In [1]:

```
import joblib
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
from sklearn.preprocessing import StandardScaler
from numpy import random
from sklearn.decomposition import PCA
from sklearn.impute import KNNImputer
from sklearn.metrics import confusion_matrix
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler
from imblearn.pipeline import Pipeline
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler
from imblearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
```

In [ ]:

```

```

In [2]:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [3]:

```
def function_1(x):


    median_imputer = joblib.load('/content/drive/MyDrive/median_impute.pkl')

    # here we are loading the logistic regression model
    #model = joblib.load('model.pkl')

    # here we are giing to replace the na value to np.NaN value

    x = x.replace('na', np.NaN)


    # here we droping the those feature name which have more than 75% of missing value an
d which data have std_daviation is zero
    x = x.drop(['cd_000', 'br_000', 'bq_000', 'bp_000', 'bq_000', 'bo_000','ab_000','cr_0
00'], axis = 1)

    # here we are manualy selecting the those feature name which have less than 15% missi
ng values
    median_data = ['aa_000', 'ac_000','ae_000','af_000','ag_001','ag_002','ag_003','ag_00
4','ag_005','ag_006','ag_007','ag_008','ag_009','ah_000','ai_000','aj_000','ak_000','al_0
00','am_0','an_000','ao_000','ap_000','aq_000','ar_000','as_000','at_000','au_000','av_00
0','ax_000','ay_000','ay_001','ay_002','ay_003','ay_004','ay_005','ay_006','ay_007','ay_0
08','ay_009','az_000','az_001','az_002','az_003','az_004','az_005','az_006','az_007','az_
008','az_009','ba_000','ba_001','ba_002','ba_003','ba_004','ba_005','ba_006','ba_007','ba
_008','ba_009','bb_000','bc_000','bd_000','be_000','bf_000','bg_000','bh_000','bi_000','b
j_000','bs_000','bt_000','bu_000','bv_000','bx_000','by_000','bz_000','ca_000','cc_000','
ce_000','ci_000','cj_000','ck_000','cn_000','cn_001','cn_002','cn_003','cn_004','cn_005',
'cn_006','cn_007','cn_008','cn_009','cp_000','cq_000','cs_000','cs_001','cs_002','cs_003'
,'cs_004','cs_005','cs_006','cs_007','cs_008','cs_009','dd_000','de_000','df_000','dg_000
','dh_000','di_000','dj_000','dk_000','dl_000','dm_000','dn_000','do_000','dp_000','dq_00
0','dr_000','ds_000','dt_000','du_000','dv_000','dx_000','dy_000','dz_000','ea_000','eb_0
00','ee_000','ee_001','ee_002','ee_003','ee_004','ee_005','ee_006','ee_007','ee_008','ee_
009','ef_000','eg_000']
```

```
    # here we are selecting the those feature which have less than 75% and more than 15%
missing values
    model_data = ['ad_000', 'bk_000', 'bl_000', 'bm_000', 'bn_000', 'cf_000', 'cg_000','c
h_000', 'cl_000', 'cm_000', 'co_000', 'ct_000', 'cu_000', 'cv_000','cx_000', 'cy_000', 'c
z_000', 'da_000', 'db_000', 'dc_000', 'ec_00','ed_000']

    # here we are seprating those feature which are going to impute by median_imputation
from data set
    median_df = x.filter(median_data)

    # here we are seperating the model_impute feature
    model_df = x.filter(model_data)

    median_imp = median_imputer.fit_transform(median_df)

    # making the data frame
    median_imp_df = pd.DataFrame(median_imp, columns= median_df.columns)

    # KNN imputation
    imputer = KNNImputer()
    model_imp_test = imputer.fit_transform(model_df)
    model_imp_df = pd.DataFrame(model_imp_test, columns= model_df.columns)

    # here we are concatinating the median_imputed dataframe and model_imputed data
    data_df = pd.concat((median_imp_df, model_imp_df), axis = 1)

    scalar =StandardScaler()
    scalar.fit(data_df)
    scal_data = scalar.transform(data_df)
    scaler_data = pd.DataFrame(scal_data, columns = data_df.columns)

    pca = PCA(n_components= 0.95)
    pca_df = pca.fit_transform(scaler_data)
    pca_df = pd.DataFrame(pca_df)

    final_df = pd.concat((pca_df, scaler_data), axis = 1)

    model = joblib.load('/content/drive/MyDrive/model.pkl')
    y = model.predict(final_df)

    return y
```

In [16]:

```
def function_2(x, y):

  median_imputer = joblib.load('/content/drive/MyDrive/median_impute.pkl')

    # here we are loading the logistic regression model
    #model = joblib.load('model.pkl')

    # here we are giing to replace the na value to np.NaN value
   # here we are neg and pos with 0 a
  x = x.replace('na', np.NaN)
  #remap = {'neg':0, 'pos': 1}
  #x = x.replace(remap)
  #y = x['class']

    # here we droping the those feature name which have more than 75% of missing value an
d which data have std_daviation is zero
  x = x.drop(['cd_000', 'br_000', 'bq_000', 'bp_000', 'bq_000', 'bo_000','ab_000','cr_000
'], axis = 1)

    # here we are manualy selecting the those feature name which have less than 15% missi
ng values
  median_data = ['aa_000', 'ac_000','ae_000','af_000','ag_001','ag_002','ag_003','ag_004'
,'ag_005','ag_006','ag_007','ag_008','ag_009','ah_000','ai_000','aj_000','ak_000','al_000
','am_0','an_000','ao_000','ap_000','aq_000','ar_000','as_000','at_000','au_000','av_000'
,'ax_000','ay_000','ay_001','ay_002','ay_003','ay_004','ay_005','ay_006','ay_007','ay_008
','ay_009','az_000','az_001','az_002','az_003','az_004','az_005','az_006','az_007','az_00
```

```
8','az_009','ba_000','ba_001','ba_002','ba_003','ba_004','ba_005','ba_006','ba_007','ba_0
08','ba_009','bb_000','bc_000','bd_000','be_000','bf_000','bg_000','bh_000','bi_000','bj_
000','bs_000','bt_000','bu_000','bv_000','bx_000','by_000','bz_000','ca_000','cc_000','ce
_000','ci_000','cj_000','ck_000','cn_000','cn_001','cn_002','cn_003','cn_004','cn_005','c
n_006','cn_007','cn_008','cn_009','cp_000','cq_000','cs_000','cs_001','cs_002','cs_003','
cs_004','cs_005','cs_006','cs_007','cs_008','cs_009','dd_000','de_000','df_000','dg_000',
'dh_000','di_000','dj_000','dk_000','dl_000','dm_000','dn_000','do_000','dp_000','dq_000'
,'dr_000','ds_000','dt_000','du_000','dv_000','dx_000','dy_000','dz_000','ea_000','eb_000
','ee_000','ee_001','ee_002','ee_003','ee_004','ee_005','ee_006','ee_007','ee_008','ee_00
9','ef_000','eg_000']

    # here we are selecting the those feature which have less than 75% and more than 15%
missing values
  model_data = ['ad_000', 'bk_000', 'bl_000', 'bm_000', 'bn_000', 'cf_000', 'cg_000','ch_
000', 'cl_000', 'cm_000', 'co_000', 'ct_000', 'cu_000', 'cv_000','cx_000', 'cy_000', 'cz_
000', 'da_000', 'db_000', 'dc_000', 'ec_00','ed_000']

    # here we are seprating those feature which are going to impute by median_imputation
from data set
  median_df = x.filter(median_data)

    # here we are seperating the model_impute feature
  model_df = x.filter(model_data)

  median_imp = median_imputer.fit_transform(median_df)

    # making the data frame
  median_imp_df = pd.DataFrame(median_imp, columns= median_df.columns)

    # KNN imputation
  imputer = KNNImputer()
  model_imp_test = imputer.fit_transform(model_df)
  model_imp_df = pd.DataFrame(model_imp_test, columns= model_df.columns)

    # here we are concatinating the median_imputed dataframe and model_imputed data
  data_df = pd.concat((median_imp_df, model_imp_df), axis = 1)
  scalar =StandardScaler()
  scalar.fit(data_df)
  scal_data = scalar.transform(data_df)
  scaler_data = pd.DataFrame(scal_data, columns = data_df.columns)

  pca = PCA(n_components= 0.95)
  pca_df = pca.fit_transform(scaler_data)
  pca_df = pd.DataFrame(pca_df)
  final_df = pd.concat((pca_df, scaler_data), axis = 1)

  model = joblib.load('/content/drive/MyDrive/model_1.pkl')
  y_pred = model.predict(final_df)

  cm = confusion_matrix(y, y_pred)
  total_cast = (cm[0][1] * 10 + cm[1][0] * 500)


  return total_cast
```

In [6]:

```
data = pd.read_csv('/content/drive/MyDrive/aps_failure_test_set.csv', header = 'infer',s
kiprows= 20)
remap = {'neg': 0, 'pos': 1}
data = data.replace(remap)
Y_test = data['class']
X_test = data.drop('class', axis = 1)
```

In [17]:

```
Total_cast = function_2(X_test, Y_test)
print("total cast is:",Total_cast)
```

```
total cast is: 11340
```

In [ ]: