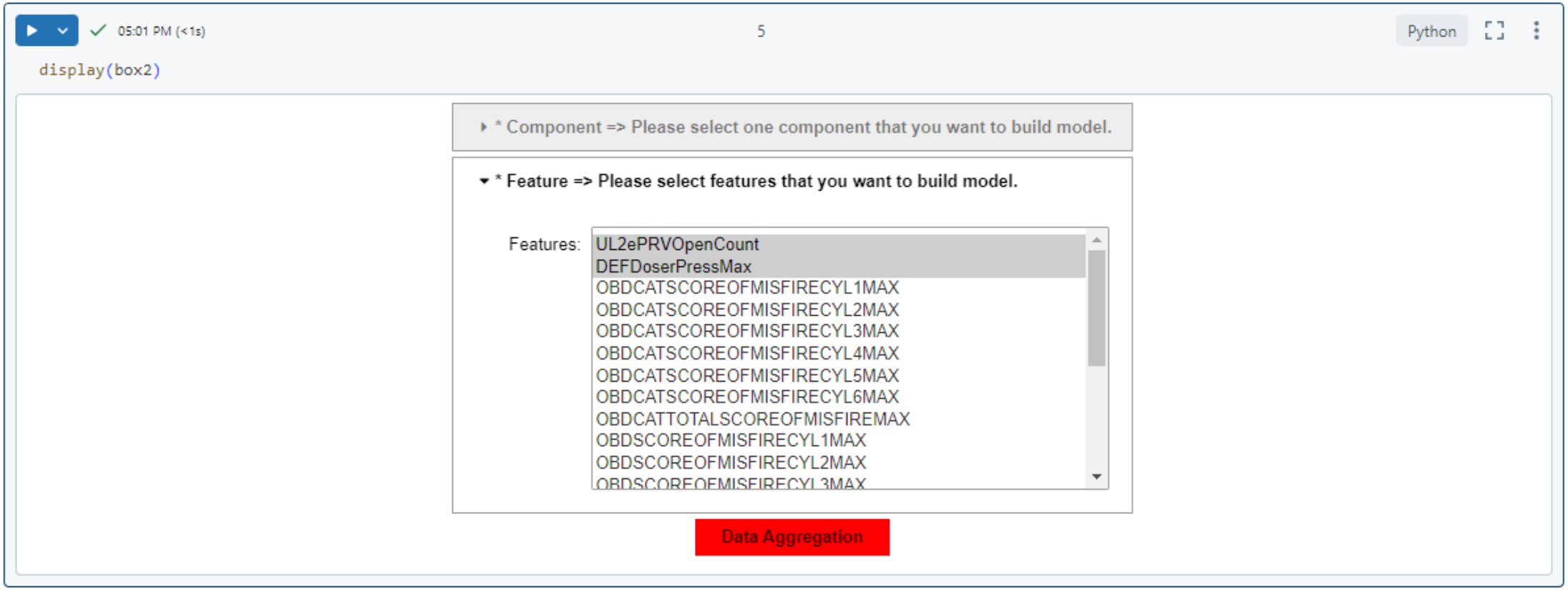
Select Model Type(for now ABOD only)



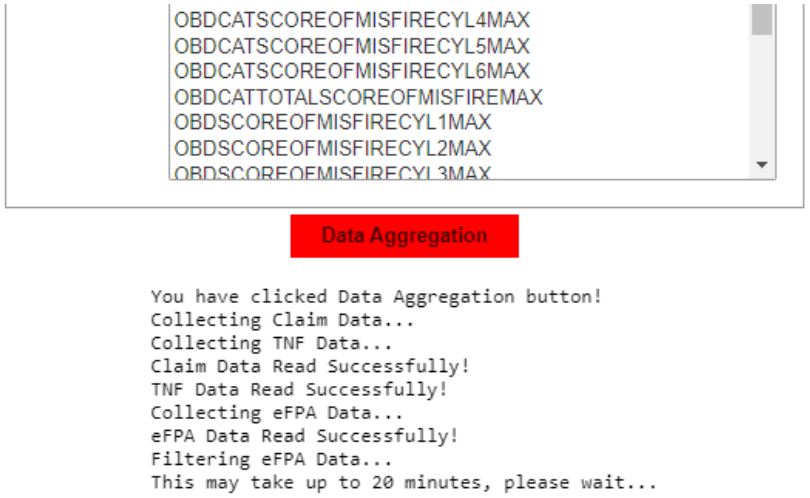
Type all fault code related components (one by one)

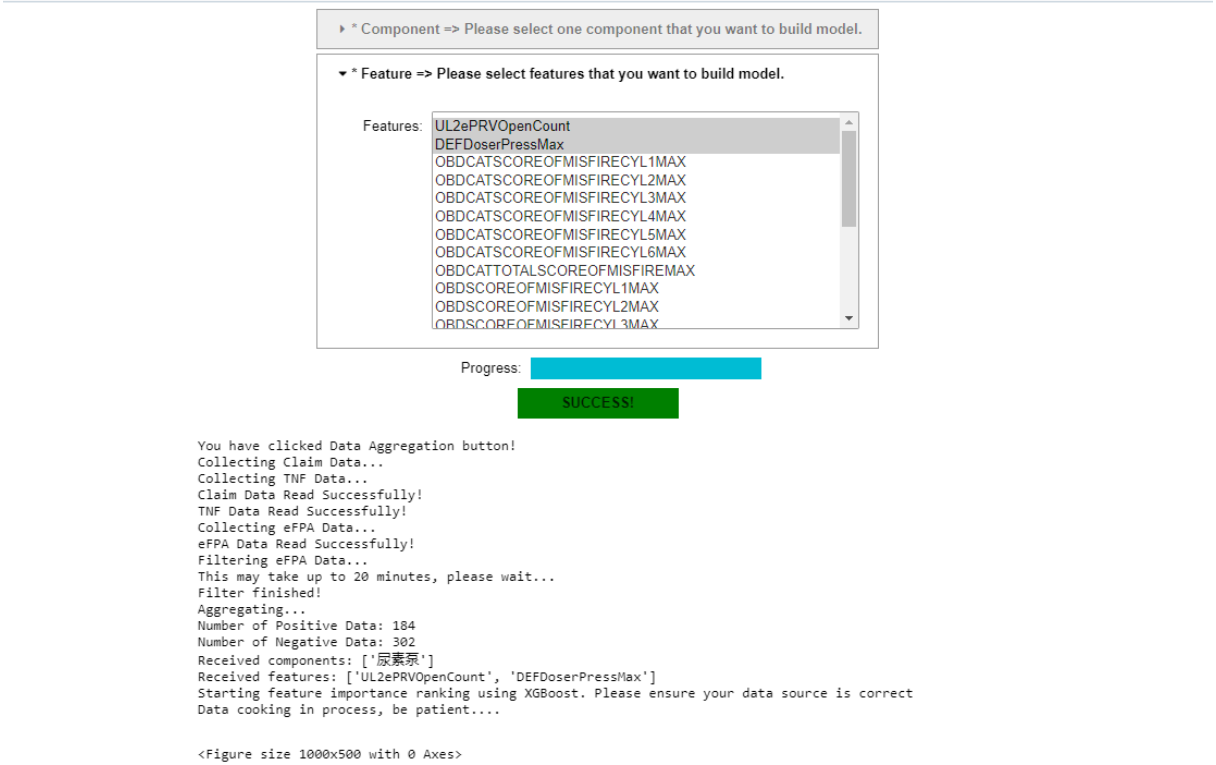


Select eFPA features to aggregation



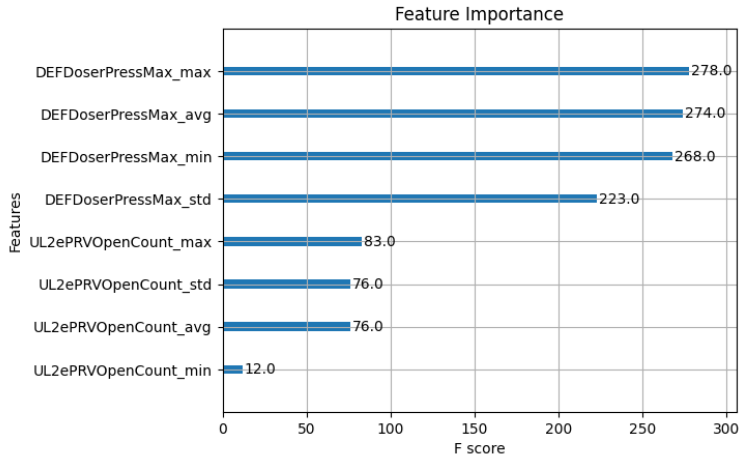
Aggregation may take up to 20 minutes



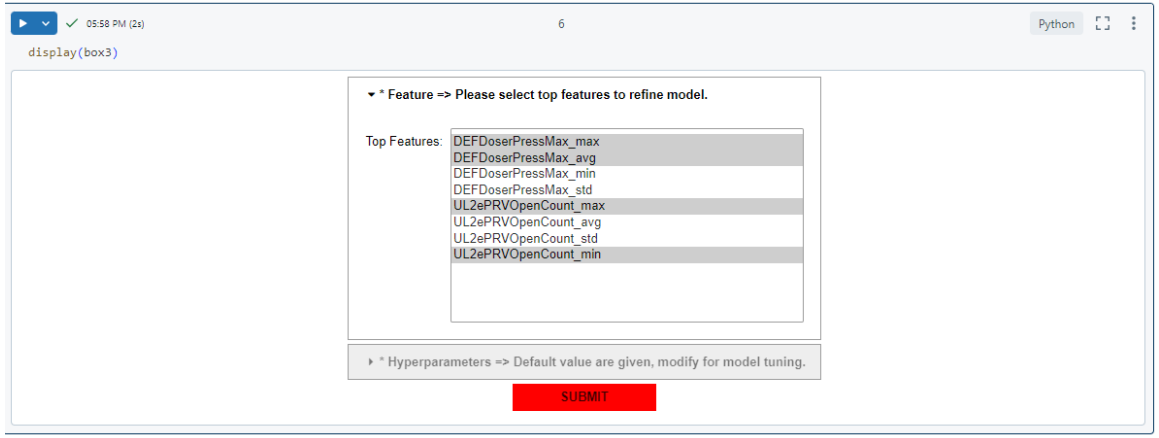


Top 10 parameter selected

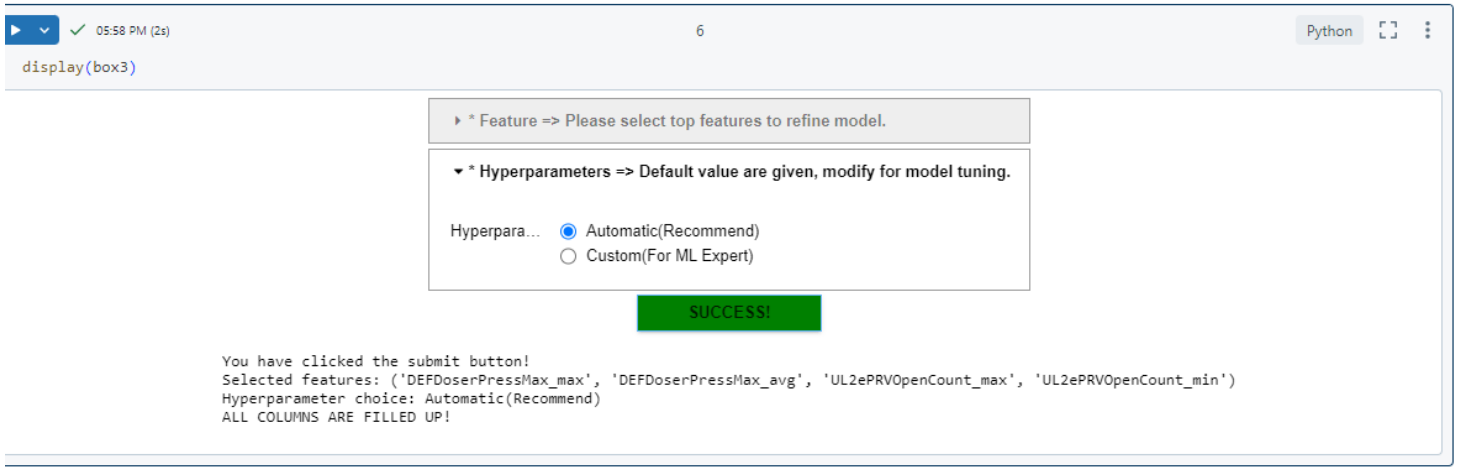
```
Filter finished!
Aggregating...
Number of Positive Data: 299
Number of Negative Data: 187
Received components: ['尿素喷嘴']
Received features: ['UL2ePRVOpenCount', 'DEFDoserPressMax']
Starting feature importance ranking using XGBoost. Please ensure your data source is correct
Data cooking in process, be patient....

<Figure size 1000x500 with 0 Axes>
```

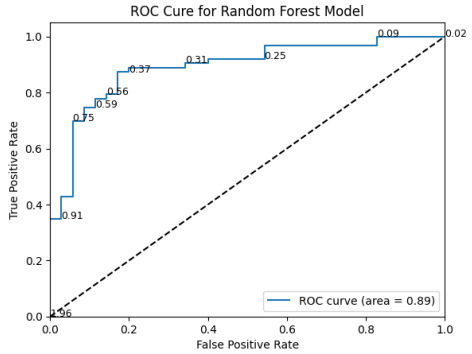


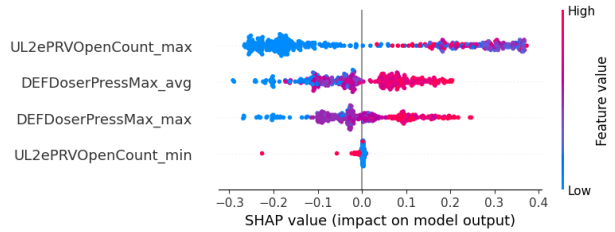

```
Top 10 feature importances:
DEFDoserPressMax_max
DEFDoserPressMax_avg
DEFDoserPressMax_min
DEFDoserPressMax_std
UL2ePRVOpenCount_max
UL2ePRVOpenCount_avg
UL2ePRVOpenCount_std
UL2ePRVOpenCount_min
```



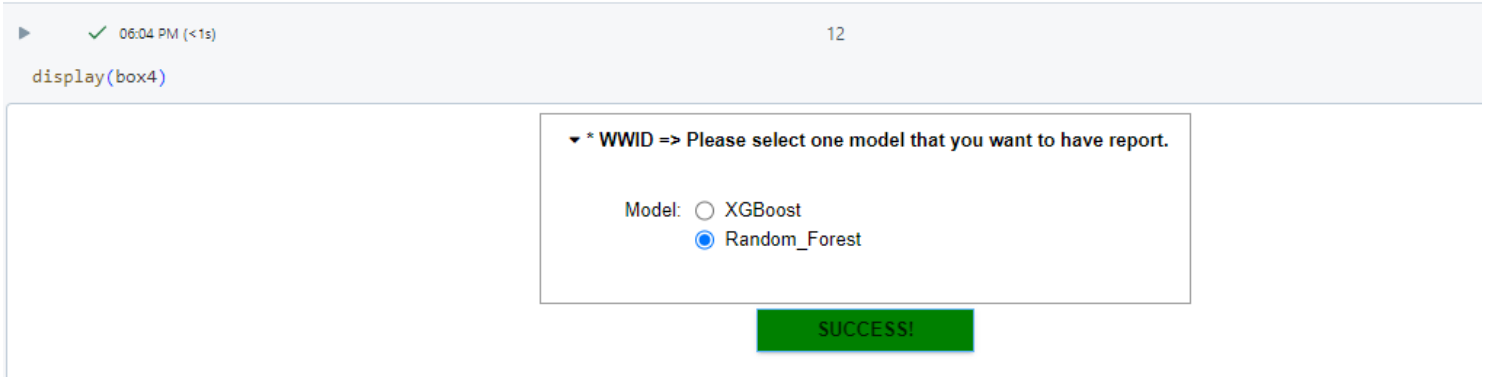
Modelling


05:58 PM (2s)
6
Python
display(box3)
* Feature => Please select top features to refine model.
* Hyperparameters => Default value are given, modify for model tuning.
Hyperpara...
Automatic(Recommend)
Custom(For ML Expert)
SUCCESS!
You have clicked the submit button!
Selected features: ('DEFDoserPressMax_max', 'DEFDoserPressMax_avg', 'UL2ePRVOpenCount_max', 'UL2ePRVOpenCount_min')
Hyperparameter choice: Automatic(Recommend)
ALL COLUMNS ARE FILLED UP!


ROC Cure for Random Forest Model
True Positive Rate
False Positive Rate
ROC curve (area = 0.89)

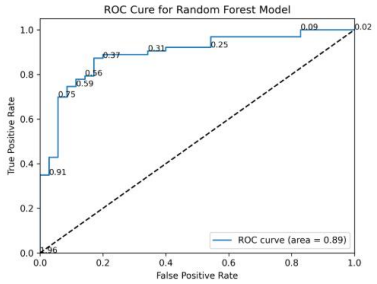

UL2ePRVOpenCount_max
DEFDoserPressMax_avg
DEFDoserPressMax_max
UL2ePRVOpenCount_min
SHAP value (impact on model output)
Feature value
High
Low

Output save as PDF

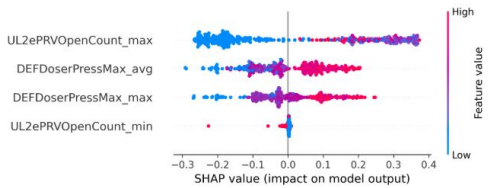


Model Evaluation Report: Random_Forest

ROC Curve:



SHAP Summary:



Thresholds Analysis:

Threshold	FPR	TPR
1.9597927642303228	0.0	0.0
0.9597927642303229	0.0	0.015873015873015872
0.9137028417246982	0.0	0.3492063492063492
0.9132766054991011	0.02857142857142857	0.3492063492063492
0.8929791623223156	0.02857142857142857	0.42857142857142855
0.890796991311656	0.05714285714285714	0.42857142857142855
0.7547136194055084	0.05714285714285714	0.6984126984126984
0.7413234931692024	0.08571428571428572	0.6984126984126984
0.5981255772038807	0.08571428571428572	0.746031746031746
0.5916510942416795	0.11428571428571428	0.746031746031746
0.5725935709738149	0.11428571428571428	0.7777777777777778
0.5718589710595192	0.14285714285714285	0.7777777777777778
0.5596629064200492	0.14285714285714285	0.7936507936507936
0.558501091250706	0.17142857142857143	0.7936507936507936
0.4926464806294362	0.17142857142857143	0.873015873015873
0.3675278721744666	0.2	0.873015873015873
0.36554798059104526	0.2	0.8888888888888888

Appendix

Data Processing Logic:

1. Use Claim table to filter out all claims that failure part match the user input component.
2. Join TNF to filter out real failure.
3. Join eFPA to get 7days data and filter out all ESN include the FC user input before.

```
def getclaimnum(select_com,component,claim_df_path,tnf_path):
    claim_df = spark.read.table(claim_df_path)
    tnf = spark.read.table(tnf_path).select('Claim_number','TNF_Flag').dropna()
    filtered_df = claim_df.where(claim_df.ABO_FAILURE_PART.isin(component)).cache()
    filtered_df = filtered_df.withColumn('label', when(claim_df.ABO_FAILURE_PART.isin(select_com), 1).otherwise(0)).cache()
    filtered_df = filtered_df.select('Claim_number','ESN','ABO_FAILURE_PART','Failure_date','label')
    filtered_df = filtered_df.join(tnf,on='Claim_number',how='left')
    filtered_df = filtered_df[filtered_df['TNF_Flag']!='Y']
    filtered_df = filtered_df.drop('TNF_Flag')
    # Drop duplicate
    counts = filtered_df.groupBy("ESN").agg(count("ESN").alias("count"))
    filtered_df = filtered_df.join(counts, "ESN")
    filtered_df = filtered_df.filter(col("count") == 1).drop("count")
    return filtered_df
## Get eFPA feature
def getefpa(unchanged_feature_list,feature_list,efpa_path,eng_condition1,eng_condition2):
    all_feature = unchanged_feature_list + feature_list
    efpa = spark.read.table(efpa_path).select(all_feature)
    efpa_filter = efpa.filter((split(col('engine_model_group'), '_')[0].isin(eng_condition1)) &  # In China eFPA Engine Family format is "platform_emissionlevel_manufacturer"
        (split(col('engine_model_group'), '_')[1].isin(eng_condition2))).cache()
    return efpa_filter
## filter eFPA with FC in 7d
def filterefpa(claim_df,efpa_df,fc_list):
    claim_with_efpa = claim_df.join(efpa_df,"ESN")
    claim_with_efpa_7d = claim_with_efpa.filter((col("Failure_date") >= col("Occurrence_time")) & (datediff(col("Failure_date"), col("Occurrence_time")) <= 7)).cache()
    conditions = [col("Faultlist").contains(f"[{fc}]") | col("Faultlist2").contains(f"[{fc}]") for fc in fc_list]
    condition = reduce(lambda x, y: x|y, conditions)
    filtered_df = claim_with_efpa_7d.filter(condition).cache()
    grouped_df = filtered_df.groupBy("ESN").count()
    final_df = claim_with_efpa_7d.join(grouped_df.select("ESN"), "ESN").cache()
    return final_df
```

Component list
[Pump, Nozzles,UQS]

Claim

Failure_part	ESN	Failure_date
Pump	10000001	2023.1.1
Nozzles	10000002	2023.4.20
UQS	10000003	2022.3.13
Pump	10000004	2021.3.30

Selected Component
[Pump]

Failure_part	ESN	Failure_date	label
Pump	10000001	2023.1.1	1
Nozzles	10000002	2023.4.20	0
UQS	10000003	2022.3.13	0
Pump	10000004	2021.3.30	1

TNF

Failure_part	ESN	Failure_date	label
Nozzles	10000002	2023.4.20	0
UQS	10000003	2022.3.13	0
Pump	10000004	2021.3.30	1

Failure_part	ESN	Failure_date	label
Nozzles	10000002	2023.4.20	0
UQS	10000003	2022.3.13	0
Pump	10000004	2021.3.30	1

User Inputs
(Faultcodes & Features)
1682

eFPA

Failure date
7 days

Failure_part	ESN	Failure_date	label	Faultlist	Occurrencetime	Features…
Nozzles	10000002	2023.4.20	0	1111,2222	2023.4.20	
Nozzles	10000002	2023.4.20	0	1682	2023.4.16	
Nozzles	10000002	2023.4.20	0	255	2023.4.13	
UQS	10000003	2022.3.13	0	1234	2022.3.13	
UQS	10000003	2022.3.13	0	1111	2022.3.12	
UQS	10000003	2022.3.13	0	255	2022.3.10	
UQS	10000003	2022.3.13	0	2222	2022.3.17	
Pump	10000004	2021.3.30	1	1682	2021.3.30	
Pump	10000004	2021.3.30	1	1682,1111	2021.3.28	
Pump	10000004	2021.3.30	1	2222	2021.3.25	

Aggregation

ESN	label	Feature_max	Feature_min	Feature_avg	Feature_std
10000002	0	100	15	55	11.4
10000004	1	100	5	50	15.7

XGBoost &
RandomForest

Training