Earthquake damage modelling: Data Cleaning

In [1]:
```
#Importing necessary Libraries
%matplotlib inline
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
import warnings
warnings.filterwarnings("ignore")
```

In [3]:
```
structure_data=pd.read_csv('csv_building_structure.csv')
ownership_data=pd.read_csv('csv_building_ownership_and_use.csv')
damage_data=pd.read_csv('csv_building_damage_assessment.csv')
```

Structure Data:

In [4]:
```
structure_data.shape
```

Out[4]:
```
(762106, 31)
```

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

Out[5]:
```
Index(['building_id', 'district_id', 'vdcmun_id', 'ward_id',
       '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', 'damage_grade', 'technical_solution_proposed'],
      dtype='object')
```

In [6]:
```
structure_data.head(2)
```

Out[6]:

	building_id	district_id	vdcmun_id	ward_id	count_floors_pre_eq	count_floors_post_eq	age_building	plinth_area_sq_ft	height_ft_pre_eq	height_ft_post_e
0	120101000011	12	1207	120703	1	1	9	288	9	
1	120101000021	12	1207	120703	1	1	15	364	9	

2 rows × 31 columns

we only keep the data that contains structural information of the properties

In [7]:
```
cols_to_drop=['condition_post_eq','technical_solution_proposed']
```

In [8]:
```
structure_data=structure_data.drop(cols_to_drop,axis=1)
structure_data.shape
```

Out[8]:
```
(762106, 29)
```

Ownership Data:

In [42]:
```
ownership_data.shape
```

Out[42]:
```
(762106, 17)
```

In [9]:
```
ownership_data.columns
```

Out[9]:
```
Index(['building_id', 'district_id', 'vdcmun_id', 'ward_id',
       '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'],
      dtype='object')
```

In [10]:
```
ownership_data.head(2)
```

Out[10]:

	building_id	district_id	vdcmun_id	ward_id	legal_ownership_status	count_families	has_secondary_use	has_secondary_use_agriculture	has_secondary_u
0	120101000011	12	1207	120703	Private	1.0	0.0	0	
1	120101000021	12	1207	120703	Private	1.0	0.0	0	

Damage Data:

In [48]:
```
damage_data.shape
```

Out[48]:
```
(762106, 79)
```

In [49]:
```
damage_data.columns
```

Out[49]:
```
Index(['building_id', 'district_id', 'vdcmun_id', 'ward_id',
       'damage_overall_collapse', 'damage_overall_leaning',
```

```
       'damage_overall_adjacent_building_risk', 'damage_foundation_severe',
       'damage_foundation_moderate', 'damage_foundation_insignificant',
       'damage_roof_severe', 'damage_roof_moderate',
       'damage_roof_insignificant', 'damage_corner_separation_severe',
       'damage_corner_separation_moderate',
       'damage_corner_separation_insignificant',
       'damage_diagonal_cracking_severe', 'damage_diagonal_cracking_moderate',
       'damage_diagonal_cracking_insignificant',
       'damage_in_plane_failure_severe', 'damage_in_plane_failure_moderate',
       'damage_in_plane_failure_insignificant',
       'damage_out_of_plane_failure_severe',
       'damage_out_of_plane_failure_moderate',
       'damage_out_of_plane_failure_insignificant',
       'damage_out_of_plane_failure_walls_ncfr_severe',
       'damage_out_of_plane_failure_walls_ncfr_moderate',
       'damage_out_of_plane_failure_walls_ncfr_insignificant',
       'damage_gable_failure_severe', 'damage_gable_failure_moderate',
       'damage_gable_failure_insignificant',
       'damage_delamination_failure_severe',
       'damage_delamination_failure_moderate',
       'damage_delamination_failure_insignificant',
       'damage_column_failure_severe', 'damage_column_failure_moderate',
       'damage_column_failure_insignificant', 'damage_beam_failure_severe',
       'damage_beam_failure_moderate', 'damage_beam_failure_insignificant',
       'damage_infill_partition_failure_severe',
       'damage_infill_partition_failure_moderate',
       'damage_infill_partition_failure_insignificant',
       'damage_staircase_severe', 'damage_staircase_moderate',
       'damage_staircase_insignificant', 'damage_parapet_severe',
       'damage_parapet_moderate', 'damage_parapet_insignificant',
       'damage_cladding_glazing_severe', 'damage_cladding_glazing_moderate',
       'damage_cladding_glazing_insignificant', 'area_assesed', 'damage_grade',
       'technical_solution_proposed', 'has_repair_started',
       'has_damage_foundation', 'has_damage_roof',
       'has_damage_corner_separation', 'has_damage_diagonal_cracking',
       'has_damage_in_plane_failure', 'has_damage_out_of_plane_failure',
       'has_damage_out_of_plane_walls_ncfr_failure',
       'has_damage_gable_failure', 'has_damage_delamination_failure',
       'has_damage_column_failure', 'has_damage_beam_failure',
       'has_damage_infill_partition_failure', 'has_damage_staircase',
       'has_damage_parapet', 'has_damage_cladding_glazing',
       'has_geotechnical_risk', 'has_geotechnical_risk_land_settlement',
       'has_geotechnical_risk_fault_crack',
       'has_geotechnical_risk_liquefaction', 'has_geotechnical_risk_landslide',
       'has_geotechnical_risk_rock_fall', 'has_geotechnical_risk_flood',
       'has_geotechnical_risk_other'],
      dtype='object')
```

In [50]:
```
damage_data.head(2)
```

Out[50]:

	building_id	district_id	vdcmun_id	ward_id	damage_overall_collapse	damage_overall_leaning	damage_overall_adjacent_building_risk	damage_foundation_
0	120101000011	12	1207	120703	Moderate-Heavy	Insignificant/light	None	
1	120101000021	12	1207	120703	Severe-Extreme	Severe-Extreme	Insignificant/light	Severe-Extreme

2 rows × 79 columns

In practice 'damage grade' of a property is assesed based on the features in the damage_data, as our goal is to predict the damage grade using structrual data,we will use damage data except damage grade for visualization purpose only.

In [11]:
```
def feature_info(feature):
    print('Unique values in the feature:',feature.unique())
    print('No of Null values:',feature.isna().sum())
```

In [12]:
```
feature_info(damage_data.damage_overall_collapse)
```

```
Unique values in the feature: ['Moderate-Heavy' 'Severe-Extreme' 'Insignificant/light' nan 'None']
No of Null values: 261363
```

In [8]:
```
feature_info(damage_data.technical_solution_proposed)
```

```
Unique values in the feature: ['Major repair' 'Reconstruction' 'Minor repair' 'No need' nan]
No of Null values: 12
```

In [9]:
```
feature_info(damage_data.damage_grade)
```

```
Unique values in the feature: ['Grade 3' 'Grade 5' 'Grade 2' 'Grade 1' 'Grade 4' nan]
No of Null values: 12
```

In [13]:
```
y_labels=damage_data['damage_grade']
```

Data Preparation:

In [218]:
```
# https://stackoverflow.com/questions/23668427/pandas-three-way-joining-multiple-dataframes-on-columns
from functools import reduce
df=[structure_data,ownership_data]
raw_data=reduce(lambda left,right:pd.merge(left,right,on='building_id',how='inner'),df)
```

In [219]:
```
raw_data.shape
```

Out[219]:
```
(762106, 45)
```

In [220]:
```
raw_data.head(5)
```

Out[220]:

	building_id	district_id_x	vdcmun_id_x	ward_id_x	count_floors_pre_eq	count_floors_post_eq	age_building	plinth_area_sq_ft	height_ft_pre_eq	height_ft
0	120101000011	12	1207	120703	1	1	9	288	9	
1	120101000021	12	1207	120703	1	1	15	364	9	
2	120101000031	12	1207	120703	1	1	20	384	9	
3	120101000041	12	1207	120703	1	1	20	312	9	
4	120101000051	12	1207	120703	1	1	30	308	9	

5 rows × 45 columns

In [221]:
```
raw_data=raw_data.rename(index=str,columns={'district_id_x':'geo1','vdcmun_id_x':'geo2','ward_id_x':'geo3'})
```

In [222]:
```
raw_data.columns
```

Out[222]:
```
Index(['building_id', 'geo1', 'geo2', 'geo3', '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',
       'damage_grade', 'district_id_y', 'vdcmun_id_y', 'ward_id_y',
       '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'],
      dtype='object')
```

In [223]:
```
cols_to_drop=['district_id_y', 'vdcmun_id_y', 'ward_id_y']
```

In [224]:
```
raw_data=raw_data.drop(cols_to_drop,axis=1)
```

In [225]:
```
raw_data.isna().sum()
```

Out[225]:
```
building_id                                0
geo1                                       0
geo2                                       0
geo3                                       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                                   1
plan_configuration                         1
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
damage_grade                              12
legal_ownership_status                     0
count_families                             2
has_secondary_use                         10
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
dtype: int64
```

Only small number of data points contains null value.Hence we will data points that have Null values.

In [227]:
```
final_data=raw_data.dropna(axis=0)
final_data.isna().sum()
```

Out[227]:
```
building_id                                0
geo1                                       0
geo2                                       0
geo3                                       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
damage_grade                               0
legal_ownership_status                     0
count_families                             0
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
dtype: int64
```

In [189]:
```
feature_info(raw_data.damage_grade)
```

```
Unique values in the feature: ['Grade 3' 'Grade 5' 'Grade 2' 'Grade 1' 'Grade 4' nan]
No of Null values: 12
```

In [228]:
```
print('Shape of final raw data matrix:',final_data.shape)
print('No of labels:',len(y_labels))
feature_info(final_data.damage_grade)
```

```
Shape of final raw data matrix: (762093, 42)
No of labels: 762093
Unique values in the feature: ['Grade 3' 'Grade 5' 'Grade 2' 'Grade 1' 'Grade 4']
No of Null values: 0
```

In [229]:
```
feature_info(final_data.land_surface_condition)
```

```
Unique values in the feature: ['Flat' 'Moderate slope' 'Steep slope']
No of Null values: 0
```

In [230]:
```
y_labels=final_data.damage_grade
```

In [231]:
```
final_data.columns
```

Out[231]:
```
Index(['building_id', 'geo1', 'geo2', 'geo3', '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',
       'damage_grade', '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'],
      dtype='object')
```

Preprocessing Categorical variables:

In [232]:
```
feature_info(final_data.land_surface_condition)
```

```
Unique values in the feature: ['Flat' 'Moderate slope' 'Steep slope']
No of Null values: 0
```

In [233]:
```
final_data.land_surface_condition=final_data.land_surface_condition.map({'Flat':'Flat','Moderate slope':'Modarate_slope','Steep s
```

In [234]:
```
feature_info(final_data.foundation_type)
```

```
Unique values in the feature: ['Other' 'Mud mortar-Stone/Brick' 'Cement-Stone/Brick' 'Bamboo/Timber'
 'RC']
No of Null values: 0
```

In [235]:
```
feature_info(final_data.roof_type)
```

```
Unique values in the feature: ['Bamboo/Timber-Light roof' 'Bamboo/Timber-Heavy roof' 'RCC/RB/RBC']
No of Null values: 0
```

In [236]:
```
feature_info(final_data.ground_floor_type)
```

```
Unique values in the feature: ['Mud' 'Brick/Stone' 'RC' 'Timber' 'Other']
No of Null values: 0
```

In [237]:
```
feature_info(final_data.other_floor_type)
```

```
Unique values in the feature: ['Not applicable' 'TImber/Bamboo-Mud' 'Timber-Planck' 'RCC/RB/RBC']
No of Null values: 0
```

In [238]:
```
feature_info(final_data.legal_ownership_status)
```

```
Unique values in the feature: ['Private' 'Other' 'Institutional' 'Public']
No of Null values: 0
```

In [239]:
```
feature_info(final_data.position)
```

```
Unique values in the feature: ['Not attached' 'Attached-1 side' 'Attached-2 side' 'Attached-3 side']
No of Null values: 0
```

In [240]:
```
feature_info(final_data.plan_configuration)
```

```
Unique values in the feature: ['Rectangular' 'L-shape' 'Square' 'T-shape' 'Multi-projected' 'H-shape'
 'U-shape' 'Others' 'E-shape' 'Building with Central Courtyard']
No of Null values: 0
```

In [241]:
```
feature_info(final_data.legal_ownership_status)
```

```
Unique values in the feature: ['Private' 'Other' 'Institutional' 'Public']
No of Null values: 0
```

In [242]:
```
final_data.foundation_type=final_data.foundation_type.map({'Mud mortar-Stone/Brick':'Mud_stone_OR_brick','Cement-Stone/Brick':'Ce
final_data.roof_type=final_data.roof_type.map({'Bamboo/Timber-Light roof':'wood_light','Bamboo/Timber-Heavy roof':'wood_heavy','R
final_data.ground_floor_type=final_data.ground_floor_type.map({'Brick/Stone':'Brick_or_Stone','Mud':'Mud','RC':'RC','Timber':'Tim
final_data.other_floor_type=final_data.other_floor_type.map({'Not applicable':'NotApplicable','TImber/Bamboo-Mud':'wood_mud','Tim
final_data.position=final_data.position.map({'Not attached':'Not_attached','Attached-1 side':'Attached_1_side','Attached-2 side':
final_data.plan_configuration=final_data.plan_configuration.map(
{
 'Rectangular':'Rectangular','L-shape':'L_Shape','Square':'Square','T-shape':'T_Shape','Multi-projected':'Multi_projected',
    'H-shape':'H_Shape','U-shape':'U_Shape','Others':'Others','E-shape':'E_Shape','Building with Central Courtyard':'Central_Cour

})
```

In [243]:
```
cat_var=['land_surface_condition','foundation_type','roof_type','ground_floor_type','other_floor_type','legal_ownership_status'
 ,'position','plan_configuration','legal_ownership_status']
print('Categorical Feature Check after Preprocessing:')
print('-----------------------------------------------')
for cat in cat_var:
```

```
for cat in cat_var:
    print('Feature Name:',cat)
    feature_info(final_data[cat])
    print('----------------------------------------')
```

```
Categorical Feature Check after Preprocessing:
-----------------------------------------------
Feature Name: land_surface_condition
Unique values in the feature: ['Flat' 'Modarate_slope' 'steep_slope']
No of Null values: 0
---------------------------------------
Feature Name: foundation_type
Unique values in the feature: ['Other' 'Mud_stone_OR_brick' 'Cement_stone_OR_brick' 'Bamboo_OR_Timber'
 'RC']
No of Null values: 0
---------------------------------------
Feature Name: roof_type
Unique values in the feature: ['wood_light' 'wood_heavy' 'RCC_RB_RBC']
No of Null values: 0
---------------------------------------
Feature Name: ground_floor_type
Unique values in the feature: ['Mud' 'Brick_or_Stone' 'RC' 'Timber' 'Other']
No of Null values: 0
---------------------------------------
Feature Name: other_floor_type
Unique values in the feature: ['NotApplicable' 'wood_mud' 'Timber_Planck' 'RCC_RB_RBC']
No of Null values: 0
---------------------------------------
Feature Name: legal_ownership_status
Unique values in the feature: ['Private' 'Other' 'Institutional' 'Public']
No of Null values: 0
---------------------------------------
Feature Name: position
Unique values in the feature: ['Not_attached' 'Attached_1_side' 'Attached_2_side' 'Attached_3_side']
No of Null values: 0
---------------------------------------
Feature Name: plan_configuration
Unique values in the feature: ['Rectangular' 'L_Shape' 'Square' 'T_Shape' 'Multi_projected' 'H_Shape'
 'U_Shape' 'Others' 'E_Shape' 'Central_Courtyard']
No of Null values: 0
---------------------------------------
Feature Name: legal_ownership_status
Unique values in the feature: ['Private' 'Other' 'Institutional' 'Public']
No of Null values: 0
---------------------------------------
```

Preparing Train and Test Data:

In [244]:
```
train_data,test_data,y_train,y_test=train_test_split(final_data,y_labels,stratify=y_labels,test_size=0.2)
```

In [245]:
```
print('Shape of train data:',train_data.shape)
print('Shape of test data:',test_data.shape)
```

```
Shape of train data: (609674, 42)
Shape of test data: (152419, 42)
```

In [246]:
```
feature_info(train_data['damage_grade'])
feature_info(test_data['damage_grade'])
```

```
Unique values in the feature: ['Grade 2' 'Grade 4' 'Grade 3' 'Grade 5' 'Grade 1']
No of Null values: 0
Unique values in the feature: ['Grade 4' 'Grade 2' 'Grade 5' 'Grade 1' 'Grade 3']
No of Null values: 0
```

In [247]:
```
# We will convert class labels in to 3 classes,if damage grade 1-2:Low,3:Medium,4-5:High
# We will Convert class label to numeric values where 1:low,2:medium,3:high in final_data
train_data.damage_grade=train_data.damage_grade.map({'Grade 5':3,'Grade 4':3,'Grade 1':1, 'Grade 3':2, 'Grade 2':1})
test_data.damage_grade=test_data.damage_grade.map({'Grade 5':3,'Grade 4':3,'Grade 1':1, 'Grade 3':2, 'Grade 2':1})
```

In [248]:
```
feature_info(train_data['damage_grade'])
feature_info(test_data['damage_grade'])
```

```
Unique values in the feature: [1 3 2]
No of Null values: 0
Unique values in the feature: [3 1 2]
No of Null values: 0
```

In [249]:
```
y_train=train_data.damage_grade
y_test=test_data.damage_grade
```

In [250]:
```
# final check for duplicate values
print('No of duplicates in train: {}'.format(sum(train_data.duplicated(['building_id']))))
print('No of duplicates in test : {}'.format(sum(test_data.duplicated())))
```

```
No of duplicates in train: 0
No of duplicates in test : 0
```

Export Data:

In [251]:
```
train_labels=pd.DataFrame({'Labels':y_train})
test_labels=pd.DataFrame({'Labels':y_test})
print('Export Start....')
train_data.to_csv('train_values.csv',index=False)
test_data.to_csv('test_values.csv',index=False)
train_labels.to_csv('train_labels.csv',index=False)
test_labels.to_csv('test_labels.csv',index=False)
print('Export End.')
```

```
Export Start....
Export End.
```