In [48]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Loading dataset and merging into one

In [49]:
```
train_labels = pd.read_csv('train_labels.csv')
train_values = pd.read_csv('train_values.csv')

merged_data = pd.merge(train_values, train_labels, on='building_id')
```

In [50]:
```
train_labels.head(5)
```

Out[50]:

	building_id	damage_grade
0	802906	3
1	28830	2
2	94947	3
3	590882	2
4	201944	3

In [51]:
```
train_values.head(5)
```

Out[51]:

	building_id	geo_level_1_id	geo_level_2_id	geo_level_3_id	count_floors_pre_eq	age	are
0	802906	6	487	12198	2	30	
1	28830	8	900	2812	2	10	
2	94947	21	363	8973	2	10	
3	590882	22	418	10694	2	10	
4	201944	11	131	1488	3	30	

5 rows × 39 columns

In [52]:
```
merged_data.head()
```

Out[52]:

	building_id	geo_level_1_id	geo_level_2_id	geo_level_3_id	count_floors_pre_eq	age	are
0	802906	6	487	12198	2	30	
1	28830	8	900	2812	2	10	
2	94947	21	363	8973	2	10	
3	590882	22	418	10694	2	10	
4	201944	11	131	1488	3	30	

5 rows × 40 columns

In [54]:
```
merged_data.tail()
```

Out[54]:

	building_id	geo_level_1_id	geo_level_2_id	geo_level_3_id	count_floors_pre_eq	age
260596	688636	25	1335	1621	1	55
260597	669485	17	715	2060	2	0
260598	602512	17	51	8163	3	55
260599	151409	26	39	1851	2	10
260600	747594	21	9	9101	3	10

5 rows × 40 columns

In [56]:
```
merged_data.shape
```

Out[56]:
```
(260601, 40)
```

In [58]:
```
merged_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 260601 entries, 0 to 260600
Data columns (total 40 columns):
 #   Column                                  Non-Null Count   Dtype
---  ------                                  --------------   -----
 0   building_id                             260601 non-null  int64
 1   geo_level_1_id                          260601 non-null  int64
 2   geo_level_2_id                          260601 non-null  int64
 3   geo_level_3_id                          260601 non-null  int64
 4   count_floors_pre_eq                     260601 non-null  int64
 5   age                                     260601 non-null  int64
 6   area_percentage                         260601 non-null  int64
 7   height_percentage                       260601 non-null  int64
 8   land_surface_condition                  260601 non-null  object
 9   foundation_type                         260601 non-null  object
 10  roof_type                               260601 non-null  object
 11  ground_floor_type                       260601 non-null  object
 12  other_floor_type                        260601 non-null  object
 13  position                                260601 non-null  object
 14  plan_configuration                      260601 non-null  object
 15  has_superstructure_adobe_mud            260601 non-null  int64
 16  has_superstructure_mud_mortar_stone     260601 non-null  int64
 17  has_superstructure_stone_flag           260601 non-null  int64
 18  has_superstructure_cement_mortar_stone  260601 non-null  int64
 19  has_superstructure_mud_mortar_brick     260601 non-null  int64
 20  has_superstructure_cement_mortar_brick  260601 non-null  int64
 21  has_superstructure_timber               260601 non-null  int64
 22  has_superstructure_bamboo               260601 non-null  int64
 23  has_superstructure_rc_non_engineered    260601 non-null  int64
 24  has_superstructure_rc_engineered        260601 non-null  int64
 25  has_superstructure_other                260601 non-null  int64
 26  legal_ownership_status                  260601 non-null  object
 27  count_families                          260601 non-null  int64
 28  has_secondary_use                       260601 non-null  int64
 29  has_secondary_use_agriculture           260601 non-null  int64
 30  has_secondary_use_hotel                 260601 non-null  int64
 31  has_secondary_use_rental                260601 non-null  int64
 32  has_secondary_use_institution           260601 non-null  int64
 33  has_secondary_use_school                260601 non-null  int64
 34  has_secondary_use_industry              260601 non-null  int64
 35  has_secondary_use_health_post           260601 non-null  int64
 36  has_secondary_use_gov_office            260601 non-null  int64
 37  has_secondary_use_use_police            260601 non-null  int64
 38  has_secondary_use_other                 260601 non-null  int64
 39  damage_grade                            260601 non-null  int64
dtypes: int64(32), object(8)
memory usage: 79.5+ MB
```

In [60]:
```
merged_data.describe().T.style.background_gradient(cmap = 'Set3')
```

Out[60]:

	count	mean	std	
building_id	260601.000000	525675.482773	304544.999032	4.000
geo_level_1_id	260601.000000	13.900353	8.033617	0.000
geo_level_2_id	260601.000000	701.074685	412.710734	0.000
geo_level_3_id	260601.000000	6257.876148	3646.369645	0.000
count_floors_pre_eq	260601.000000	2.129723	0.727665	1.000
age	260601.000000	26.535029	73.565937	0.000
area_percentage	260601.000000	8.018051	4.392231	1.000
height_percentage	260601.000000	5.434365	1.918418	2.000
has_superstructure_adobe_mud	260601.000000	0.088645	0.284231	0.000
has_superstructure_mud_mortar_stone	260601.000000	0.761935	0.425900	0.000
has_superstructure_stone_flag	260601.000000	0.034332	0.182081	0.000
has_superstructure_cement_mortar_stone	260601.000000	0.018235	0.133800	0.000
has_superstructure_mud_mortar_brick	260601.000000	0.068154	0.252010	0.000
has_superstructure_cement_mortar_brick	260601.000000	0.075268	0.263824	0.000
has_superstructure_timber	260601.000000	0.254988	0.435855	0.000
has_superstructure_bamboo	260601.000000	0.085011	0.278899	0.000
has_superstructure_rc_non_engineered	260601.000000	0.042590	0.201931	0.000
has_superstructure_rc_engineered	260601.000000	0.015859	0.124932	0.000
has_superstructure_other	260601.000000	0.014985	0.121491	0.000
count_families	260601.000000	0.983949	0.418389	0.000
has_secondary_use	260601.000000	0.111880	0.315219	0.000
has_secondary_use_agriculture	260601.000000	0.064378	0.245426	0.000
has_secondary_use_hotel	260601.000000	0.033626	0.180265	0.000
has_secondary_use_rental	260601.000000	0.008101	0.089638	0.000
has_secondary_use_institution	260601.000000	0.000940	0.030647	0.000
has_secondary_use_school	260601.000000	0.000361	0.018989	0.000
has_secondary_use_industry	260601.000000	0.001071	0.032703	0.000
has_secondary_use_health_post	260601.000000	0.000188	0.013711	0.000
has_secondary_use_gov_office	260601.000000	0.000146	0.012075	0.000
has_secondary_use_use_police	260601.000000	0.000088	0.009394	0.000

	count	mean	std	
has_secondary_use_other	260601.000000	0.005119	0.071364	0.000
damage_grade	260601.000000	2.238272	0.611814	1.000

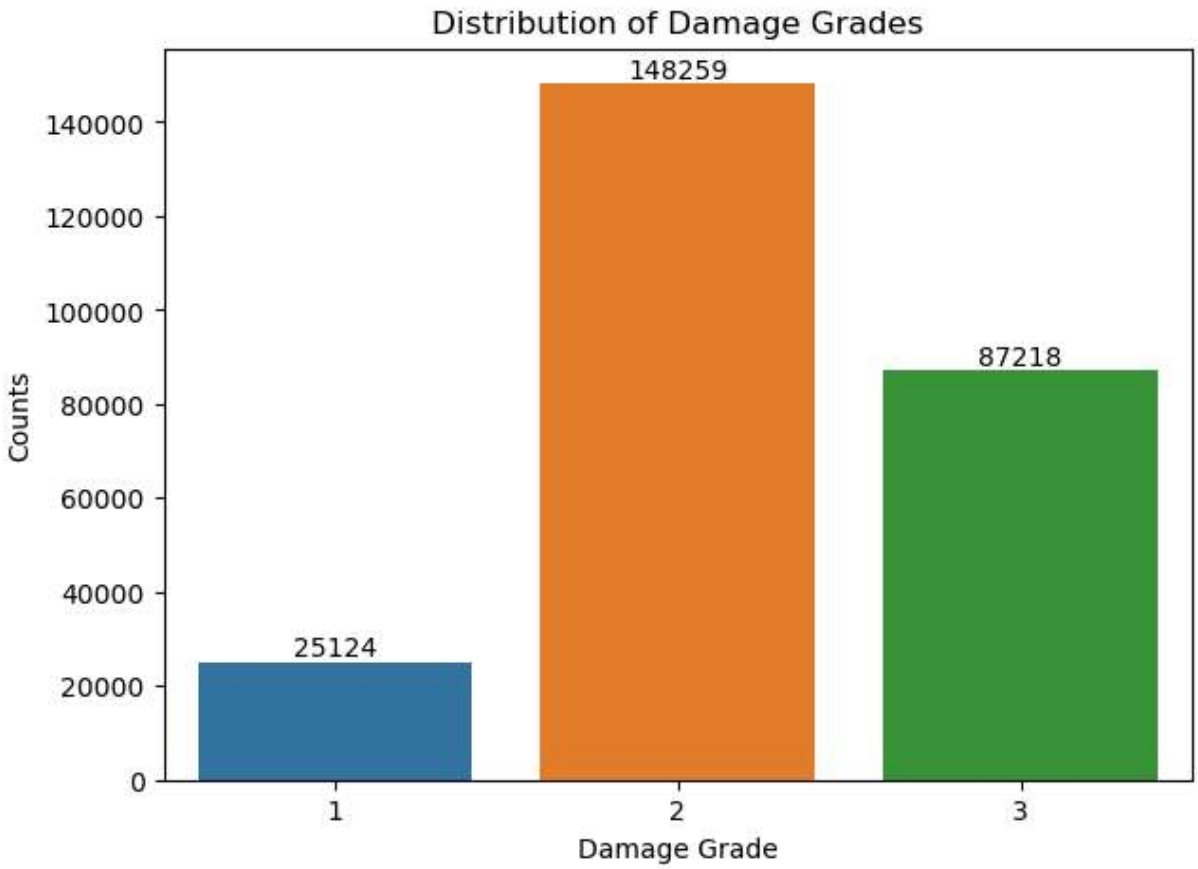
In [62]:
```
merged_data.describe(include = 'O').T
```

Out[62]:

	count	unique	top	freq
land_surface_condition	260601	3	t	216757
foundation_type	260601	5	r	219196
roof_type	260601	3	n	182842
ground_floor_type	260601	5	f	209619
other_floor_type	260601	4	q	165282
position	260601	4	s	202090
plan_configuration	260601	10	d	250072
legal_ownership_status	260601	4	v	250939

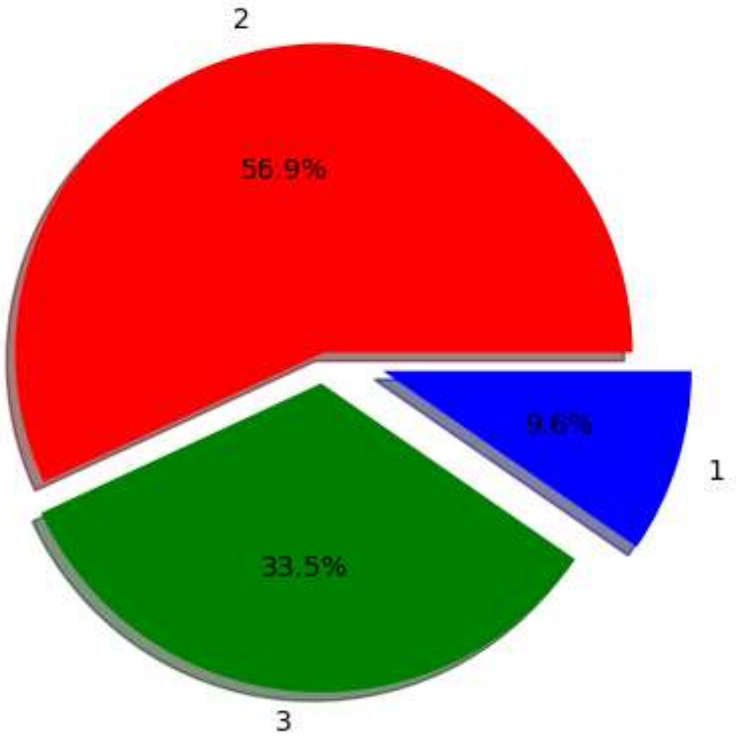
Count-Plot

In [64]:
```
plt.figure(figsize=(7, 5))

ax=sns.countplot(data=merged_data, x='damage_grade')
ax.bar_label(ax.containers[0])

plt.xlabel('Damage Grade')
plt.ylabel('Counts')
plt.title('Distribution of Damage Grades')

plt.show()
```



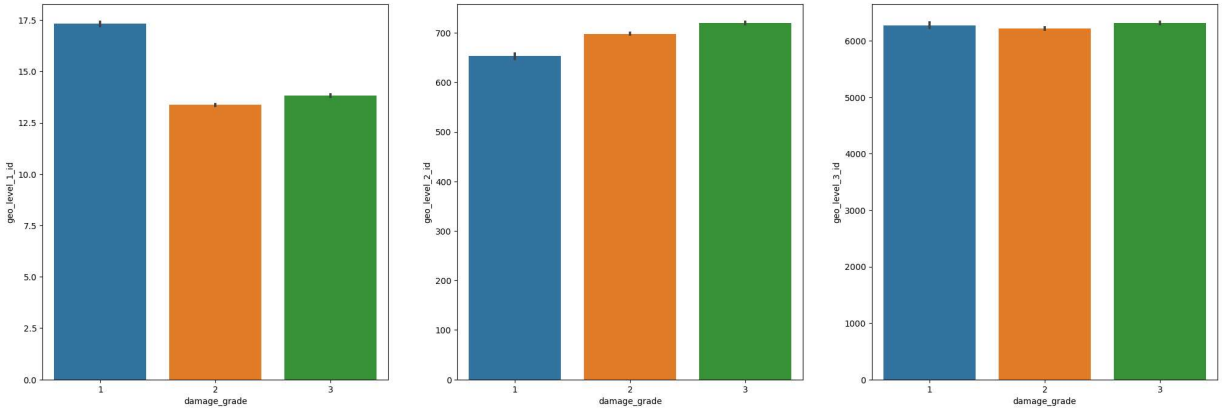
Pie chart

In [66]:
```
value_counts = merged_data['damage_grade'].value_counts()

plt.figure(figsize=(7, 5))
plt.pie(value_counts, labels=value_counts.index, autopct='%.1f%%', explode=(0, 0.1,
plt.show()
```



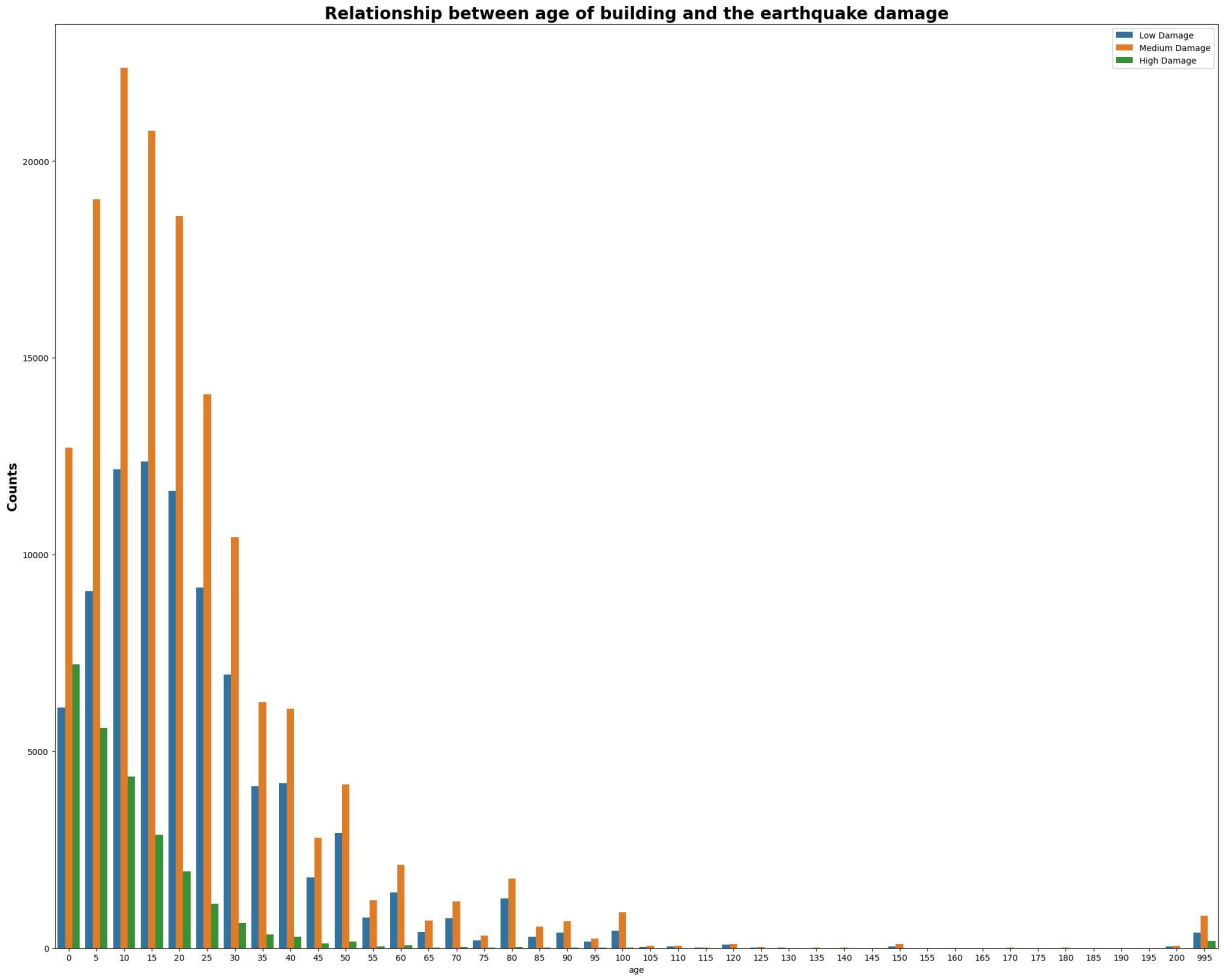
Bar Graph

In [68]:
```
plt.figure(figsize=(25, 8))

plt.subplot(1, 3, 1)
sns.barplot(data=merged_data, x='damage_grade', y='geo_level_1_id')

plt.subplot(1, 3, 2)
sns.barplot(data=merged_data, x='damage_grade', y='geo_level_2_id')

plt.subplot(1, 3, 3)
sns.barplot(data=merged_data, x='damage_grade', y='geo_level_3_id')

plt.show()
```



Countplot

In [70]:
```
plt.figure(figsize=(25, 20))
sns.countplot(x = merged_data['age'], hue = merged_data['damage_grade'].astype(str)
plt.title('Relationship between age of building and the earthquake damage',fontweig
plt.ylabel('Counts',fontsize = 15, fontweight = 'bold')
plt.legend(['Low Damage', 'Medium Damage', 'High Damage'], loc = 'upper right')
plt.show()
```



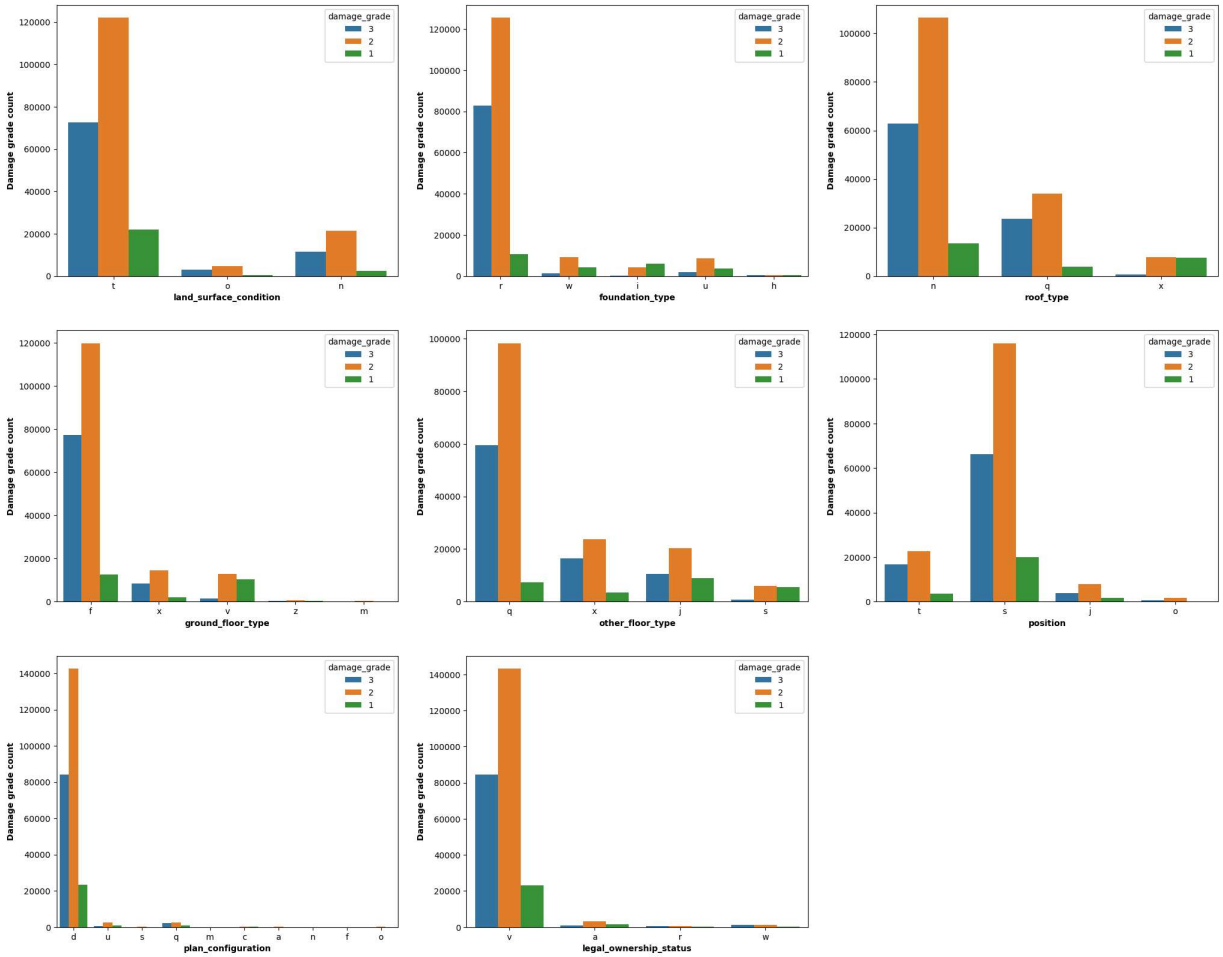
In [72]:
```
cat_features = []
for columns in merged_data.columns:
    if merged_data[columns].dtype=='O':
        cat_features.append(columns)
```

In [74]:
```
cat_features
```

Out[74]:
```
['land_surface_condition',
 'foundation_type',
 'roof_type',
 'ground_floor_type',
 'other_floor_type',
 'position',
 'plan_configuration',
 'legal_ownership_status']
```

In [76]:
```
cat_features_df = pd.DataFrame({col: merged_data[col] for col in cat_features})
```

In [78]:
```
plt.figure(figsize = (25,20))
plotnumber = 1

for column in cat_features_df:
    if plotnumber<=8:
        ax = plt.subplot(3,3,plotnumber)
        sns.countplot(x = cat_features_df[column], hue = merged_data['damage_grade'
        plt.xlabel(column,fontweight = 'bold')
        plt.ylabel('Damage grade count',fontweight = 'bold')
    plotnumber+=1

plt.show()
```



In [80]:
```
bin_features = []
for column in merged_data.columns:
    if set(merged_data[column].unique())=={0,1}:
        bin_features.append(column)
```

In [82]:
```
bin_features
```

Out[82]:
```
['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_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']
```

In [84]:
```
bin_features_df = pd.DataFrame({col: merged_data[col] for col in bin_features})
bin_features_df
```

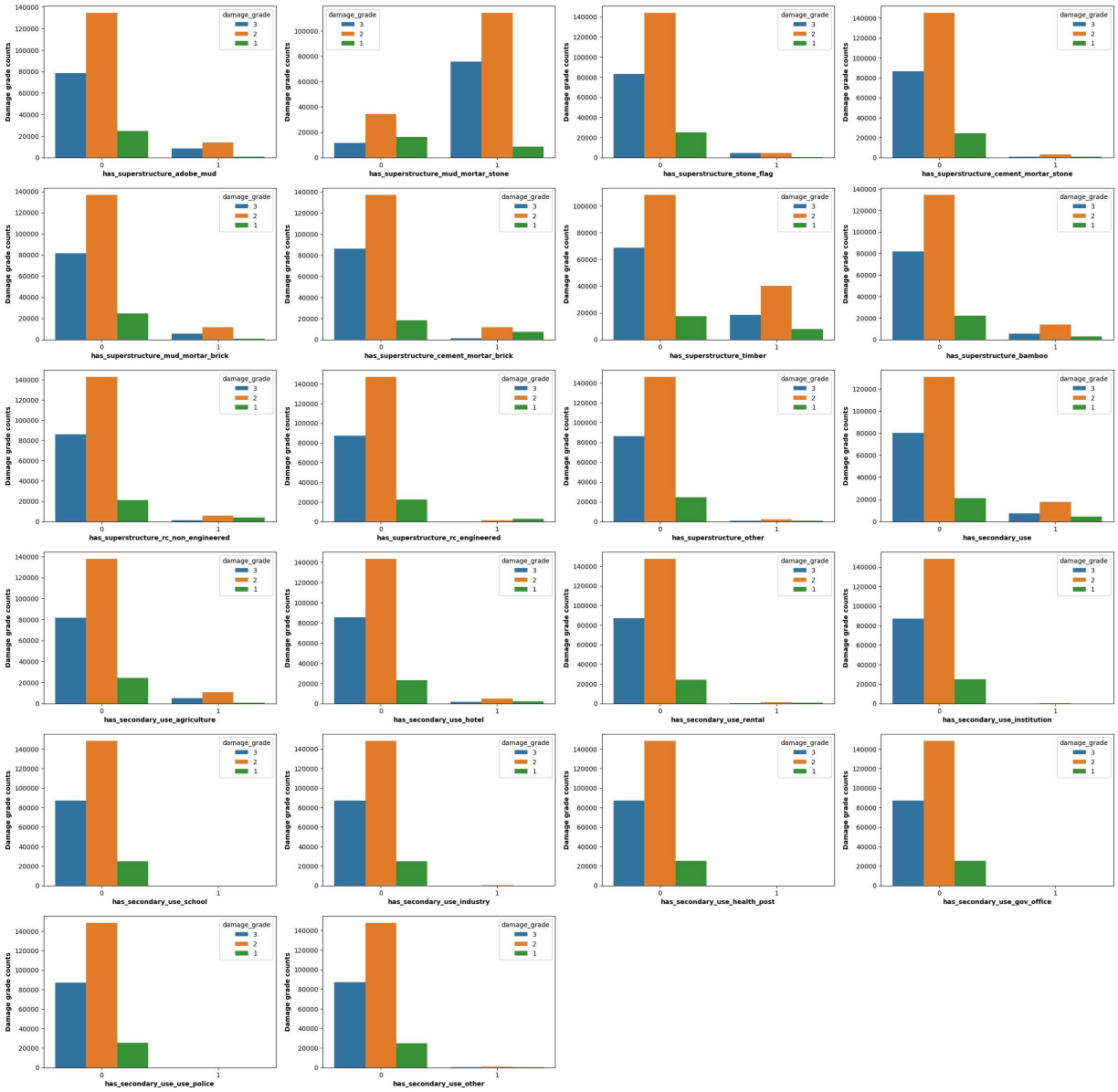
Out[84]:

	has_superstructure_adobe_mud	has_superstructure_mud_mortar_stone	has_superstru
0	1	1	
1	0	1	
2	0	1	
3	0	1	
4	1	0	
...	...	...	
260596	0	1	
260597	0	1	
260598	0	1	
260599	0	0	
260600	0	1	

260601 rows × 22 columns

In [86]:
```
plt.figure(figsize = (30,30))
plotnumber = 1

for column in bin_features_df:
```

```
    if plotnumber <= 22:
        ax = plt.subplot(6,4,plotnumber)
        sns.countplot(x = bin_features_df[column], hue = merged_data['damage_grade'
        plt.xlabel(column, fontweight = 'bold')
        plt.ylabel('Damage grade counts', fontweight = 'bold')
    plotnumber+=1
plt.show()
```



In [88]:
```
con_features = []
for column in merged_data:
    if (merged_data[column].dtype!= 'O' and
        column not in bin_features and
        column!='damage_grade' and
        column!='building_id') :
        con_features.append(column)
```

In [90]:
```
con_features
```

Out[90]:
```
['geo_level_1_id',
 'geo_level_2_id',
 'geo_level_3_id',
 'count_floors_pre_eq',
 'age',
 'area_percentage',
 'height_percentage',
 'count_families']
```

In [92]:
```
con_features_df = pd.DataFrame({col: merged_data[col] for col in con_features})
con_features_df
```

Out[92]:

	geo_level_1_id	geo_level_2_id	geo_level_3_id	count_floors_pre_eq	age	area_perce
0	6	487	12198	2	30	
1	8	900	2812	2	10	
2	21	363	8973	2	10	
3	22	418	10694	2	10	
4	11	131	1488	3	30	
...	...	...	...	...	...	
260596	25	1335	1621	1	55	
260597	17	715	2060	2	0	
260598	17	51	8163	3	55	
260599	26	39	1851	2	10	
260600	21	9	9101	3	10	

260601 rows × 8 columns

In [94]:
```
merged_data.isnull().sum()
```

Out[94]:
```
building_id                             0
geo_level_1_id                          0
geo_level_2_id                          0
geo_level_3_id                          0
count_floors_pre_eq                     0
age                                     0
area_percentage                         0
height_percentage                       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
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
damage_grade                            0
dtype: int64
```

In [96]:
```
from sklearn.preprocessing import LabelEncoder
lc = LabelEncoder()

merged_data['land_surface_condition'] = lc.fit_transform(merged_data['land_surface_
merged_data['foundation_type'] = lc.fit_transform(merged_data['foundation_type'])
merged_data['roof_type'] = lc.fit_transform(merged_data['roof_type'])
merged_data['ground_floor_type'] = lc.fit_transform(merged_data['ground_floor_type'
merged_data['other_floor_type'] = lc.fit_transform(merged_data['other_floor_type'])
merged_data['position'] = lc.fit_transform(merged_data['position'])
merged_data['plan_configuration'] = lc.fit_transform(merged_data['plan_configuratio
merged_data['legal_ownership_status'] = lc.fit_transform(merged_data['legal_ownersh
```

In [98]:
```
merged_data
```

Out[98]:

	building_id	geo_level_1_id	geo_level_2_id	geo_level_3_id	count_floors_pre_eq	age
0	802906	6	487	12198	2	30
1	28830	8	900	2812	2	10
2	94947	21	363	8973	2	10
3	590882	22	418	10694	2	10
4	201944	11	131	1488	3	30
...	...	...	...	...	...	...
260596	688636	25	1335	1621	1	55
260597	669485	17	715	2060	2	0
260598	602512	17	51	8163	3	55
260599	151409	26	39	1851	2	10
260600	747594	21	9	9101	3	10

260601 rows × 40 columns

In [100…

```
merged_data.describe().T
```

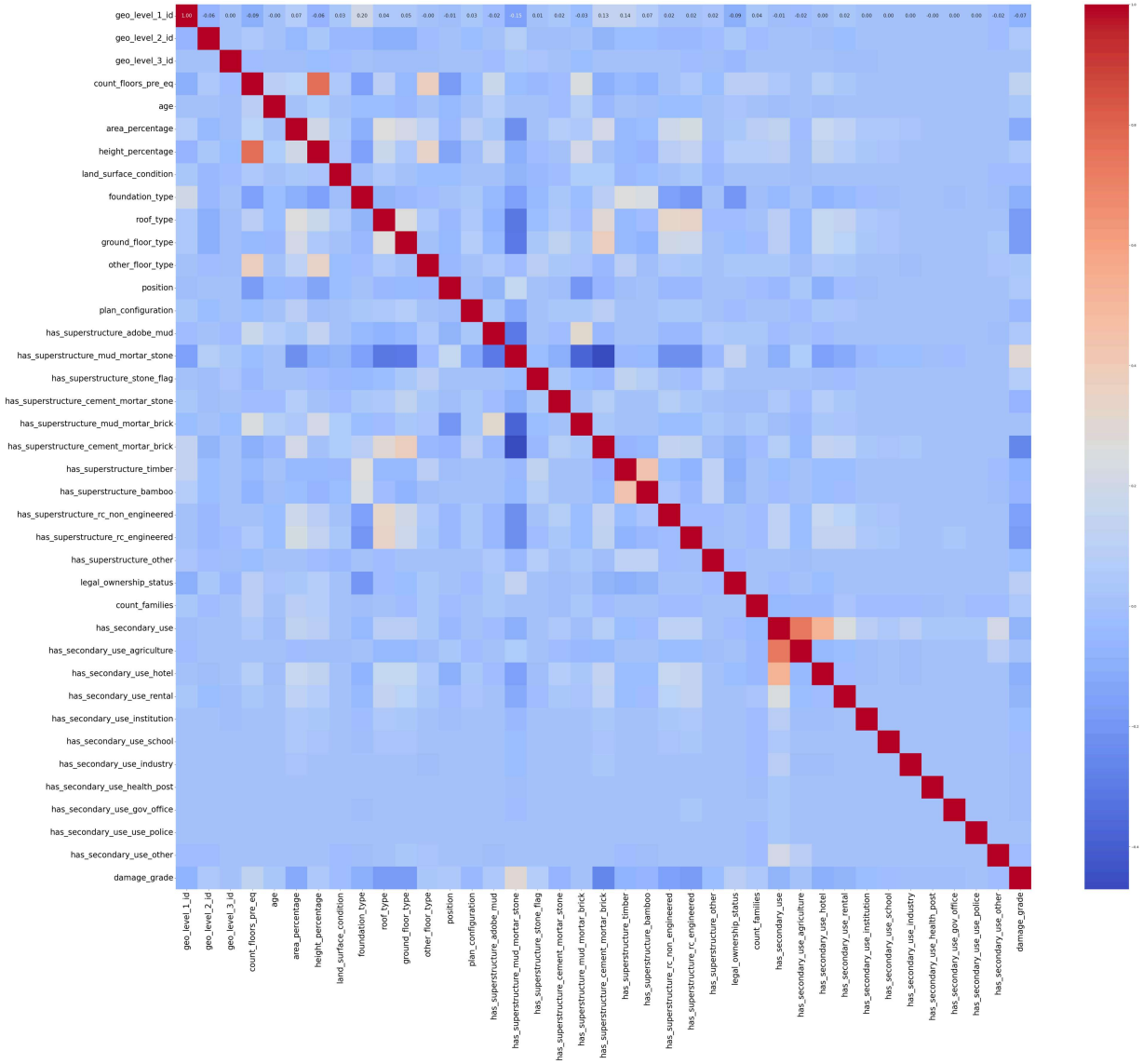
Out[100…

	count	mean	std	min	
building_id	260601.0	525675.482773	304544.999032	4.0	2611
geo_level_1_id	260601.0	13.900353	8.033617	0.0	
geo_level_2_id	260601.0	701.074685	412.710734	0.0	3
geo_level_3_id	260601.0	6257.876148	3646.369645	0.0	3(
count_floors_pre_eq	260601.0	2.129723	0.727665	1.0	
age	260601.0	26.535029	73.565937	0.0	
area_percentage	260601.0	8.018051	4.392231	1.0	
height_percentage	260601.0	5.434365	1.918418	2.0	
land_surface_condition	260601.0	1.695427	0.696040	0.0	
foundation_type	260601.0	2.119036	0.579154	0.0	
roof_type	260601.0	0.360482	0.595595	0.0	
ground_floor_type	260601.0	0.492481	1.028436	0.0	
other_floor_type	260601.0	1.226711	0.902518	0.0	
position	260601.0	2.053718	0.611996	0.0	
plan_configuration	260601.0	2.215325	1.118290	0.0	
has_superstructure_adobe_mud	260601.0	0.088645	0.284231	0.0	
has_superstructure_mud_mortar_stone	260601.0	0.761935	0.425900	0.0	
has_superstructure_stone_flag	260601.0	0.034332	0.182081	0.0	
has_superstructure_cement_mortar_stone	260601.0	0.018235	0.133800	0.0	
has_superstructure_mud_mortar_brick	260601.0	0.068154	0.252010	0.0	
has_superstructure_cement_mortar_brick	260601.0	0.075268	0.263824	0.0	
has_superstructure_timber	260601.0	0.254988	0.435855	0.0	
has_superstructure_bamboo	260601.0	0.085011	0.278899	0.0	
has_superstructure_rc_non_engineered	260601.0	0.042590	0.201931	0.0	
has_superstructure_rc_engineered	260601.0	0.015859	0.124932	0.0	
has_superstructure_other	260601.0	0.014985	0.121491	0.0	
legal_ownership_status	260601.0	1.962318	0.314817	0.0	
count_families	260601.0	0.983949	0.418389	0.0	
has_secondary_use	260601.0	0.111880	0.315219	0.0	
has_secondary_use_agriculture	260601.0	0.064378	0.245426	0.0	

	count	mean	std	min
has_secondary_use_hotel	260601.0	0.033626	0.180265	0.0
has_secondary_use_rental	260601.0	0.008101	0.089638	0.0
has_secondary_use_institution	260601.0	0.000940	0.030647	0.0
has_secondary_use_school	260601.0	0.000361	0.018989	0.0
has_secondary_use_industry	260601.0	0.001071	0.032703	0.0
has_secondary_use_health_post	260601.0	0.000188	0.013711	0.0
has_secondary_use_gov_office	260601.0	0.000146	0.012075	0.0
has_secondary_use_use_police	260601.0	0.000088	0.009394	0.0
has_secondary_use_other	260601.0	0.005119	0.071364	0.0
damage_grade	260601.0	2.238272	0.611814	1.0

In [102…
```
merged_data.drop('building_id', axis =1, inplace=True)
```

In [104…
```
plt.figure(figsize = (60,50))
heatmap = sns.heatmap(merged_data.corr(), annot = True, cmap = 'coolwarm', fmt = ".
heatmap.tick_params(axis='both', labelsize=25)
```



In [106…
```
x = merged_data.drop('damage_grade', axis = 1)
y = merged_data['damage_grade']
```

In [108…
```
x.head()
```

Out[108…

	geo_level_1_id	geo_level_2_id	geo_level_3_id	count_floors_pre_eq	age	area_percentage
0	6	487	12198	2	30	6
1	8	900	2812	2	10	8
2	21	363	8973	2	10	5
3	22	418	10694	2	10	6
4	11	131	1488	3	30	8

5 rows × 38 columns

Model Building

In [110…
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.25, random_state
```

In [112…
```
print('x_train: ',x_train.shape)
print('x_test: ',x_test.shape)
print('y_train: ',y_train.shape)
print('y_test: ',y_test.shape)
```

```
x_train:  (195450, 38)
x_test:  (65151, 38)
y_train:  (195450,)
y_test:  (65151,)
```

In [113…
```
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier


lr = LogisticRegression(multi_class = 'ovr')
kn = KNeighborsClassifier()
dt = DecisionTreeClassifier()
rf = RandomForestClassifier()


lr.fit(x_train,y_train)
kn.fit(x_train,y_train)
dt.fit(x_train,y_train)
rf.fit(x_train,y_train)

y_predlr = lr.predict(x_test)
y_predkn = kn.predict(x_test)
y_preddt = dt.predict(x_test)
y_predrf = rf.predict(x_test)
```

```
C:\Users\lenovo\.conda\envs\ds_env\Lib\site-packages\sklearn\linear_model\_logistic.
py:458: ConvergenceWarning: 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
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

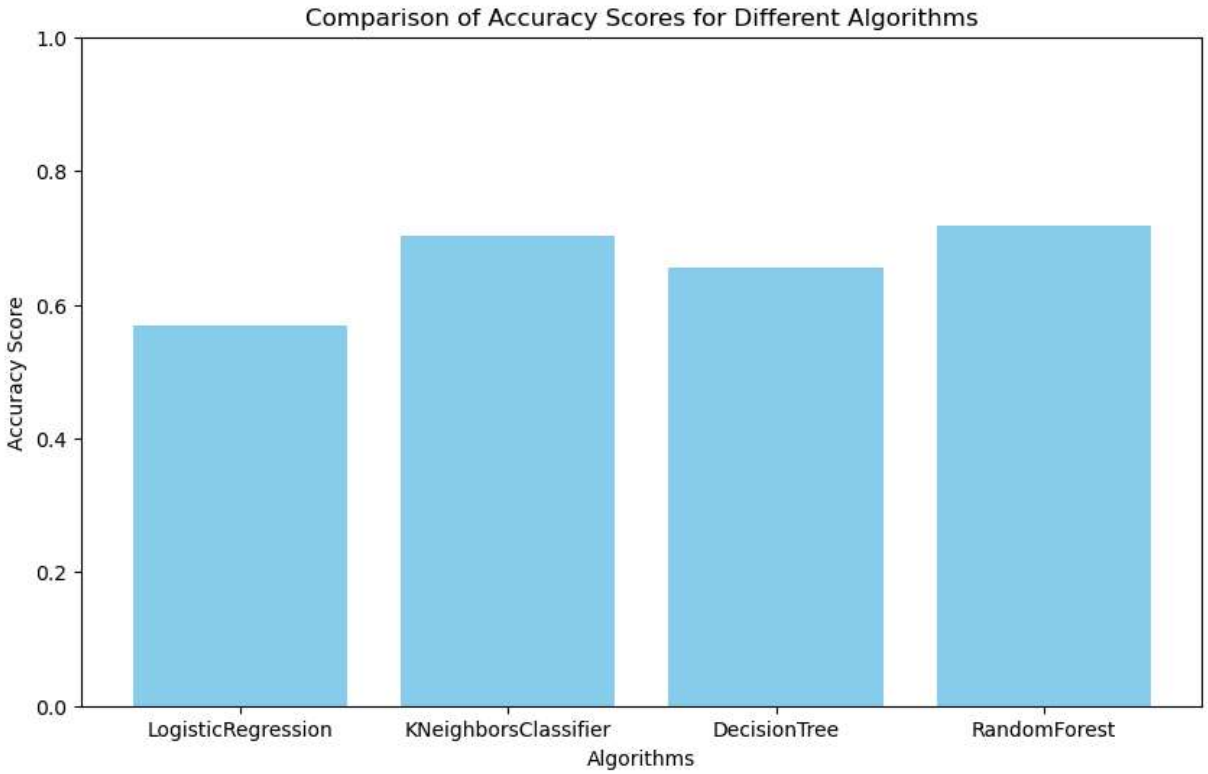
Accuracy score

In [114…
```
from sklearn.metrics import accuracy_score

print('Accuracy score of various models: ')
print('LogisticRegression:            ',accuracy_score(y_test,y_predlr))
print('KNeighborsClassifier:          ',accuracy_score(y_test,y_predkn))
print('DecisionTree:                  ',accuracy_score(y_test,y_preddt))
print('RandomForest:                  ',accuracy_score(y_test,y_predrf))
```

```
Accuracy score of various models:
LogisticRegression:             0.5684026338812912
KNeighborsClassifier:           0.7033660266150942
DecisionTree:                   0.6531749320808583
RandomForest:                   0.7192368497797424
```

In [115…

```
Accuracy_scores = [0.5684026338812912, 0.7033660266150942, 0.6544642446010038, 0.71

algorithms = ['LogisticRegression','KNeighborsClassifier','DecisionTree','RandomFor

plt.figure(figsize=(10, 6))
plt.bar(algorithms, Accuracy_scores, color='skyblue')


plt.xlabel('Algorithms')
plt.ylabel('Accuracy Score')
plt.title('Comparison of Accuracy Scores for Different Algorithms')
plt.ylim(0, 1)
plt.show()
```

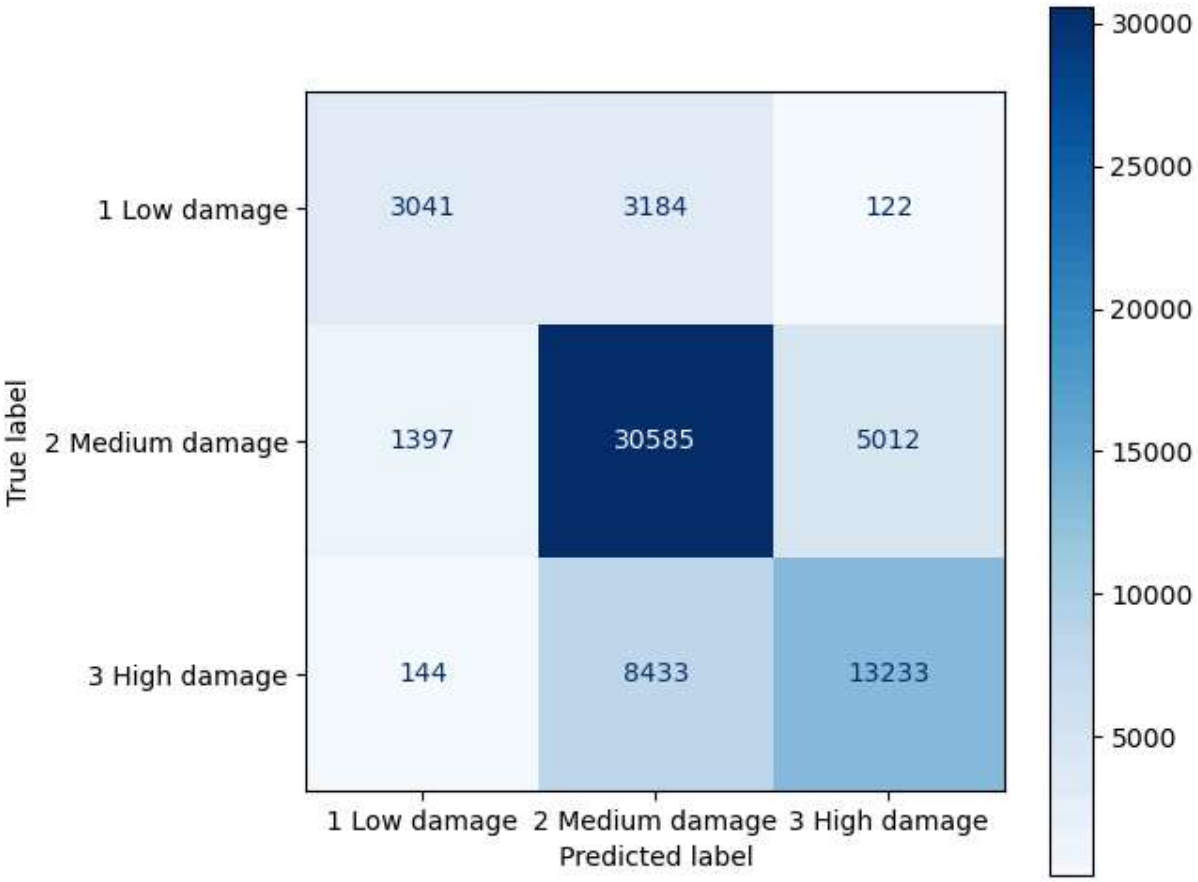


confusion matrix

In [116…

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

cm = confusion_matrix(y_test, y_predrf)
display = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['1 Low damage

fig, axes = plt.subplots(figsize=(6, 6))
display.plot(cmap=plt.cm.Blues, ax=axes)
```

```
plt.show()
```



In [117…
```
y_pred_train_first_50 = rf.predict(x_train[:50])

y_pred_first_50 = rf.predict(x_test[:50])

print("Predicted outputs for the first 50 records of training data:")
print(y_pred_train_first_50)

print("\nPredicted outputs for the first 50 records of testing data:")
print(y_pred_first_50)
```

```
Predicted outputs for the first 50 records of training data:
[2 3 2 2 2 2 3 2 2 1 2 2 3 3 2 2 2 3 2 2 2 2 2 3 2 2 2 2 2 2 1 1 2 2 3 2 1
 1 2 3 2 2 2 2 2 3 2 3 2 2]

Predicted outputs for the first 50 records of testing data:
[3 2 2 2 2 2 3 2 3 3 2 2 2 3 2 2 3 2 2 2 3 2 2 2 2 2 2 2 2 3 2 2 2 3 3 2
 2 2 1 2 2 3 2 2 2 2 2 3 3]
```

In [118…
```
print(x_train.head(1))
```

```
       geo_level_1_id  geo_level_2_id  geo_level_3_id  count_floors_pre_eq  \
66737              12              32            5472                    2

       age  area_percentage  height_percentage  land_surface_condition  \
66737   30               10                  6                       2

       foundation_type  roof_type  ...  has_secondary_use_agriculture  \
66737                2          0  ...                              0

       has_secondary_use_hotel  has_secondary_use_rental  \
66737                        0                        0

       has_secondary_use_institution  has_secondary_use_school  \
66737                              0                         0

       has_secondary_use_industry  has_secondary_use_health_post  \
66737                           0                              0

       has_secondary_use_gov_office  has_secondary_use_use_police  \
66737                             0                             0

       has_secondary_use_other
66737                        0

[1 rows x 38 columns]
```

In [119…
```
merged_data['damage_grade'].value_counts()
```

Out[119…
```
damage_grade
2    148259
3     87218
1     25124
Name: count, dtype: int64
```

In [120…
```
from imblearn.over_sampling import SMOTE
smote = SMOTE()
x_smote,y_smote = smote.fit_resample(x_train,y_train)
```

In [121…
```
lr.fit(x_smote,y_smote)
kn.fit(x_smote,y_smote)
dt.fit(x_smote,y_smote)
rf.fit(x_smote,y_smote)


y_smote_predlr = lr.predict(x_test)
y_smote_predkn = kn.predict(x_test)
y_smote_preddt = dt.predict(x_test)
y_smote_predrf = rf.predict(x_test)
```

```
C:\Users\lenovo\.conda\envs\ds_env\Lib\site-packages\sklearn\linear_model\_logistic.
py:458: ConvergenceWarning: 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
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

In [122…

```
from sklearn.metrics import accuracy_score

print('Accuracy score of various models: ')
print('LogisticRegression:            ',accuracy_score(y_test,y_smote_predlr))
print('KNeighborsClassifier:          ',accuracy_score(y_test,y_smote_predkn))
print('DecisionTree:                  ',accuracy_score(y_test,y_smote_preddt))
print('RandomForest:                  ',accuracy_score(y_test,y_smote_predrf))
```

```
Accuracy score of various models:
LogisticRegression:             0.431612715077282
KNeighborsClassifier:           0.6622922134733158
DecisionTree:                   0.6429678746297064
RandomForest:                   0.7035502141179721
```

In [ ]: