Earth Quake Damage Prediction Model Using Machine Learning

Importing essential libraries

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as ss
from scipy.stats import chi2_contingency
from scipy.stats import chi2
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import homogeneity_score
from sklearn.metrics import silhouette_score
from sklearn.metrics import classification_report,f1_score,precision_score,recall_sc

#To Ignore Warnings
import warnings
warnings.filterwarnings('ignore')
```

Importing and understanding our dataset

In [2]:
```
train=pd.read_csv('D:\Major Project\Dataset/train.csv')
```

Shape of dataset

In [3]:
```
train.shape
```

Out[3]: (631761, 14)

Printing out a few columns

In [4]:
```
train.head()
```

Out[4]:

	area_assesed	building_id	damage_grade	district_id	has_geotechnical_risk	has_geotechnical_risk_f
0	Both	24385bfd2a2	Grade 4	24	0.0	
1	Both	405d1bbebbf	Grade 2	44	0.0	
2	Both	351d9bc71f6	Grade 1	36	0.0	
3	Building removed	2be3a971166	Grade 5	30	0.0	
4	Both	34c7d073ea6	Grade 3	36	0.0	

In [5]:
```
train.columns
```

Out[5]:
```
Index(['area_assesed', 'building_id', 'damage_grade', 'district_id',
       'has_geotechnical_risk', 'has_geotechnical_risk_fault_crack',
       'has_geotechnical_risk_flood', 'has_geotechnical_risk_land_settlement',
       'has_geotechnical_risk_landslide', 'has_geotechnical_risk_liquefaction',
       'has_geotechnical_risk_other', 'has_geotechnical_risk_rock_fall',
       'has_repair_started', 'vdcmun_id'],
      dtype='object')
```

In [6]:
```
len(train.columns)
```

Out[6]:
```
14
```

In [7]:
```
train.isnull().sum()
```

Out[7]:
```
area_assesed                                 0
building_id                                  0
damage_grade                                 0
district_id                                  0
has_geotechnical_risk                        0
has_geotechnical_risk_fault_crack            0
has_geotechnical_risk_flood                  0
has_geotechnical_risk_land_settlement        0
has_geotechnical_risk_landslide              0
has_geotechnical_risk_liquefaction           0
has_geotechnical_risk_other                  0
has_geotechnical_risk_rock_fall              0
has_repair_started                       33417
vdcmun_id                                    0
dtype: int64
```

Using of other datasets

In [8]:

```python
owner = pd.read_csv('D:\Major Project\Dataset/Building_Ownership_Use.csv')
structure = pd.read_csv('D:\Major Project\Dataset/Building_Structure.csv')
owner.info()
structure.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1052948 entries, 0 to 1052947
Data columns (total 17 columns):
 #   Column                         Non-Null Count    Dtype  
---  ------                         --------------    -----  
 0   building_id                    1052948 non-null  object 
 1   district_id                    1052948 non-null  int64  
 2   vdcmun_id                      1052948 non-null  int64  
 3   ward_id                        1052948 non-null  int64  
 4   legal_ownership_status         1052948 non-null  object 
 5   count_families                 1052946 non-null  float64
 6   has_secondary_use              1052938 non-null  float64
 7   has_secondary_use_agriculture  1052948 non-null  int64  
 8   has_secondary_use_hotel        1052948 non-null  int64  
 9   has_secondary_use_rental       1052948 non-null  int64  
 10  has_secondary_use_institution  1052948 non-null  int64  
 11  has_secondary_use_school       1052948 non-null  int64  
 12  has_secondary_use_industry     1052948 non-null  int64  
 13  has_secondary_use_health_post  1052948 non-null  int64  
 14  has_secondary_use_gov_office   1052948 non-null  int64  
 15  has_secondary_use_use_police   1052948 non-null  int64  
 16  has_secondary_use_other        1052948 non-null  int64  
dtypes: float64(2), int64(13), object(2)
memory usage: 136.6+ MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1052948 entries, 0 to 1052947
Data columns (total 29 columns):
 #   Column                                  Non-Null Count    Dtype 
---  ------                                  --------------    ----- 
 0   building_id                             1052948 non-null  object
 1   district_id                             1052948 non-null  int64 
 2   vdcmun_id                               1052948 non-null  int64 
 3   ward_id                                 1052948 non-null  int64 
 4   count_floors_pre_eq                     1052948 non-null  int64 
 5   count_floors_post_eq                    1052948 non-null  int64 
 6   age_building                            1052948 non-null  int64 
 7   plinth_area_sq_ft                       1052948 non-null  int64 
 8   height_ft_pre_eq                        1052948 non-null  int64 
 9   height_ft_post_eq                       1052948 non-null  int64 
 10  land_surface_condition                  1052948 non-null  object
 11  foundation_type                         1052948 non-null  object
 12  roof_type                               1052948 non-null  object
 13  ground_floor_type                       1052948 non-null  object
 14  other_floor_type                        1052948 non-null  object
 15  position                                1052947 non-null  object
 16  plan_configuration                      1052947 non-null  object
 17  has_superstructure_adobe_mud            1052948 non-null  int64 
 18  has_superstructure_mud_mortar_stone     1052948 non-null  int64 
 19  has_superstructure_stone_flag           1052948 non-null  int64 
 20  has_superstructure_cement_mortar_stone  1052948 non-null  int64 
 21  has_superstructure_mud_mortar_brick     1052948 non-null  int64 
 22  has_superstructure_cement_mortar_brick  1052948 non-null  int64 
 23  has_superstructure_timber               1052948 non-null  int64 
 24  has_superstructure_bamboo               1052948 non-null  int64 
 25  has_superstructure_rc_non_engineered    1052948 non-null  int64 
 26  has_superstructure_rc_engineered        1052948 non-null  int64 
 27  has_superstructure_other                1052948 non-null  int64 
 28  condition_post_eq                       1052948 non-null  object
dtypes: int64(20), object(9)
memory usage: 233.0+ MB
```

Combining of other datasets for preprocessing

In [9]:
```
combine = pd.merge(owner,structure, on='building_id')
trainfinal = pd.merge(combine,train, on = 'building_id')
```

Train Data before preprocessing

In [10]:
```
trainfinal.columns,len(trainfinal.columns)
```

Out[10]:
```
(Index(['building_id', 'district_id_x', 'vdcmun_id_x', 'ward_id_x',
        'legal_ownership_status', 'count_families', 'has_secondary_use',
        'has_secondary_use_agriculture', 'has_secondary_use_hotel',
        'has_secondary_use_rental', 'has_secondary_use_institution',
        'has_secondary_use_school', 'has_secondary_use_industry',
        'has_secondary_use_health_post', 'has_secondary_use_gov_office',
        'has_secondary_use_use_police', 'has_secondary_use_other',
        'district_id_y', 'vdcmun_id_y', 'ward_id_y', 'count_floors_pre_eq',
        'count_floors_post_eq', 'age_building', 'plinth_area_sq_ft',
        'height_ft_pre_eq', 'height_ft_post_eq', 'land_surface_condition',
        'foundation_type', 'roof_type', 'ground_floor_type', 'other_floor_type',
        'position', 'plan_configuration', 'has_superstructure_adobe_mud',
        'has_superstructure_mud_mortar_stone', 'has_superstructure_stone_flag',
        'has_superstructure_cement_mortar_stone',
        'has_superstructure_mud_mortar_brick',
        'has_superstructure_cement_mortar_brick', 'has_superstructure_timber',
        'has_superstructure_bamboo', 'has_superstructure_rc_non_engineered',
        'has_superstructure_rc_engineered', 'has_superstructure_other',
        'condition_post_eq', 'area_assesed', 'damage_grade', 'district_id',
        'has_geotechnical_risk', 'has_geotechnical_risk_fault_crack',
        'has_geotechnical_risk_flood', 'has_geotechnical_risk_land_settlement',
        'has_geotechnical_risk_landslide', 'has_geotechnical_risk_liquefaction',
        'has_geotechnical_risk_other', 'has_geotechnical_risk_rock_fall',
        'has_repair_started', 'vdcmun_id'],
       dtype='object'),
 58)
```

In [11]:

```
trainfinal.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 631761 entries, 0 to 631760
Data columns (total 58 columns):
 #   Column                                  Non-Null Count   Dtype  
---  ------                                  --------------   -----  
 0   building_id                             631761 non-null  object 
 1   district_id_x                           631761 non-null  int64  
 2   vdcmun_id_x                             631761 non-null  int64  
 3   ward_id_x                               631761 non-null  int64  
 4   legal_ownership_status                  631761 non-null  object 
 5   count_families                          631760 non-null  float64
 6   has_secondary_use                       631761 non-null  float64
 7   has_secondary_use_agriculture           631761 non-null  int64  
 8   has_secondary_use_hotel                 631761 non-null  int64  
 9   has_secondary_use_rental                631761 non-null  int64  
 10  has_secondary_use_institution           631761 non-null  int64  
 11  has_secondary_use_school                631761 non-null  int64  
 12  has_secondary_use_industry              631761 non-null  int64  
 13  has_secondary_use_health_post           631761 non-null  int64  
 14  has_secondary_use_gov_office            631761 non-null  int64  
 15  has_secondary_use_use_police            631761 non-null  int64  
 16  has_secondary_use_other                 631761 non-null  int64  
 17  district_id_y                           631761 non-null  int64  
 18  vdcmun_id_y                             631761 non-null  int64  
 19  ward_id_y                               631761 non-null  int64  
 20  count_floors_pre_eq                     631761 non-null  int64  
 21  count_floors_post_eq                    631761 non-null  int64  
 22  age_building                            631761 non-null  int64  
 23  plinth_area_sq_ft                       631761 non-null  int64  
 24  height_ft_pre_eq                        631761 non-null  int64  
 25  height_ft_post_eq                       631761 non-null  int64  
 26  land_surface_condition                  631761 non-null  object 
 27  foundation_type                         631761 non-null  object 
 28  roof_type                               631761 non-null  object 
 29  ground_floor_type                       631761 non-null  object 
 30  other_floor_type                        631761 non-null  object 
 31  position                                631761 non-null  object 
 32  plan_configuration                      631761 non-null  object 
 33  has_superstructure_adobe_mud            631761 non-null  int64  
 34  has_superstructure_mud_mortar_stone     631761 non-null  int64  
 35  has_superstructure_stone_flag           631761 non-null  int64  
 36  has_superstructure_cement_mortar_stone  631761 non-null  int64  
 37  has_superstructure_mud_mortar_brick     631761 non-null  int64  
 38  has_superstructure_cement_mortar_brick  631761 non-null  int64  
 39  has_superstructure_timber               631761 non-null  int64  
 40  has_superstructure_bamboo               631761 non-null  int64  
 41  has_superstructure_rc_non_engineered    631761 non-null  int64  
 42  has_superstructure_rc_engineered        631761 non-null  int64  
 43  has_superstructure_other                631761 non-null  int64  
 44  condition_post_eq                       631761 non-null  object 
 45  area_assesed                            631761 non-null  object 
 46  damage_grade                            631761 non-null  object 
 47  district_id                             631761 non-null  int64  
 48  has_geotechnical_risk                   631761 non-null  float64
 49  has_geotechnical_risk_fault_crack       631761 non-null  int64  
 50  has_geotechnical_risk_flood             631761 non-null  int64  
 51  has_geotechnical_risk_land_settlement   631761 non-null  int64  
 52  has_geotechnical_risk_landslide         631761 non-null  int64  
 53  has_geotechnical_risk_liquefaction      631761 non-null  int64  
 54  has_geotechnical_risk_other             631761 non-null  int64  
 55  has_geotechnical_risk_rock_fall         631761 non-null  int64  
 56  has_repair_started                      598344 non-null  float64
 57  vdcmun_id                               631761 non-null  int64  
dtypes: float64(4), int64(42), object(12)
memory usage: 284.4+ MB
```

Type *Markdown* and LaTeX: α^2

In [12]:
```
trainfinal.isnull().sum()
```

Out[12]:
```
building_id                                0
district_id_x                              0
vdcmun_id_x                                0
ward_id_x                                  0
legal_ownership_status                     0
count_families                             1
has_secondary_use                          0
has_secondary_use_agriculture              0
has_secondary_use_hotel                    0
has_secondary_use_rental                   0
has_secondary_use_institution              0
has_secondary_use_school                   0
has_secondary_use_industry                 0
has_secondary_use_health_post              0
has_secondary_use_gov_office               0
has_secondary_use_use_police               0
has_secondary_use_other                    0
district_id_y                              0
vdcmun_id_y                                0
ward_id_y                                  0
count_floors_pre_eq                        0
count_floors_post_eq                       0
age_building                               0
plinth_area_sq_ft                          0
height_ft_pre_eq                           0
height_ft_post_eq                          0
land_surface_condition                     0
foundation_type                            0
roof_type                                  0
ground_floor_type                          0
other_floor_type                           0
position                                   0
plan_configuration                         0
has_superstructure_adobe_mud               0
has_superstructure_mud_mortar_stone        0
has_superstructure_stone_flag              0
has_superstructure_cement_mortar_stone     0
has_superstructure_mud_mortar_brick        0
has_superstructure_cement_mortar_brick     0
has_superstructure_timber                  0
has_superstructure_bamboo                  0
has_superstructure_rc_non_engineered       0
has_superstructure_rc_engineered           0
has_superstructure_other                   0
condition_post_eq                          0
area_assesed                               0
damage_grade                               0
district_id                                0
has_geotechnical_risk                      0
has_geotechnical_risk_fault_crack          0
has_geotechnical_risk_flood                0
has_geotechnical_risk_land_settlement      0
has_geotechnical_risk_landslide            0
has_geotechnical_risk_liquefaction         0
has_geotechnical_risk_other                0
has_geotechnical_risk_rock_fall            0
has_repair_started                     33417
vdcmun_id                                  0
dtype: int64
```

In [13]:
```
trainfinal.drop('has_repair_started',axis=1,inplace=True)
```

In [14]:
```
trainfinal.drop(['vdcmun_id_y','district_id_y','ward_id_y','vdcmun_id','district_id
```

In [15]:
```
features = list(trainfinal.columns)
```

Let's understand our columns better:

In [16]:
```
trainfinal.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 631761 entries, 0 to 631760
Data columns (total 52 columns):
 #   Column                                  Non-Null Count   Dtype  
---  ------                                  --------------   -----  
 0   building_id                             631761 non-null  object 
 1   district_id_x                           631761 non-null  int64  
 2   vdcmun_id_x                             631761 non-null  int64  
 3   ward_id_x                               631761 non-null  int64  
 4   legal_ownership_status                  631761 non-null  object 
 5   count_families                          631760 non-null  float64
 6   has_secondary_use                       631761 non-null  float64
 7   has_secondary_use_agriculture           631761 non-null  int64  
 8   has_secondary_use_hotel                 631761 non-null  int64  
 9   has_secondary_use_rental                631761 non-null  int64  
 10  has_secondary_use_institution           631761 non-null  int64  
 11  has_secondary_use_school                631761 non-null  int64  
 12  has_secondary_use_industry              631761 non-null  int64  
 13  has_secondary_use_health_post           631761 non-null  int64  
 14  has_secondary_use_gov_office            631761 non-null  int64  
 15  has_secondary_use_use_police            631761 non-null  int64  
 16  has_secondary_use_other                 631761 non-null  int64  
 17  count_floors_pre_eq                     631761 non-null  int64  
 18  count_floors_post_eq                    631761 non-null  int64  
 19  age_building                            631761 non-null  int64  
 20  plinth_area_sq_ft                       631761 non-null  int64  
 21  height_ft_pre_eq                        631761 non-null  int64  
 22  height_ft_post_eq                       631761 non-null  int64  
 23  land_surface_condition                  631761 non-null  object 
 24  foundation_type                         631761 non-null  object 
 25  roof_type                               631761 non-null  object 
 26  ground_floor_type                       631761 non-null  object 
 27  other_floor_type                        631761 non-null  object 
 28  position                                631761 non-null  object 
 29  plan_configuration                      631761 non-null  object 
 30  has_superstructure_adobe_mud            631761 non-null  int64  
 31  has_superstructure_mud_mortar_stone     631761 non-null  int64  
 32  has_superstructure_stone_flag           631761 non-null  int64  
 33  has_superstructure_cement_mortar_stone  631761 non-null  int64  
 34  has_superstructure_mud_mortar_brick     631761 non-null  int64  
 35  has_superstructure_cement_mortar_brick  631761 non-null  int64  
 36  has_superstructure_timber               631761 non-null  int64  
 37  has_superstructure_bamboo               631761 non-null  int64  
 38  has_superstructure_rc_non_engineered    631761 non-null  int64  
 39  has_superstructure_rc_engineered        631761 non-null  int64  
 40  has_superstructure_other                631761 non-null  int64  
 41  condition_post_eq                       631761 non-null  object 
 42  area_assesed                            631761 non-null  object 
 43  damage_grade                            631761 non-null  object 
 44  has_geotechnical_risk                   631761 non-null  float64
 45  has_geotechnical_risk_fault_crack       631761 non-null  int64  
 46  has_geotechnical_risk_flood             631761 non-null  int64  
 47  has_geotechnical_risk_land_settlement   631761 non-null  int64  
 48  has_geotechnical_risk_landslide         631761 non-null  int64  
 49  has_geotechnical_risk_liquefaction      631761 non-null  int64  
 50  has_geotechnical_risk_other             631761 non-null  int64  
 51  has_geotechnical_risk_rock_fall         631761 non-null  int64  
dtypes: float64(3), int64(37), object(12)
memory usage: 255.5+ MB
```

Converting Nominal to Numeric for the purpose of classification(Target Variable)

In [17]:
```
conversion = {'damage_grade' : {"Grade 1" : 1, "Grade 2" : 2, "Grade 3" : 3,"Grade 4
train_temp = pd.DataFrame()
train_temp['damage_grade'] = trainfinal['damage_grade']

train_temp.replace(conversion, inplace = True)
trainfinal['damage_grade'] = train_temp['damage_grade']
```

In [18]:
```
cat = [c for c in trainfinal if trainfinal[c].dtypes == "object"]
cat.remove('building_id')
print(cat)
```

```
['legal_ownership_status', 'land_surface_condition', 'foundation_type', 'roof_typ
e', 'ground_floor_type', 'other_floor_type', 'position', 'plan_configuration', 'con
dition_post_eq', 'area_assesed']
```

Chi-Square test on Each Types for Understanding dependency of Target with each type

In [19]:
```
def ChiSquareTest(cat,res_train):

  for c in cat:
    print(c)
    tab = pd.crosstab(res_train['damage_grade'], res_train[c])
    stat, p, dof, expected = chi2_contingency(tab)
    print('dof=%d' % dof)
    prob = 0.95
    critical = chi2.ppf(prob, dof)
    print('probability=%.3f, critical=%.3f, stat=%.3f' % (prob, critical, stat))
    if abs(stat) >= critical:
      print('Dependent (reject H0)')
    else:
      print('Independent (fail to reject H0)')
    # interpret p-value
    alpha = 1.0 - prob
    print('significance=%.3f, p=%.3f' % (alpha, p))
    if p <= alpha:
      print('Dependent (reject H0)')
    else:
      print('Independent (fail to reject H0)')

    print(" ")
```

Performing Chi-Square test on Object Types for Understanding dependency of Target with each object type

In [20]:
```
ChiSquareTest(cat,trainfinal)
```

```
legal_ownership_status
dof=12
probability=0.950, critical=21.026, stat=8113.932
Dependent (reject H0)
significance=0.050, p=0.000
Dependent (reject H0)

land_surface_condition
dof=8
probability=0.950, critical=15.507, stat=1408.392
Dependent (reject H0)
significance=0.050, p=0.000
Dependent (reject H0)

foundation_type
dof=16
probability=0.950, critical=26.296, stat=138103.743
Dependent (reject H0)
significance=0.050, p=0.000
Dependent (reject H0)

roof_type
dof=8
probability=0.950, critical=15.507, stat=85099.545
Dependent (reject H0)
significance=0.050, p=0.000
Dependent (reject H0)

ground_floor_type
dof=16
probability=0.950, critical=26.296, stat=107177.045
Dependent (reject H0)
significance=0.050, p=0.000
Dependent (reject H0)

other_floor_type
dof=12
probability=0.950, critical=21.026, stat=93978.074
Dependent (reject H0)
significance=0.050, p=0.000
Dependent (reject H0)

position
dof=12
probability=0.950, critical=21.026, stat=5026.896
Dependent (reject H0)
significance=0.050, p=0.000
Dependent (reject H0)

plan_configuration
dof=36
probability=0.950, critical=50.998, stat=4672.139
Dependent (reject H0)
significance=0.050, p=0.000
Dependent (reject H0)

condition_post_eq
dof=28
probability=0.950, critical=41.337, stat=1112041.847
Dependent (reject H0)
significance=0.050, p=0.000
Dependent (reject H0)

area_assesed
dof=16
```

```
probability=0.950, critical=26.296, stat=429576.136
Dependent (reject H0)
significance=0.050, p=0.000
Dependent (reject H0)
```

In [21]:

```python
cat_binary = [c for c in trainfinal if len(trainfinal[c].unique()) == 2]
cat_binary
```

Out[21]:

```
['has_secondary_use',
 'has_secondary_use_agriculture',
 'has_secondary_use_hotel',
 'has_secondary_use_rental',
 'has_secondary_use_institution',
 'has_secondary_use_school',
 'has_secondary_use_industry',
 'has_secondary_use_health_post',
 'has_secondary_use_gov_office',
 'has_secondary_use_use_police',
 'has_secondary_use_other',
 'has_superstructure_adobe_mud',
 'has_superstructure_mud_mortar_stone',
 'has_superstructure_stone_flag',
 'has_superstructure_cement_mortar_stone',
 'has_superstructure_mud_mortar_brick',
 'has_superstructure_cement_mortar_brick',
 'has_superstructure_timber',
 'has_superstructure_bamboo',
 'has_superstructure_rc_non_engineered',
 'has_superstructure_rc_engineered',
 'has_superstructure_other',
 'has_geotechnical_risk',
 'has_geotechnical_risk_fault_crack',
 'has_geotechnical_risk_flood',
 'has_geotechnical_risk_land_settlement',
 'has_geotechnical_risk_landslide',
 'has_geotechnical_risk_liquefaction',
 'has_geotechnical_risk_other',
 'has_geotechnical_risk_rock_fall']
```

Performing Chi-Square test on Binary Types for Understanding dependency of Target with each object type

In [22]:

```python
ChiSquareTest(cat_binary,trainfinal)
```

```
dof=4
probability=0.950, critical=9.488, stat=3825.490
Dependent (reject H0)
significance=0.050, p=0.000
Dependent (reject H0)

has_superstructure_rc_non_engineered
dof=4
probability=0.950, critical=9.488, stat=28652.127
Dependent (reject H0)
significance=0.050, p=0.000
Dependent (reject H0)

has_superstructure_rc_engineered
dof=4
probability=0.950, critical=9.488, stat=37327.720
Dependent (reject H0)
significance=0.050, p=0.000
Dependent (reject H0)

has_superstructure_other
```

Removal columns on which Target is not dependent(By Chi-Square Test)

In [23]:
```
trainfinal.drop(['has_secondary_use_use_police','building_id'], axis = 1,inplace=Tru
```

Converting Nominal to Numeric in preprocessed data for the purpose of classification

In [24]:
```
cont = [c for c in trainfinal if len(trainfinal[c].unique()) > 15]
indices = 0,1,2,3
cont = [i for j, i in enumerate(cont) if j not in indices]
```

In [25]:
```
train_copy = trainfinal
```

In [26]:
```
train_copy['IsPrivate'] = (train_copy["legal_ownership_status"] == "Private") * 1
```

In [27]:
```
train_copy['IsFlat'] = (train_copy["land_surface_condition"] == "Flat") * 1
train_copy['IsMudFoundation'] = (train_copy["foundation_type"] == "Mud mortar-Stone/
train_copy['IsBambooRoofLight'] = (train_copy["roof_type"] == "Bamboo/Timber-Light r
train_copy['IsFloorTypeMud'] = (train_copy["ground_floor_type"] == "Mud") * 1
train_copy['OtherFloorTypeMud'] = (train_copy["other_floor_type"] == "TImber/Bamboo-
train_copy['IsNotAttached'] = (train_copy["position"] == "Not attached") * 1
train_copy['IsPlanConfigRectangular'] = (train_copy["plan_configuration"] == "Rectan
train_copy['count_floors_change'] = (train_copy['count_floors_post_eq'] - train_copy
train_copy['height_ft_change'] = (train_copy['height_ft_post_eq'] - train_copy['heig
```

In [28]:
```
train_copy.drop(['count_floors_pre_eq', 'height_ft_pre_eq'], axis=1, inplace=True)
```

In [29]:
```
remove_columns = ["legal_ownership_status","land_surface_condition","foundation_type
def dropColumns(res_train_copy,remove_columns):
  for i in remove_columns:
    res_train_copy.drop([i],axis = 1, inplace = True)
  return res_train_copy
```

In [30]:
```
train_copy = dropColumns(train_copy,remove_columns)
```

In [31]:
```
train_copy.shape
```

Out[31]:
```
(631761, 48)
```

In [32]:
```
train_copy['count_families'].fillna(train_copy['count_families'].mode()[0],inplace=T
```

In [33]:
```
train_one_hot = pd.get_dummies(train_copy)
train_one_hot.drop(["district_id_x","vdcmun_id_x","ward_id_x"],axis = 1, inplace = T
```

Supervised Learning

Modeling,Training and Prediction

The dataset has been cleaned, pre-processed and analyzed for understanding the dataset. After such a process, and yet before coming to modeling, the dataset has to split up into two parts: Train and Test dataset. The training dataset is used to train the algorithm used to prepare an algorithm to comprehend. To learn and deliver results. It incorporates both input data and the desired output. The test data collection is utilized to assess how well the algorithm was prepared with the trained dataset.

By using the Supervised learning Algorithms to train the dataset and also to test, predictions were made as to the desired outcome. The system was able to split, train and test the dataset. Along with that, the feature importance was also given as the output where it had the percentage of these possibilities in occurring in the mere future datasets that would be added.

The aim for us is to predict the Damage grade assigned to the building after assessment of Earth Quake The following are Supervised Algorithms which we used for prediction

In [34]:
```
z_train = train_one_hot['damage_grade']
train_one_hot.drop(['damage_grade'], axis = 1, inplace = True)
```

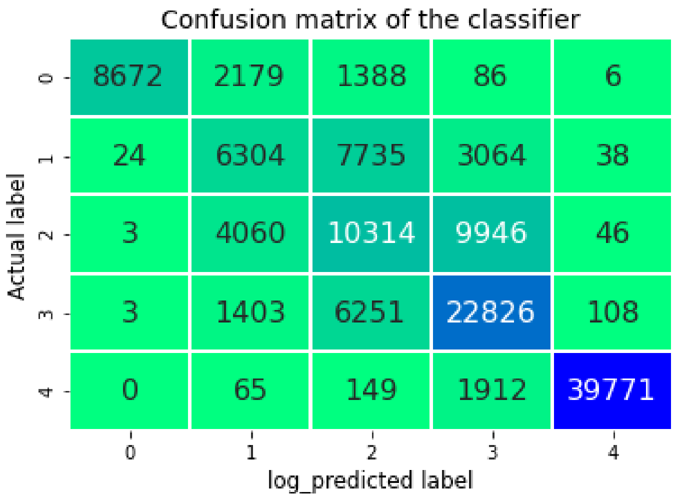
In [35]:
```
from sklearn.model_selection import train_test_split
X_train, X_test,y_train, y_test = train_test_split(train_one_hot, z_train, test_size
```

Logistic Regresion

Logistic regression is named for the function used at the core of the method, the logistic function. The logistic function, also called the sigmoid function was developed by statisticians to describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits. 1 / (1 + e^-value) Where e is the base of the natural logarithms (Euler's number or the EXP() function in your spreadsheet) and value is the actual numerical value that you want to transform.

In [36]:

```
#importing required module
from sklearn.linear_model import LogisticRegression

#intializating model
logistic_reg = LogisticRegression(solver="liblinear")

#fitting the train data
logistic_reg.fit(X_train, y_train)

# predicting the data using test data
log_predicted = logistic_reg.predict(X_test)
probs = logistic_reg.predict_proba(X_test)

#finding the confusion matrix
conf_mat = confusion_matrix(y_true=y_test, y_pred=log_predicted)

# visualizing confusion matrix
sns.heatmap(conf_mat, annot=True, annot_kws={"size":16}, fmt="d", cbar=False, linewi
plt.title("Confusion matrix of the classifier", fontsize=14)
plt.ylabel("Actual label", fontsize=12)
plt.xlabel("log_predicted label", fontsize=12)
plt.show()

# printing the classifcation report
print("Understanding Classifciation:\n")
print(classification_report(y_test, log_predicted))
```



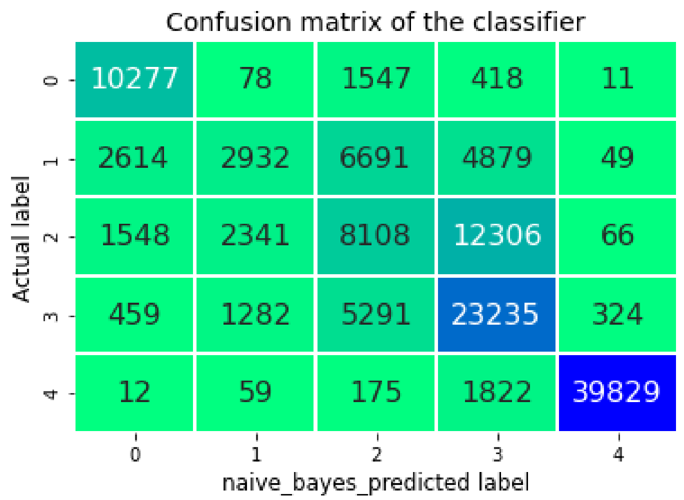

```
Understanding Classifciation:

              precision    recall  f1-score   support

           1       1.00      0.70      0.82     12331
           2       0.45      0.37      0.40     17165
           3       0.40      0.42      0.41     24369
           4       0.60      0.75      0.67     30591
           5       1.00      0.95      0.97     41897

    accuracy                           0.70    126353
   macro avg       0.69      0.64      0.66    126353
weighted avg       0.71      0.70      0.70    126353
```

Naive Bayes

Naive Bayes is a classification algorithm for binary (two-class) and multi-class classification problems. The technique is easiest to understand when described using binary or categorical input values. It is called naive Bayes or idiot Bayes because the calculation of the probabilities for each hypothesis are simplified to make their calculation tractable. Rather than attempting to calculate the values of each attribute value P(d1, d2, d3|h), they are assumed to be conditionally independent given the target value and calculated as P(d1|h) * P(d2|H) and so on.This is a very strong assumption that is most unlikely in real data, i.e. that the attributes do not interact. Nevertheless, the approach performs surprisingly well on data where this assumption does not hold.

Here our classification is for Multi-class

In [37]:
```
from sklearn.naive_bayes import GaussianNB
naive_bayes = GaussianNB()

#fitting the train data
naive_bayes.fit(X_train, y_train)

# predicting the data using test data
naive_bayes_predicted = naive_bayes.predict(X_test)
probs = naive_bayes.predict_proba(X_test)


#finding the confusion matrix
conf_mat = confusion_matrix(y_true=y_test, y_pred=naive_bayes_predicted)

# visualizing confusion matrix
sns.heatmap(conf_mat, annot=True, annot_kws={"size":16}, fmt="d", cbar=False, linewi
plt.title("Confusion matrix of the classifier", fontsize=14)
plt.ylabel("Actual label", fontsize=12)
plt.xlabel("naive_bayes_predicted label", fontsize=12)
plt.show()

# printing the classifcation report
print("Understanding Classifciation:\n")
print(classification_report(y_test, naive_bayes_predicted))
```



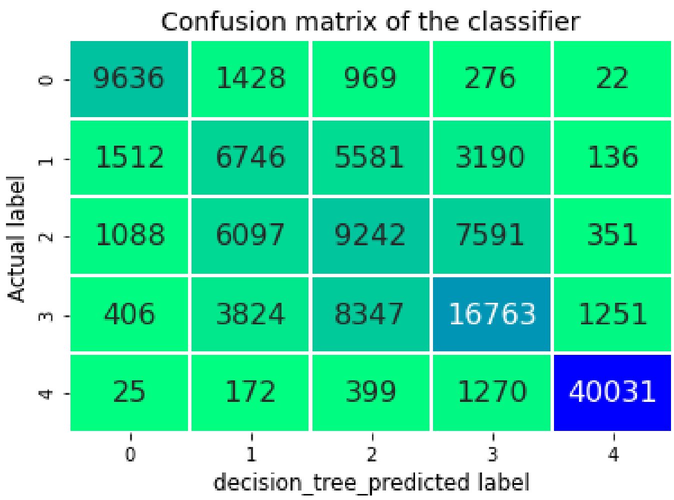

```
Understanding Classifciation:

              precision    recall  f1-score   support

           1       0.69      0.83      0.75     12331
           2       0.44      0.17      0.25     17165
           3       0.37      0.33      0.35     24369
           4       0.54      0.76      0.63     30591
           5       0.99      0.95      0.97     41897

    accuracy                           0.67    126353
   macro avg       0.61      0.61      0.59    126353
weighted avg       0.66      0.67      0.65    126353
```

Decision Tree

Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. An instance is classified by starting at the root node of the tree,testing the attribute specified by this node,then moving down the tree branch corresponding to the value of the attribute as shown in the above figure.This process is then repeated for the subtree rooted at the new node

In [38]:

```
#importing required module
from sklearn.tree import DecisionTreeClassifier
decision_tree = DecisionTreeClassifier(random_state=0, class_weight="balanced")

#fitting the train data
decision_tree.fit(X_train, y_train)

# predicting the data using test data
decision_tree_predicted = decision_tree.predict(X_test)
probs = decision_tree.predict_proba(X_test)

#finding the confusion matrix
conf_mat = confusion_matrix(y_true=y_test, y_pred=decision_tree_predicted)

# visualizing confusion matrix
sns.heatmap(conf_mat, annot=True, annot_kws={"size":16}, fmt="d", cbar=False, linewi
plt.title("Confusion matrix of the classifier", fontsize=14)
plt.ylabel("Actual label", fontsize=12)
plt.xlabel("decision_tree_predicted label", fontsize=12)
plt.show()

# printing the classifcation report
print("Understanding Classifciation:\n")
print(classification_report(y_test, decision_tree_predicted))
```




```
Understanding Classifciation:

              precision    recall  f1-score   support

           1       0.76      0.78      0.77     12331
           2       0.37      0.39      0.38     17165
           3       0.38      0.38      0.38     24369
           4       0.58      0.55      0.56     30591
           5       0.96      0.96      0.96     41897

    accuracy                           0.65    126353
   macro avg       0.61      0.61      0.61    126353
weighted avg       0.65      0.65      0.65    126353
```

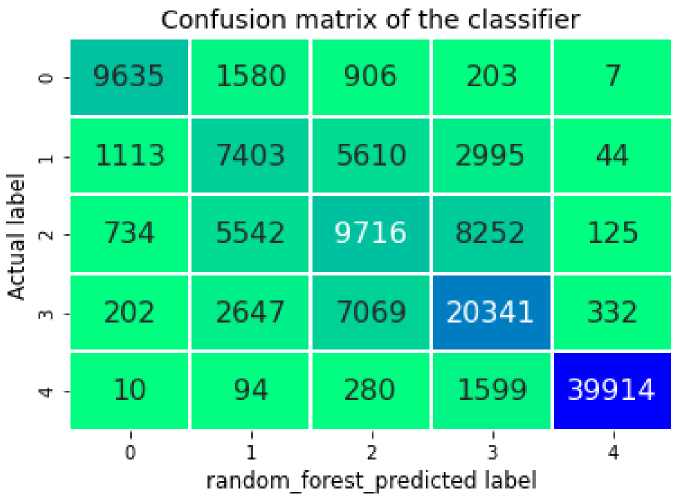
RandomForest

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

In [39]:

```
#importing required module
from sklearn.ensemble import RandomForestClassifier
random_forest = RandomForestClassifier(class_weight="balanced_subsample", random_sta

#fitting the train data
random_forest.fit(X_train, y_train)

# predicting the data using test data
random_forest_predicted = random_forest.predict(X_test)
probs = random_forest.predict_proba(X_test)


#finding the confusion matrix
conf_mat = confusion_matrix(y_true=y_test, y_pred=random_forest_predicted)

# visualizing confusion matrix
sns.heatmap(conf_mat, annot=True, annot_kws={"size":16}, fmt="d", cbar=False, linewi
plt.title("Confusion matrix of the classifier", fontsize=14)
plt.ylabel("Actual label", fontsize=12)
plt.xlabel("random_forest_predicted label", fontsize=12)
plt.show()


# printing the classifcation report
print("Understanding Classifciation:\n")
print(classification_report(y_test, random_forest_predicted))
```

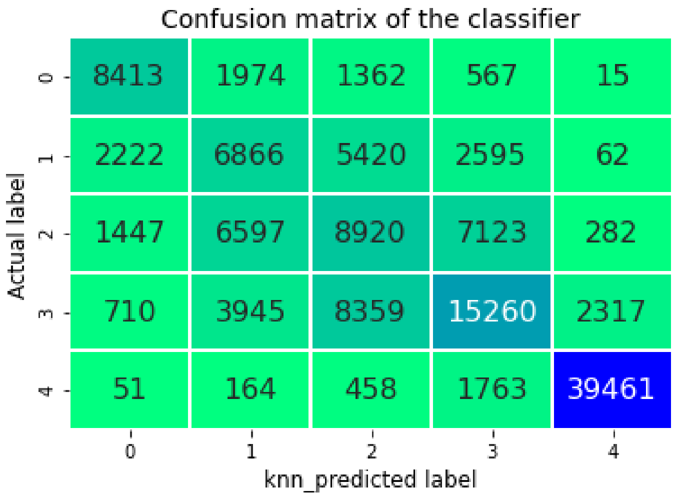



```
Understanding Classifciation:

              precision    recall  f1-score   support

           1       0.82      0.78      0.80     12331
           2       0.43      0.43      0.43     17165
           3       0.41      0.40      0.41     24369
           4       0.61      0.66      0.64     30591
           5       0.99      0.95      0.97     41897

    accuracy                           0.69    126353
   macro avg       0.65      0.65      0.65    126353
weighted avg       0.69      0.69      0.69    126353
```

K-Nearest Neighbours

K-Nearest Neighbors is a machine learning technique and algorithm that can be used for both regression and classification tasks. K-Nearest Neighbors examines the labels of a chosen number of data points surrounding a target data point, in order to make a prediction about the class that the data point falls into. K-Nearest Neighbors (KNN) is a conceptually simple yet very powerful algorithm, and for those reasons, it's one of the most popular machine learning algorithms. Let's take a deep dive into the KNN algorithm and see exactly how it works. Having a good understanding of how KNN operates will let you appreciated the best and worst use cases for KNN.

In [40]:
```
#imported required model
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()

#fitting the train data
knn.fit(X_train, y_train)

# predicting the data using test data
knn_predicted = knn.predict(X_test)
probs = knn.predict_proba(X_test)


#finding the confusion matrix
conf_mat = confusion_matrix(y_true=y_test, y_pred=knn_predicted)

# visualizing confusion matrix
sns.heatmap(conf_mat, annot=True, annot_kws={"size":16}, fmt="d", cbar=False, linewi
plt.title("Confusion matrix of the classifier", fontsize=14)
plt.ylabel("Actual label", fontsize=12)
plt.xlabel("knn_predicted label", fontsize=12)
plt.show()

# printing the classifcation report
print("Understanding Classifciation:\n")
print(classification_report(y_test, knn_predicted))
```




```
Understanding Classifciation:

              precision    recall  f1-score   support

           1       0.66      0.68      0.67     12331
           2       0.35      0.40      0.37     17165
           3       0.36      0.37      0.36     24369
           4       0.56      0.50      0.53     30591
           5       0.94      0.94      0.94     41897

    accuracy                           0.62    126353
   macro avg       0.57      0.58      0.57    126353
weighted avg       0.63      0.62      0.63    126353
```

Confusion Matrix

A confusion matrix is a technique for summarizing the performance of a classification algorithm.

Classification accuracy alone can be misleading if you have an unequal number of observations in each class or if you have more than two classes in your dataset.

Calculating a confusion matrix can give you a better idea of what your classification model is getting right and what types of errors it is making.

Confusion matrix is sqaure matrix of size n where n is the number of values for targeted variable

Let us consider a confusion matrix A nxn

A[i,j] indicates the number of times i is predicted as j

In our case n=5 0th row - grade 1,1st row - grade 2,2nd row - grade 3,3rd row - grade 4,4th row - grade 5

Classification Report

True Positives (TP) -

These are the correctly predicted positive values which means that the value of actual class is yes and the value of predicted class is also yes.

True Negatives (TN) -

These are the correctly predicted negative values which means that the value of actual class is no and value of predicted class is also no.

False Positives (FP) –

When actual class is no and predicted class is yes.

False Negatives (FN) –

When actual class is yes but predicted class in no.

Accuracy -

Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. One may think that, if we have high accuracy then our model is best. Yes, accuracy is a great measure but only when you have symmetric datasets where values of false positive and false negatives are almost same. Therefore, you have to look at other parameters to evaluate the performance of your model. For our model, we have got 0.803 which means our model is approx. 80% accurate.

Accuracy = TP+TN/TP+FP+FN+TN

Precision -

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. The question that this metric answer is of all passengers that labeled as survived, how many actually survived? High precision relates to the low false positive rate. We have got 0.788 precision which is pretty good.

Precision = TP/TP+FP

Recall (Sensitivity) -

Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes. The question recall answers is: Of all the passengers that truly survived, how many did we label? We have got recall of 0.631 which is good for this model as it’s above 0.5.

Recall = TP/TP+FN

F1 score -

F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall. In our case, F1 score is 0.701.

F1 Score = 2*(Recall * Precision) / (Recall + Precision)

Accuracy Calculation

In [41]:
```
score_lr=accuracy_score(y_test, log_predicted)
score_nb=accuracy_score(y_test, naive_bayes_predicted)
score_dt=accuracy_score(y_test, decision_tree_predicted)
score_rf=accuracy_score(y_test, random_forest_predicted)
score_knn=accuracy_score(y_test, knn_predicted)
```

In [42]:
```
scores = [score_lr,score_dt,score_nb,score_knn,score_rf]
percentscore=[i*100 for i in scores]
```

In [43]:
```
algorithms = ["Logistic Regression","Decision Tree","Naive Bayes","K-Nearest Neighbo

for i in range(len(algorithms)):
    print("The accuracy score achieved using "+algorithms[i]+" is: "+str(percentscor
```

```
The accuracy score achieved using Logistic Regression is: 69.55671808346457%
The accuracy score achieved using Decision Tree is: 65.2283681432178%
The accuracy score achieved using Naive Bayes is: 66.78195214992917%
The accuracy score achieved using K-Nearest Neighbors is: 62.45993367787073%
The accuracy score achieved using Random Forest is: 68.86183944979541%
```

Accuracy Table

In [44]:
```
from tabulate import tabulate
data=[]
for i in range(len(algorithms)):
    list=[algorithms[i],percentscore[i]]
    data.append(list)
print(tabulate(data, headers=["Algorithm","Accuracy"]))
```

```
Algorithm              Accuracy
-------------------  ----------
Logistic Regression     69.5567
Decision Tree           65.2284
Naive Bayes             66.782
K-Nearest Neighbors     62.4599
Random Forest           68.8618
```

Accuracy Graph

In [45]:

```
sns.set(rc={'figure.figsize':(9,6)})
plt.xlabel("Algorithms")
plt.ylabel("Accuracy score")

sns.barplot(algorithms,percentscore)
```

Out[45]: <AxesSubplot:xlabel='Algorithms', ylabel='Accuracy score'>