def calculate\_woe\_iv(dataset, feature\_b, target, targetcat):  
     
    dataset[feature\_b].replace([np.inf, -np.inf], np.nan, inplace = True)  
     
    if dataset[feature\_b].dtype in ['float64', 'int64']:  
        #dataset[dataset[feature\_b]< 0][feature\_b] = 0  
        dataset['feature'] = pd.qcut(dataset[feature\_b], q = [0, .05, .1, .2,.4, .6, .8,0.9,0.95, 1],  
                                     duplicates = 'drop', precision = 0).values.add\_categories(['Missing', 'Others'])  
    else:  
        dataset['feature'] = dataset[feature\_b]  
  
    dataset['feature'].fillna('Missing', inplace = True)  
  
    lst = []  
    num\_rows = dataset.shape[0]  
  
    val\_list = dataset['feature'].unique()  
    for val in val\_list:  
        occ\_val = dataset[dataset['feature'] == val].count()['feature']  
        if occ\_val < 0.005 \* num\_rows :  
            dataset['feature'] = np.where(dataset['feature'] == val, 'Others', dataset['feature'])  
  
    val\_list = dataset['feature'].unique()  
    for val in val\_list:  
        lst.append({  
        'Feature': feature\_b,  
        'Value': val,  
        'Min\_val': dataset[dataset['feature'] == val].min()[feature\_b],  
        'All': dataset[dataset['feature'] == val].count()['feature'],  
        'Good': dataset[(dataset['feature'] == val) & (dataset[target] != targetcat )].count()['feature'],  
        'Bad': dataset[(dataset['feature'] == val) & (dataset[target] == targetcat)].count()['feature']  
        })  
  
    dset = pd.DataFrame(lst)  
    dset['Event\_rate'] = dset['Bad'] / dset['All']  
    dset['Prop'] = dset['All'] / dset['All'].sum()  
    dset['Prop\_Good'] = dset['Good'] / dset['Good'].sum()  
    dset['Prop\_Bad'] = dset['Bad'] / dset['Bad'].sum()  
    dset['WoE'] = np.log(dset['Prop\_Good'] / dset['Prop\_Bad'])  
    dset = dset.replace({'WoE': {np.inf: 0, -np.inf: 0}})  
    dset['IV\_component'] = (dset['Prop\_Good'] - dset['Prop\_Bad']) \* dset['WoE']  
  
    iv = dset['IV\_component'].sum()  
  
    dset = dset.sort\_values(by = 'Min\_val')  
  
    return dset, iv  
  
def bivariate\_graphs(base01,feat,targetcol, target\_category):  
  
    import matplotlib.pyplot as plt  
    import numpy as np  
    import pandas as pd  
  
  
    w , i = calculate\_woe\_iv(base01[[feat,targetcol]], feat, targetcol,target\_category)  
     
    fig = plt.figure()  
    plt.rcParams["figure.figsize"] = (18, 7)  
  
    width = .75 # width of a bar  
  
    i\_temp = w.head(10)  
    i\_temp['Proportion of Observations'] = i\_temp['Prop']  
    i\_temp[ 'Percentage of '+ str(targetcol )+ " : " + str(target\_category)] = i\_temp['Event\_rate']  
  
    i\_temp[['Proportion of Observations','Percentage of '+ str(targetcol )+ " : " + str(target\_category)]].plot(kind='bar', width = width )  
        #i\_temp['Conversion\_Bad'].plot()  
  
    ax = plt.gca()  
    plt.xlim([-width, len(i\_temp['Value'])-width])  
    ax.set\_xticklabels(i\_temp['Value'] )  
  
    ax.set\_title(str('Relationship with ' + target\_category + ' : ' + feat),fontsize = 28)  
       #plt.legend(loc='upper right', fontsize = 18)  
  
    box = ax.get\_position()  
    ax.set\_position([box.x0, box.y0 + box.height \* 0.1,  
                         box.width, box.height \* 0.9])  
  
    ax.legend(loc='upper center', bbox\_to\_anchor=(0.5, -0.3),  
                  fancybox=True, shadow=True, ncol=5,fontsize = 18)  
  
  
    plt.xticks(fontsize=18, rotation=60)  
    plt.yticks(fontsize=18)  
  
  
    plt.show()