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

In [12]:
```
data = pd.read_csv("C:/Users/SANDEEP/OneDrive/Desktop/NASA datasers/kc1_csv.csv")
```

In [11]:
```
for i in data.columns:
    data[i]=data[i].astype(int)
data
```

Out[11]:

	loc	v(g)	ev(g)	iv(g)	n	v	l	d	i	e	...	lOCode	lOComment	lOBlank	lo
0	1	1	1	1	1	1	1	1	1	1	...	2	2	2	
1	1	1	1	1	1	1	1	1	1	1	...	1	1	1	
2	83	11	1	11	171	927	0	23	40	21378	...	65	10	6	
3	46	8	6	8	141	769	0	14	51	11436	...	37	2	5	
4	25	3	1	3	58	254	0	9	27	2381	...	21	0	2	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2104	19	2	1	2	40	175	0	6	25	1197	...	12	1	2	
2105	23	3	3	3	60	278	0	9	28	2700	...	18	1	2	
2106	2	1	1	1	4	8	0	1	5	12	...	0	0	0	
2107	13	1	1	1	17	60	0	4	15	243	...	6	0	5	
2108	11	2	1	2	27	102	0	6	17	616	...	9	0	0	

2109 rows × 22 columns

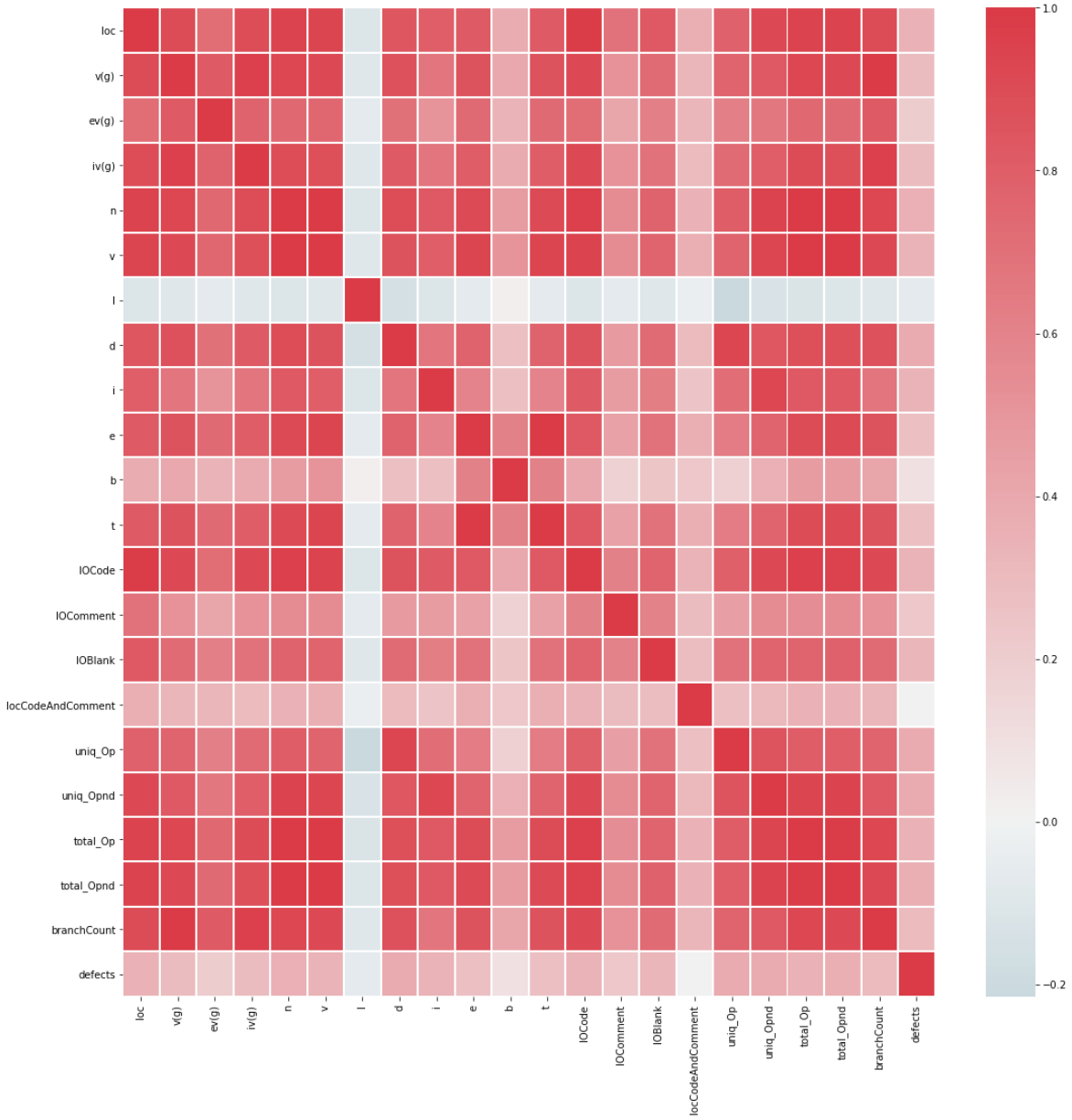
In [10]:
```
corr=data.corr()
corr
```

Out[10]:

	loc	v(g)	ev(g)	iv(g)	n	v	l
loc	1.000000	0.902632	0.718875	0.895183	0.948510	0.938509	-0.110648
v(g)	0.902632	1.000000	0.819979	0.965691	0.922520	0.915569	-0.096869
ev(g)	0.718875	0.819979	1.000000	0.775755	0.746289	0.753612	-0.062994
iv(g)	0.895183	0.965691	0.775755	1.000000	0.893780	0.884937	-0.094170
n	0.948510	0.922520	0.746289	0.893780	1.000000	0.994944	-0.113230
v	0.938509	0.915569	0.753612	0.884937	0.994944	1.000000	-0.100045
l	-0.110648	-0.096869	-0.062994	-0.094170	-0.113230	-0.100045	1.000000
d	0.846915	0.869069	0.699902	0.816050	0.891045	0.859948	-0.151379
i	0.803307	0.673686	0.511792	0.674865	0.827143	0.800298	-0.108742
e	0.821335	0.862408	0.738875	0.807284	0.908209	0.934954	-0.061712
b	0.375858	0.401335	0.336838	0.387409	0.463649	0.511154	0.023958
t	0.821337	0.862404	0.738865	0.807277	0.908202	0.934949	-0.061659
lOCode	0.985491	0.918526	0.723286	0.916928	0.961050	0.949673	-0.108407
lOComment	0.686717	0.517046	0.411720	0.519591	0.547562	0.545153	-0.061285
lOBlank	0.824426	0.731419	0.624956	0.689650	0.777578	0.772091	-0.091932
locCodeAndComment	0.365863	0.328269	0.321557	0.304393	0.347555	0.359965	-0.031088
uniq_Op	0.783487	0.768653	0.616156	0.729793	0.809491	0.771043	-0.215276
uniq_Opnd	0.915715	0.827922	0.666772	0.803426	0.945880	0.931039	-0.128103
total_Op	0.946597	0.925185	0.747860	0.901764	0.998146	0.992036	-0.115961
total_Opnd	0.944167	0.910997	0.737913	0.873896	0.995162	0.991849	-0.107831
branchCount	0.901113	0.998609	0.819810	0.964736	0.923873	0.917393	-0.096723
defects	0.348405	0.295585	0.205192	0.295712	0.355101	0.339487	-0.062956

22 rows × 22 columns

In [13]:
```
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 [18]:
```
dat=data.drop(['l','loc','ev(g)','b','t','lOComment','locCodeAndComment','branchCount'],axis=1)
dat
```

Out[18]:

	v(g)	iv(g)	n	v	d	i	e	lOCode	lOBlank	uniq_Op	uniq_Opnd
0	1.4	1.4	1.3	1.30	1.30	1.30	1.30	2	2	1.2	1.2
1	1.0	1.0	1.0	1.00	1.00	1.00	1.00	1	1	1.0	1.0
2	11.0	11.0	171.0	927.89	23.04	40.27	21378.61	65	6	18.0	25.0
3	8.0	8.0	141.0	769.78	14.86	51.81	11436.73	37	5	16.0	28.0
4	3.0	3.0	58.0	254.75	9.35	27.25	2381.95	21	2	11.0	10.0
...	...	...	...	...	...	...	...	...	...	...	...
2104	2.0	2.0	40.0	175.69	6.82	25.77	1197.90	12	2	10.0	11.0
2105	3.0	3.0	60.0	278.63	9.69	28.75	2700.58	18	2	12.0	13.0
2106	1.0	1.0	4.0	8.00	1.50	5.33	12.00	0	0	3.0	1.0
2107	1.0	1.0	17.0	60.94	4.00	15.24	243.78	6	5	6.0	6.0
2108	2.0	2.0	27.0	102.80	6.00	17.13	616.79	9	0	8.0	6.0

2109 rows × 14 columns

In [19]:
```
lis= ['v(g)','iv(g)','n','v','d','i','e','lOCode','lOBlank','uniq_Op','uniq_Opnd','total_Op','total_Opnd']
fea=data[lis]
fea
```

Out[19]:

	v(g)	iv(g)	n	v	d	i	e	lOCode	lOBlank	uniq_Op	uniq_Opnd
0	1.4	1.4	1.3	1.30	1.30	1.30	1.30	2	2	1.2	1.2
1	1.0	1.0	1.0	1.00	1.00	1.00	1.00	1	1	1.0	1.0
2	11.0	11.0	171.0	927.89	23.04	40.27	21378.61	65	6	18.0	25.0
3	8.0	8.0	141.0	769.78	14.86	51.81	11436.73	37	5	16.0	28.0
4	3.0	3.0	58.0	254.75	9.35	27.25	2381.95	21	2	11.0	10.0
...	...	...	...	...	...	...	...	...	...	...	...
2104	2.0	2.0	40.0	175.69	6.82	25.77	1197.90	12	2	10.0	11.0
2105	3.0	3.0	60.0	278.63	9.69	28.75	2700.58	18	2	12.0	13.0
2106	1.0	1.0	4.0	8.00	1.50	5.33	12.00	0	0	3.0	1.0
2107	1.0	1.0	17.0	60.94	4.00	15.24	243.78	6	5	6.0	6.0
2108	2.0	2.0	27.0	102.80	6.00	17.13	616.79	9	0	8.0	6.0

2109 rows × 13 columns

In [20]:
```
#KNN
```

In [21]:
```
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from imblearn.under_sampling import RandomUnderSampler
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
```

In [23]:
```
X = fea
y = data['defects']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
```

In [25]:
```
smote = SMOTE(random_state=1)
X_train_resampled,y_train_resampled = smote.fit_resample(X_train,y_train)
```

In [26]:
```
knn = KNeighborsClassifier(n_neighbors=3)
```

In [27]:
```
knn.fit(X_train, y_train)
```

Out[27]:
```
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 [28]:
```
y_pred = knn.predict(X_test)
```

In [29]:
```
a=metrics.accuracy_score(y_test,y_pred)
p=metrics.precision_score(y_test,y_pred)
r=metrics.recall_score(y_test,y_pred)
f1=2*(p*r)/(p+r)
print("Accuracy:",a,"   Precision:",p,"   Recall:",r,"   F1Score:",f1)
```

```
Accuracy: 0.8262243285939969    Precision: 0.39655172413793105    Recall: 0.23469387755
102042    F1Score: 0.2948717948717949
```

In [30]:
```
#RUS
```

In [31]:
```
Xr = fea
yr = data['defects']
Xr_train, Xr_test, yr_train, yr_test = train_test_split(Xr, yr, test_size=0.3, random_state=1)
```

In [32]:
```
smote = SMOTE(random_state=42)
X_train_resampled,y_train_resampled = smote.fit_resample(Xr_train,yr_train)
```

In [33]:
```
rus = RandomUnderSampler(random_state=42)
X_train_rus, y_train_rus = rus.fit_resample(Xr_train, yr_train)
```

In [34]:
```
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[34]:
```
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 [35]:
```
yr_pred = lr.predict(Xr_test)
```

In [36]:
```
a=metrics.accuracy_score(yr_test,yr_pred)
p=metrics.precision_score(yr_test,yr_pred)
r=metrics.recall_score(yr_test,yr_pred)
f1=2*(p*r)/(p+r)
print("Accuracy:",a,"   Precision:",p," Recall:",r,"   F1Score:",f1)
```

```
Accuracy: 0.7472353870458136    Precision: 0.32967032967032966  Recall: 0.612244897959
1837    F1Score: 0.4285714285714286
```

In [37]:
```
#XGBOOST
```

In [38]:
```
import xgboost as xgb
```

In [39]:
```
Xx = data[lis]
yx = data["defects"]
Xx_train, Xx_test, yx_train, yx_test = train_test_split(Xx, yx, test_size=0.3, random_state=1)
```

In [40]:
```
smote = SMOTE(random_state=1)
X_train_resampled,y_train_resampled = smote.fit_resample(Xx_train,yx_train)
```

In [41]:
```
params = {'objective': 'binary:logistic', 'eval_metric': 'logloss'}
```

In [42]:
```
dtrain = xgb.DMatrix(Xx_train, label=yx_train)
dtest = xgb.DMatrix(Xx_test, label=yx_test)
```

In [43]:
```
model = xgb.train(params, dtrain, num_boost_round=100)
```

In [44]:
```
yx_pred = model.predict(dtest)
```

In [45]:
```
yx_pred = [1 if p >= 0.5 else 0 for p in yx_pred]
```

In [46]:
```
a=metrics.accuracy_score(yx_test,yx_pred)
p=metrics.precision_score(yx_test,yx_pred)
r=metrics.recall_score(yx_test,yx_pred)
f1=2*(p*r)/(p+r)
print("Accuracy:",a,"   Precision:",p,"   Recall:",r,"   F1Score:",f1)
```

```
Accuracy: 0.8530805687203792    Precision: 0.5454545454545454    Recall: 0.306122448979
59184    F1Score: 0.39215686274509803
```

In [47]:
```
#Voting Ensemble
```

In [48]:
```
from sklearn.ensemble import VotingClassifier
from xgboost import XGBClassifier
```

In [50]:
```
X = fea
y = data['defects']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

In [51]:
```
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 [52]:
```
voting_clf = VotingClassifier(estimators=[('xgb', model1), ('rus', model2),('knn', model3)], voting=
```

In [53]:
```
voting_clf.fit(X_train, y_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[53]:
```
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 [54]:
```
y_predic = voting_clf.predict(X_test)
```

In [55]:
```
a=metrics.accuracy_score(y_test,y_predic)
p=metrics.precision_score(y_test,y_predic)
r=metrics.recall_score(y_test,y_predic)
f1=metrics.f1_score(y_test,y_predic)
print("Accuracy:",a,"Precision: ",p,"Recall: ",r,"F1score: ",f1)
```

```
Accuracy: 0.8436018957345972 Precision:  0.5128205128205128 Recall:  0.2 F1score:  0.287
7697841726619
```

In [ ]:
```

```