In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
data = pd.read_csv("C:/Users/SANDEEP/OneDrive/Desktop/Eclipse/Eclipsedata2.csv")
```

In [5]:
```
for i in data.columns:
    data[i]=data[i].astype(int)
data.shape
```

Out[5]: (7888, 200)

In [6]:
```
zero_cols = (data == 0).all()
data = data.loc[:, ~zero_cols]
data
```

Out[6]:

	pre	ACD	FOUT_avg	FOUT_max	FOUT_sum	MLOC_avg	MLOC_max	MLOC_sum	NBD
0	0	0	3	15	35	7	35	79	
1	0	0	2	12	25	10	16	92	
2	0	0	0	0	0	0	0	0	
3	0	0	4	9	32	7	15	49	
4	0	0	4	11	24	8	39	53	
...	...	...	...	...	...	...	...	...	
7883	0	0	3	10	31	5	15	52	
7884	0	0	0	2	2	2	6	8	
7885	1	6	10	46	120	17	81	204	
7886	0	0	6	25	106	8	34	134	
7887	0	0	0	2	4	1	6	13	

7888 rows × 96 columns

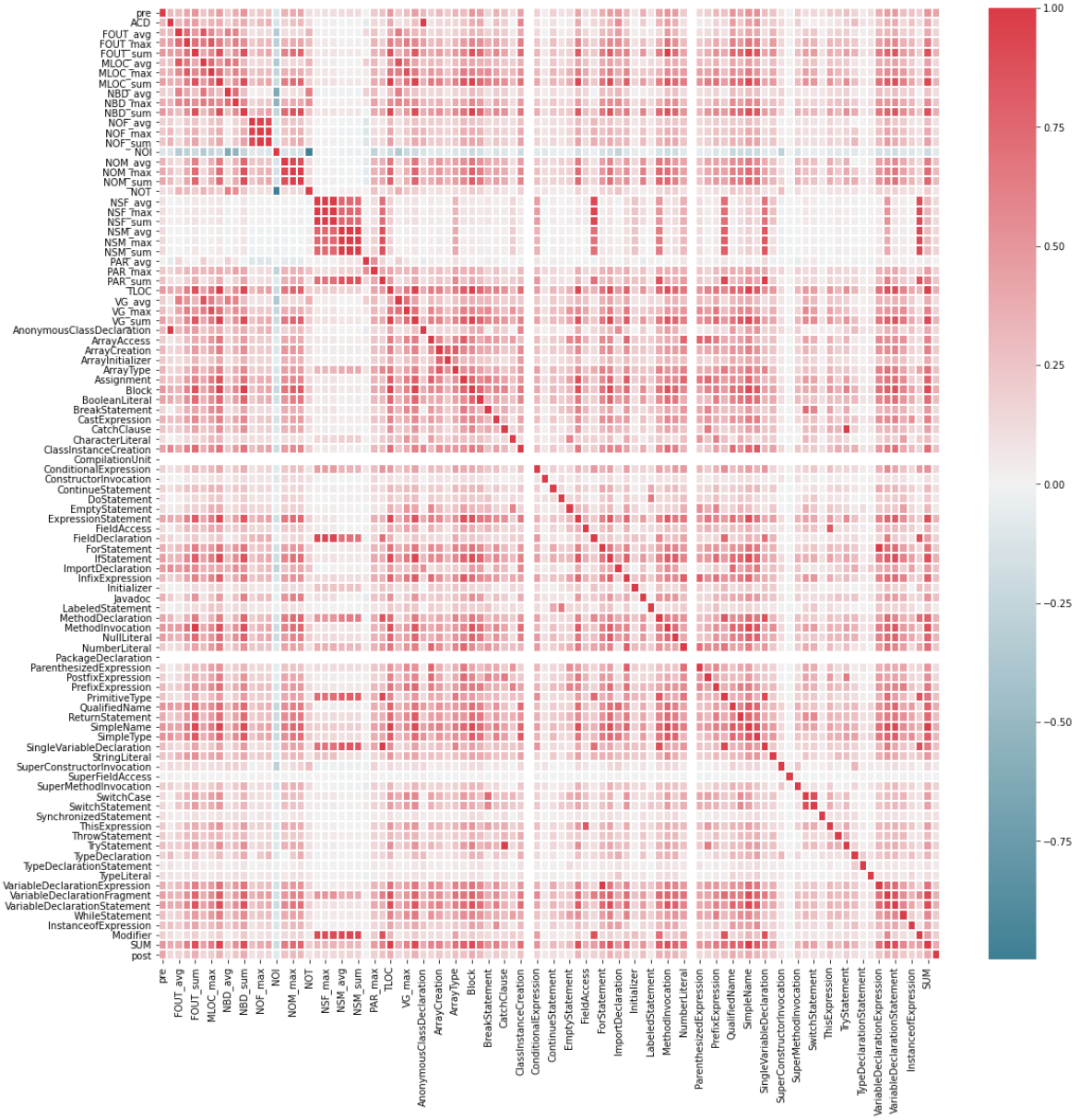
In [7]:
```
corr=data.corr()
corr
```

Out[7]:

	pre	ACD	FOUT_avg	FOUT_max	FOUT_sum	MLOC_avg	MLO
pre	1.000000	0.313830	0.283869	0.407779	0.501487	0.251237	0.3
ACD	0.313830	1.000000	0.308763	0.448432	0.454992	0.248542	0.3
FOUT_avg	0.283869	0.308763	1.000000	0.733544	0.471107	0.836103	0.5
FOUT_max	0.407779	0.448432	0.733544	1.000000	0.697394	0.660235	0.8
FOUT_sum	0.501487	0.454992	0.471107	0.697394	1.000000	0.398068	0.5
...	...	...	...	...	...	...	
WhileStatement	0.278649	0.140794	0.181262	0.351975	0.472553	0.308678	0.4
InstanceofExpression	0.262850	0.263854	0.191771	0.322750	0.418362	0.204886	0.2
Modifier	0.189775	0.102344	0.045596	0.122223	0.195417	0.052720	0.
SUM	0.507290	0.396681	0.378159	0.629589	0.876762	0.432298	0.6
post	0.429497	0.144605	0.157955	0.236446	0.357571	0.142240	0.1

96 rows × 96 columns

In [8]:
```
f,ax=plt.subplots(figsize=(18,18))
cmap=sns.diverging_palette(220,10,as_cmap=True)
heatmap=sns.heatmap(corr,cmap=cmap,center=0.0,vmax=1,linewidths=1,ax=ax)
plt.show()
```



In [9]:
```
#Identify the features with highest correlation to the target variable
target_corr = corr['post']
top_features = target_corr.abs().sort_values(ascending=False)[:100].index
```

In [10]:
```
selected_data = data[top_features]
selected_data
```

Out[10]:

	post	pre	SUM	SimpleName	NBD_sum	Block	QualifiedName	TLOC	VariableDeclaratio
0	0	0	506	181	18	25	12	115	
1	0	0	547	240	16	24	11	145	
2	0	0	86	33	0	0	7	14	
3	0	0	429	184	12	12	13	72	
4	0	0	298	129	9	14	17	73	
...	...	...	...	...	...	...	...	...	
7883	0	0	438	182	13	14	23	80	
7884	0	0	102	40	3	3	15	20	
7885	0	1	1378	590	27	34	84	265	
7886	0	0	911	399	29	38	35	177	
7887	0	0	181	67	7	7	15	34	

7888 rows × 96 columns

In [11]:
```
selected_data = selected_data.drop(['post'],axis=1)
selected_data
```

Out[11]:

	pre	SUM	SimpleName	NBD_sum	Block	QualifiedName	TLOC	VariableDeclarationState
0	0	506	181	18	25	12	115	
1	0	547	240	16	24	11	145	
2	0	86	33	0	0	7	14	
3	0	429	184	12	12	13	72	
4	0	298	129	9	14	17	73	
...	...	...	...	...	...	...	...	
7883	0	438	182	13	14	23	80	
7884	0	102	40	3	3	15	20	
7885	1	1378	590	27	34	84	265	
7886	0	911	399	29	38	35	177	
7887	0	181	67	7	7	15	34	

7888 rows × 95 columns

In [14]:
```
import xgboost as xgb
from sklearn.model_selection import train_test_split
from sklearn import metrics
```

In [15]:
```
X = selected_data
y = data['post']
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2, random_state=1)
```

In [16]:
```
params = {'objective': 'binary:hinge', 'eval_metric': 'logloss'}
```

In [17]:
```
dtrain = xgb.DMatrix(X_train, label=y_train)
dtest = xgb.DMatrix(X_test, label=y_test)
```

In [18]:
```
model = xgb.train(params, dtrain, num_boost_round=100)
```

In [19]:
```
y_pred = model.predict(dtest)
```

In [20]:
```
y_pred = [1 if p >= 0.5 else 0 for p in y_pred]
```

In [21]:
```
a=metrics.accuracy_score(y_test,y_pred)
p=metrics.precision_score(y_test,y_pred,average='weighted',zero_division=0)
r=metrics.recall_score(y_test,y_pred,average='weighted',zero_division=0)
f1=metrics.f1_score(y_test,y_pred,average='weighted',zero_division=0)
print("Accuracy:",a,"Precision: ",p,"Recall: ",r,"F1score: ",f1)
```

```
Accuracy: 0.8605830164765526 Precision:  0.8340547447855596 Recall:  0.86058301647655
26 F1score:  0.8470135537988538
```

In [ ]:
```
#RUS
```

In [22]:
```
from imblearn.under_sampling import RandomUnderSampler
from sklearn.linear_model import LogisticRegression
from sklearn.multiclass import OneVsRestClassifier
```

In [23]:
```
Xr = selected_data
yr = data['post']
Xr_train, Xr_test, yr_train, yr_test = train_test_split(Xr, yr, test_size=0.3, random_state=1)
```

In [24]:
```
rus = RandomUnderSampler(random_state=42)
X_train_rus, y_train_rus = rus.fit_resample(Xr_train, yr_train)
```

In [25]:
```
lr = LogisticRegression(random_state=42)
lr.fit(X_train_rus, y_train_rus)
```

```
C:\Users\SANDEEP\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:458: Conv
ergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/mod
ules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-lea
rn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[25]:
```
LogisticRegression(random_state=42)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [26]:
```
yr_pred = lr.predict(Xr_test)
```

In [27]:
```
a=metrics.accuracy_score(yr_test,yr_pred)
p=metrics.precision_score(yr_test,yr_pred,average='weighted',zero_division=0)
r=metrics.recall_score(yr_test,yr_pred,average='weighted',zero_division=0)
f1=2*(p*r)/(p+r)
print("Accuracy:",a,"Precision: ",p,"Recall: ",r,"F1score: ",f1)
```

```
Accuracy: 0.5369666244190959 Precision:  0.826294355946815 Recall:  0.536966624419095
9 F1score:  0.6509281751322792
```

In [28]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [30]:
```
Xn= selected_data
yn = data['post']
Xn_train, Xn_test, yn_train, yn_test = train_test_split(Xn, yn, test_size=0.3, random_state=1)
```

In [31]:
```
knn = KNeighborsClassifier(n_neighbors=3)
```

In [32]:
```
knn.fit(Xn_train, yn_train)
```

Out[32]:
```
KNeighborsClassifier(n_neighbors=3)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [33]:
```
yn_pred = knn.predict(Xn_test)
```

In [35]:
```
a=metrics.accuracy_score(yn_test,yn_pred)
p=metrics.precision_score(yn_test,yn_pred,average='weighted',zero_division=0)
r=metrics.recall_score(yn_test,yn_pred,average='weighted',zero_division=0)
f1=metrics.f1_score(y_test,y_pred,average='weighted',zero_division=0)
print("Accuracy:",a,"Precision: ",p,"Recall: ",r,"F1score: ",f1)
```

```
Accuracy: 0.8719898605830165 Precision:  0.8213355403725865 Recall:  0.8719898605830165 F1score:  0.8470135537988538
```

In [37]:
```
from sklearn.ensemble import VotingClassifier
from xgboost import XGBClassifier
```

In [38]:
```
Xv= selected_data
yv = data['post']
Xv_train, Xv_test, yv_train, yv_test = train_test_split(Xv, yv, test_size=0.3, random_state=1)
```

In [39]:
```
model1 = KNeighborsClassifier(n_neighbors=5)
model3 = XGBClassifier(n_estimators=100, learning_rate=0.1,objective='binary:logistic',eval_me
model2 = LogisticRegression(random_state=42)
```

In [40]:
```
voting_clf = VotingClassifier(estimators=[('xgb', model1), ('rus', model2),('knn', model3)], voting=
```

In [41]:
```
voting_clf.fit(Xv_train, yv_train)
```

```
C:\Users\SANDEEP\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:458: Conv
ergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/mod
ules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-lea
rn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[41]:
```
VotingClassifier(estimators=[('xgb', KNeighborsClassifier()),
                             ('rus', LogisticRegression(random_state=42)),
                             ('knn',
                              XGBClassifier(base_score=None, booster=None,
                                            callbacks=None,
                                            colsample_bylevel=None,
                                            colsample_bynode=None,
                                            colsample_bytree=None,
                                            early_stopping_rounds=None,
                                            enable_categorical=False,
                                            eval_metric='logloss',
                                            feature_types=None, gamma=None,
                                            gpu_id=None, grow_policy=None,
                                            importance_type=None,
                                            interaction_constraints=None,
                                            learning_rate=0.1, max_bin=None,
                                            max_cat_threshold=None,
                                            max_cat_to_onehot=None,
                                            max_delta_step=None, max_depth=None,
                                            max_leaves=None,
                                            min_child_weight=None, missing=nan,
                                            monotone_constraints=None,
                                            n_estimators=100, n_jobs=None,
                                            num_parallel_tree=None,
                                            predictor=None, random_state=None, ...))])
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [43]:
```
yv_pred = voting_clf.predict(Xv_test)
```

In [44]:
```
a=metrics.accuracy_score(yv_test,yv_pred)
p=metrics.precision_score(yv_test,yv_pred,average='weighted',zero_division=0)
r=metrics.recall_score(yv_test,yv_pred,average='weighted',zero_division=0)
f1=metrics.f1_score(y_test,y_pred,average='weighted',zero_division=0)
print("Accuracy:",a,"Precision: ",p,"Recall: ",r,"F1score: ",f1)
```

```
Accuracy: 0.8859315589353612 Precision:  0.8205093489640337 Recall:  0.8859315589353612 F1score:  0.8470135537988538
```