earthquake

November 17, 2022

[4]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn import preprocessing
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import␣
↪train_test_split,learning_curve,StratifiedKFold,␣
↪KFold,GridSearchCV,RandomizedSearchCV,cross_validate,train_test_split
from sklearn.ensemble import␣
↪RandomForestClassifier,AdaBoostClassifier,GradientBoostingClassifier
from sklearn.metrics import␣
↪recall_score,precision_score,f1_score,accuracy_score,confusion_matrix,make_scorer
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn import tree
from xgboost import XGBClassifier
from sklearn.metrics import confusion_matrix
import sklearn
import featuretools as ft
from imblearn.under_sampling import RandomUnderSampler
from imblearn.over_sampling import SMOTE
from collections import Counter
import tpot
import warnings
warnings.filterwarnings('ignore')
```

[9]:
```
```

[9]: '1.1.1'

[5]:
```python
values = pd.read_csv('train_values.csv')
values_raw = pd.read_csv('train_values.csv').drop(['building_id'],axis=1)

labels = pd.read_csv('train_labels.csv')
test_values = pd.read_csv('test_values.csv')
df =values.merge(labels,on='building_id',how='left')
df =df.drop(['building_id'],axis=1)
columns = df.columns
labels = labels["damage_grade"]
df.head(5)
```

[5]:
```
   geo_level_1_id  geo_level_2_id  geo_level_3_id  count_floors_pre_eq  age  \
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

   area_percentage  height_percentage land_surface_condition foundation_type  \
0                6                  5                      t               r
1                8                  7                      o               r
2                5                  5                      t               r
3                6                  5                      t               r
4                8                  9                      t               r

  roof_type  … has_secondary_use_hotel has_secondary_use_rental  \
0         n  …                       0                        0
1         n  …                       0                        0
2         n  …                       0                        0
3         n  …                       0                        0
4         n  …                       0                        0

  has_secondary_use_institution has_secondary_use_school  \
0                             0                        0
1                             0                        0
2                             0                        0
3                             0                        0
4                             0                        0

   has_secondary_use_industry  has_secondary_use_health_post  \
0                           0                              0
1                           0                              0
2                           0                              0
3                           0                              0
4                           0                              0

   has_secondary_use_gov_office  has_secondary_use_use_police  \
```

```
0                              0                              0
1                              0                              0
2                              0                              0
3                              0                              0
4                              0                              0

   has_secondary_use_other  damage_grade
0                        0             3
1                        0             2
2                        0             3
3                        0             2
4                        0             3

[5 rows x 39 columns]
```

[32]:
```
df.describe()
```

[32]:
```
       geo_level_1_id  geo_level_2_id  geo_level_3_id  count_floors_pre_eq  \
count   260601.000000   260601.000000   260601.000000        260601.000000
mean        13.900353      701.074685     6257.876148             2.129723
std          8.033617      412.710734     3646.369645             0.727665
min          0.000000        0.000000        0.000000             1.000000
25%          7.000000      350.000000     3073.000000             2.000000
50%         12.000000      702.000000     6270.000000             2.000000
75%         21.000000     1050.000000     9412.000000             2.000000
max         30.000000     1427.000000    12567.000000             9.000000

                 age  area_percentage  height_percentage  \
count  260601.000000    260601.000000      260601.000000
mean       26.535029         8.018051           5.434365
std        73.565937         4.392231           1.918418
min         0.000000         1.000000           2.000000
25%        10.000000         5.000000           4.000000
50%        15.000000         7.000000           5.000000
75%        30.000000         9.000000           6.000000
max       995.000000       100.000000          32.000000

       has_superstructure_adobe_mud  has_superstructure_mud_mortar_stone  \
count                 260601.000000                        260601.000000
mean                       0.088645                             0.761935
std                        0.284231                             0.425900
min                        0.000000                             0.000000
25%                        0.000000                             1.000000
50%                        0.000000                             1.000000
75%                        0.000000                             1.000000
max                        1.000000                             1.000000
```

```
       has_superstructure_stone_flag  …  has_secondary_use_hotel  \
count                  260601.000000  …            260601.000000
mean                        0.034332  …                 0.033626
std                         0.182081  …                 0.180265
min                         0.000000  …                 0.000000
25%                         0.000000  …                 0.000000
50%                         0.000000  …                 0.000000
75%                         0.000000  …                 0.000000
max                         1.000000  …                 1.000000

       has_secondary_use_rental  has_secondary_use_institution  \
count             260601.000000                  260601.000000
mean                   0.008101                       0.000940
std                    0.089638                       0.030647
min                    0.000000                       0.000000
25%                    0.000000                       0.000000
50%                    0.000000                       0.000000
75%                    0.000000                       0.000000
max                    1.000000                       1.000000

       has_secondary_use_school  has_secondary_use_industry  \
count             260601.000000               260601.000000
mean                   0.000361                    0.001071
std                    0.018989                    0.032703
min                    0.000000                    0.000000
25%                    0.000000                    0.000000
50%                    0.000000                    0.000000
75%                    0.000000                    0.000000
max                    1.000000                    1.000000

       has_secondary_use_health_post  has_secondary_use_gov_office  \
count                  260601.000000                 260601.000000
mean                        0.000188                      0.000146
std                         0.013711                      0.012075
min                         0.000000                      0.000000
25%                         0.000000                      0.000000
50%                         0.000000                      0.000000
75%                         0.000000                      0.000000
max                         1.000000                      1.000000

       has_secondary_use_use_police  has_secondary_use_other   damage_grade
count                 260601.000000            260601.000000  260601.000000
mean                       0.000088                 0.005119       2.238272
std                        0.009394                 0.071364       0.611814
min                        0.000000                 0.000000       1.000000
25%                        0.000000                 0.000000       2.000000
50%                        0.000000                 0.000000       2.000000
```

```
75%                      0.000000                  0.000000       3.000000
max                      1.000000                  1.000000       3.000000

[8 rows x 31 columns]
```

[33]:
```python
df[df.duplicated()] # no duplicated data
```

[33]:
```
        geo_level_1_id  geo_level_2_id  geo_level_3_id  count_floors_pre_eq  \
2702                20             508           11256                    2
3003                21             111           11714                    2
3563                 6            1108            5909                    2
4188                27             269           11121                    2
4937                10            1382            5036                    2
…                  …              …              …                    …
260546              10              90           10884                    2
260575               4             144            5751                    2
260580              17             875           10462                    2
260583               7             322            2843                    2
260595               8             268            4718                    2

        age  area_percentage  height_percentage land_surface_condition  \
2702     10                9                  6                      t
3003     10                9                  5                      t
3563     30                4                  7                      t
4188     15               10                  7                      n
4937     15                5                  5                      t
…      …              …                …                    …
260546   20                5                  5                      t
260575    5                4                  4                      t
260580    5                6                  5                      t
260583   10                6                  6                      t
260595   20                8                  5                      t

       foundation_type roof_type  … has_secondary_use_hotel  \
2702                 w         q  …                       0
3003                 r         q  …                       0
3563                 r         n  …                       0
4188                 r         n  …                       0
4937                 r         n  …                       0
…                  …       …  …                     …
260546               r         q  …                       0
260575               r         q  …                       0
260580               r         n  …                       0
260583               r         n  …                       0
260595               r         n  …                       0

       has_secondary_use_rental has_secondary_use_institution  \
```

2702	0	0
3003	0	0
3563	0	0
4188	0	0
4937	0	0
…	…	…
260546	0	0
260575	0	0
260580	0	0
260583	0	0
260595	0	0

	has_secondary_use_school	has_secondary_use_industry \
2702	0	0
3003	0	0
3563	0	0
4188	0	0
4937	0	0
…	…	…
260546	0	0
260575	0	0
260580	0	0
260583	0	0
260595	0	0

	has_secondary_use_health_post	has_secondary_use_gov_office \
2702	0	0
3003	0	0
3563	0	0
4188	0	0
4937	0	0
…	…	…
260546	0	0
260575	0	0
260580	0	0
260583	0	0
260595	0	0

	has_secondary_use_use_police	has_secondary_use_other	damage_grade
2702	0	0	1
3003	0	0	3
3563	0	0	2
4188	0	0	3
4937	0	0	3
…	…	…	…
260546	0	0	2
260575	0	0	2

```
260580                        0                        0              3
260583                        0                        0              2
260595                        0                        0              3

[12319 rows x 39 columns]
```

[23]:
```
df.isna().sum()/df.shape[0] #percentage of null values
```

[23]:
```
building_id                                 0.0
geo_level_1_id                              0.0
geo_level_2_id                              0.0
geo_level_3_id                              0.0
count_floors_pre_eq                         0.0
age                                         0.0
area_percentage                             0.0
height_percentage                           0.0
land_surface_condition                      0.0
foundation_type                             0.0
roof_type                                   0.0
ground_floor_type                           0.0
other_floor_type                            0.0
position                                    0.0
plan_configuration                          0.0
has_superstructure_adobe_mud                0.0
has_superstructure_mud_mortar_stone         0.0
has_superstructure_stone_flag               0.0
has_superstructure_cement_mortar_stone      0.0
has_superstructure_mud_mortar_brick         0.0
has_superstructure_cement_mortar_brick      0.0
has_superstructure_timber                   0.0
has_superstructure_bamboo                   0.0
has_superstructure_rc_non_engineered        0.0
has_superstructure_rc_engineered            0.0
has_superstructure_other                    0.0
legal_ownership_status                      0.0
count_families                              0.0
has_secondary_use                           0.0
has_secondary_use_agriculture               0.0
has_secondary_use_hotel                     0.0
has_secondary_use_rental                    0.0
has_secondary_use_institution               0.0
has_secondary_use_school                    0.0
has_secondary_use_industry                  0.0
has_secondary_use_health_post               0.0
has_secondary_use_gov_office                0.0
has_secondary_use_use_police                0.0
has_secondary_use_other                     0.0
```

```
damage_grade                                0.0
dtype: float64
```

[ ]:

```

```

0.1 Exploratory Data Analysis

[24]:

```python
print('Shape of DF:',df.shape)
print(df.dtypes,'\n') #there is no null values
df.info()
```

```
Shape of DF: (260601, 40)
building_id                                 int64
geo_level_1_id                              int64
geo_level_2_id                              int64
geo_level_3_id                              int64
count_floors_pre_eq                         int64
age                                         int64
area_percentage                             int64
height_percentage                           int64
land_surface_condition                     object
foundation_type                            object
roof_type                                  object
ground_floor_type                          object
other_floor_type                           object
position                                   object
plan_configuration                         object
has_superstructure_adobe_mud                int64
has_superstructure_mud_mortar_stone         int64
has_superstructure_stone_flag               int64
has_superstructure_cement_mortar_stone      int64
has_superstructure_mud_mortar_brick         int64
has_superstructure_cement_mortar_brick      int64
has_superstructure_timber                   int64
has_superstructure_bamboo                   int64
has_superstructure_rc_non_engineered        int64
has_superstructure_rc_engineered            int64
has_superstructure_other                    int64
legal_ownership_status                     object
count_families                              int64
has_secondary_use                           int64
has_secondary_use_agriculture               int64
has_secondary_use_hotel                     int64
has_secondary_use_rental                    int64
has_secondary_use_institution               int64
has_secondary_use_school                    int64
has_secondary_use_industry                  int64
```

```
has_secondary_use_health_post            int64
has_secondary_use_gov_office             int64
has_secondary_use_use_police             int64
has_secondary_use_other                  int64
damage_grade                             int64
dtype: object

<class 'pandas.core.frame.DataFrame'>
Int64Index: 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: 81.5+ MB
```

[25]:
```
df.describe()
```

[25]:
```
        building_id  geo_level_1_id  geo_level_2_id  geo_level_3_id  \
count  2.606010e+05   260601.000000   260601.000000   260601.000000
mean   5.256755e+05       13.900353      701.074685     6257.876148
std    3.045450e+05        8.033617      412.710734     3646.369645
min    4.000000e+00        0.000000        0.000000        0.000000
25%    2.611900e+05        7.000000      350.000000     3073.000000
50%    5.257570e+05       12.000000      702.000000     6270.000000
75%    7.897620e+05       21.000000     1050.000000     9412.000000
max    1.052934e+06       30.000000     1427.000000    12567.000000

       count_floors_pre_eq            age  area_percentage  height_percentage  \
count        260601.000000  260601.000000    260601.000000      260601.000000
mean              2.129723      26.535029         8.018051           5.434365
std               0.727665      73.565937         4.392231           1.918418
min               1.000000       0.000000         1.000000           2.000000
25%               2.000000      10.000000         5.000000           4.000000
50%               2.000000      15.000000         7.000000           5.000000
75%               2.000000      30.000000         9.000000           6.000000
max               9.000000     995.000000       100.000000          32.000000

       has_superstructure_adobe_mud  has_superstructure_mud_mortar_stone  …  \
count                 260601.000000                        260601.000000  …
mean                       0.088645                             0.761935  …
std                        0.284231                             0.425900  …
min                        0.000000                             0.000000  …
25%                        0.000000                             1.000000  …
50%                        0.000000                             1.000000  …
75%                        0.000000                             1.000000  …
max                        1.000000                             1.000000  …

       has_secondary_use_hotel  has_secondary_use_rental  \
count            260601.000000             260601.000000
mean                  0.033626                  0.008101
std                   0.180265                  0.089638
min                   0.000000                  0.000000
25%                   0.000000                  0.000000
50%                   0.000000                  0.000000
75%                   0.000000                  0.000000
```

```
max                   1.000000                  1.000000

       has_secondary_use_institution  has_secondary_use_school  \
count                  260601.000000             260601.000000
mean                        0.000940                  0.000361
std                         0.030647                  0.018989
min                         0.000000                  0.000000
25%                         0.000000                  0.000000
50%                         0.000000                  0.000000
75%                         0.000000                  0.000000
max                         1.000000                  1.000000

       has_secondary_use_industry  has_secondary_use_health_post  \
count               260601.000000                  260601.000000
mean                     0.001071                       0.000188
std                      0.032703                       0.013711
min                      0.000000                       0.000000
25%                      0.000000                       0.000000
50%                      0.000000                       0.000000
75%                      0.000000                       0.000000
max                      1.000000                       1.000000

       has_secondary_use_gov_office  has_secondary_use_use_police  \
count                 260601.000000                 260601.000000
mean                       0.000146                      0.000088
std                        0.012075                      0.009394
min                        0.000000                      0.000000
25%                        0.000000                      0.000000
50%                        0.000000                      0.000000
75%                        0.000000                      0.000000
max                        1.000000                      1.000000

       has_secondary_use_other   damage_grade
count            260601.000000  260601.000000
mean                  0.005119       2.238272
std                   0.071364       0.611814
min                   0.000000       1.000000
25%                   0.000000       2.000000
50%                   0.000000       2.000000
75%                   0.000000       3.000000
max                   1.000000       3.000000

[8 rows x 32 columns]
```

[26]:
```
df[df.duplicated()] # no duplicated data
```

[26]: Empty DataFrame
Columns: [building_id, geo_level_1_id, geo_level_2_id, geo_level_3_id,
count_floors_pre_eq, age, area_percentage, height_percentage,
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,
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, damage_grade]
Index: []

[0 rows x 40 columns]

[27]:
```
df.isna().sum()/df.shape[0] #percentage of null values
```

[27]:
```
building_id                               0.0
geo_level_1_id                            0.0
geo_level_2_id                            0.0
geo_level_3_id                            0.0
count_floors_pre_eq                       0.0
age                                       0.0
area_percentage                           0.0
height_percentage                         0.0
land_surface_condition                    0.0
foundation_type                           0.0
roof_type                                 0.0
ground_floor_type                         0.0
other_floor_type                          0.0
position                                  0.0
plan_configuration                        0.0
has_superstructure_adobe_mud              0.0
has_superstructure_mud_mortar_stone       0.0
has_superstructure_stone_flag             0.0
has_superstructure_cement_mortar_stone    0.0
has_superstructure_mud_mortar_brick       0.0
has_superstructure_cement_mortar_brick    0.0
has_superstructure_timber                 0.0
has_superstructure_bamboo                 0.0
has_superstructure_rc_non_engineered      0.0
has_superstructure_rc_engineered          0.0
```

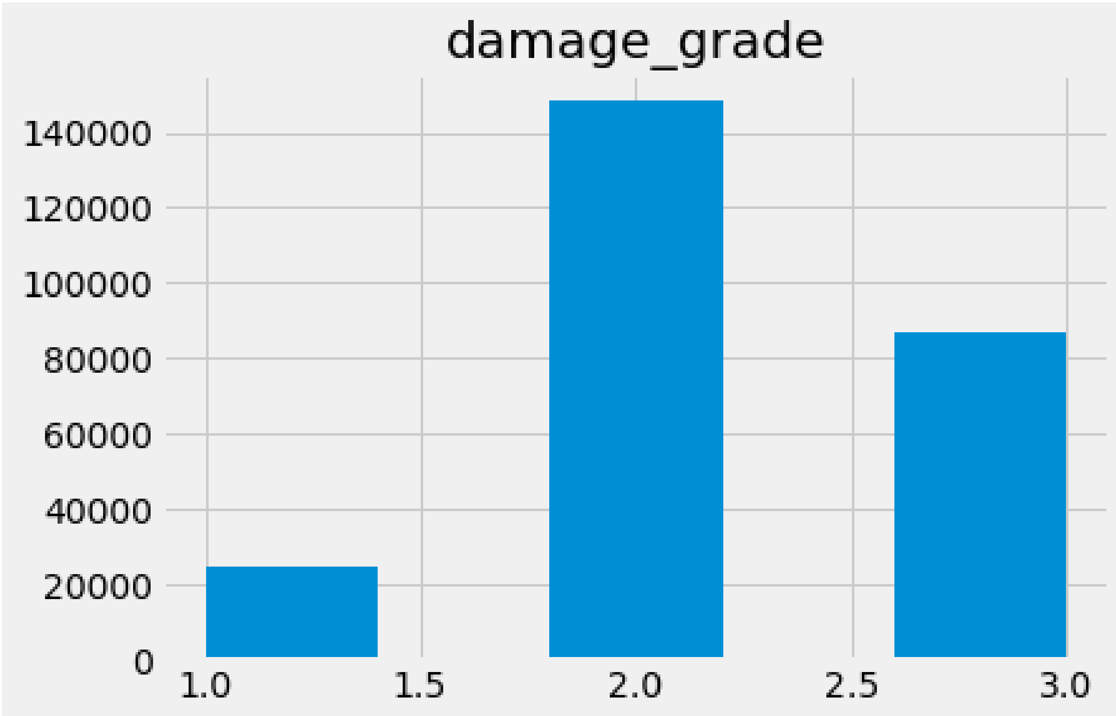
```
has_superstructure_other                  0.0
legal_ownership_status                    0.0
count_families                            0.0
has_secondary_use                         0.0
has_secondary_use_agriculture             0.0
has_secondary_use_hotel                   0.0
has_secondary_use_rental                  0.0
has_secondary_use_institution             0.0
has_secondary_use_school                  0.0
has_secondary_use_industry                0.0
has_secondary_use_health_post             0.0
has_secondary_use_gov_office              0.0
has_secondary_use_use_police              0.0
has_secondary_use_other                   0.0
damage_grade                              0.0
dtype: float64
```

[28]:
```
df['damage_grade'].value_counts() #not goodly balanced
```

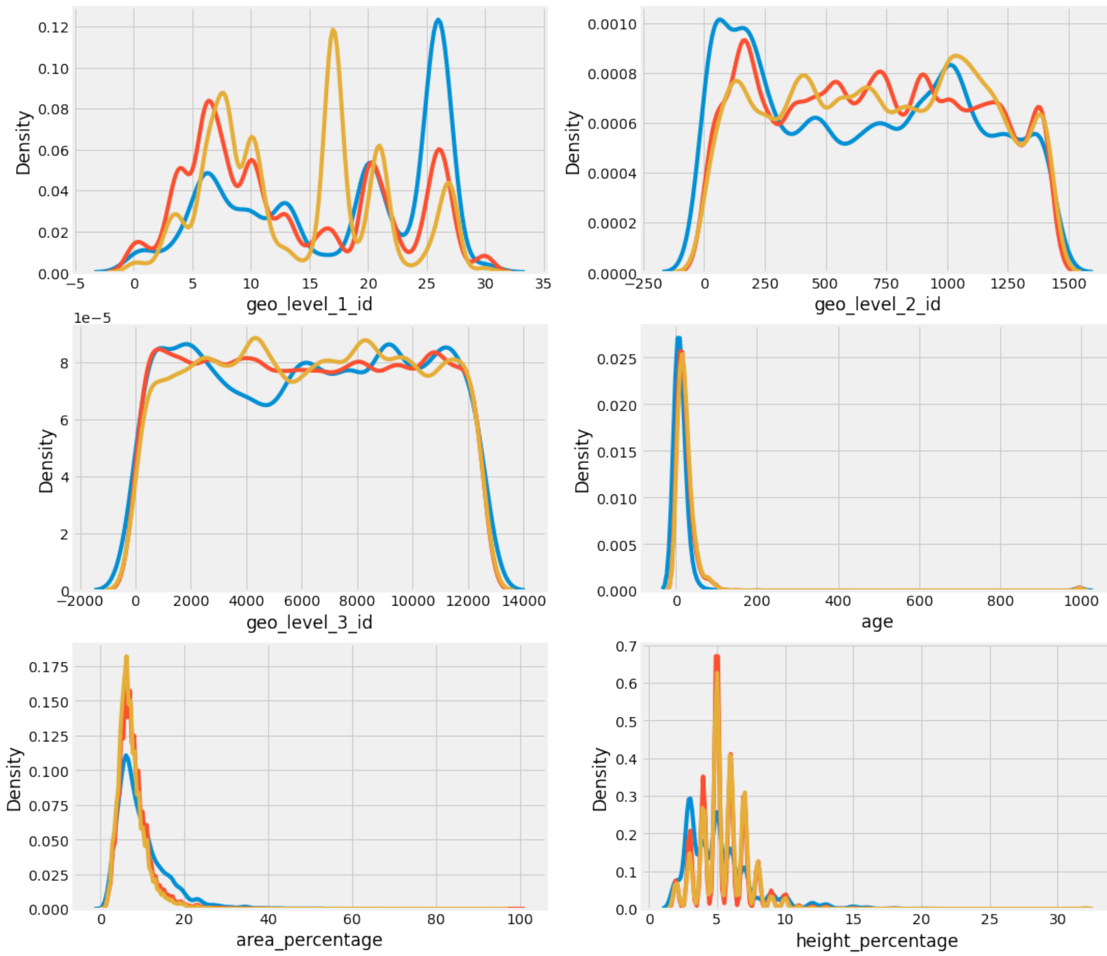
[28]:
```
2    148259
3     87218
1     25124
Name: damage_grade, dtype: int64
```

[91]:
```
undersample = RandomUnderSampler(sampling_strategy='0.5')
SMOTE = SMOTE()
```

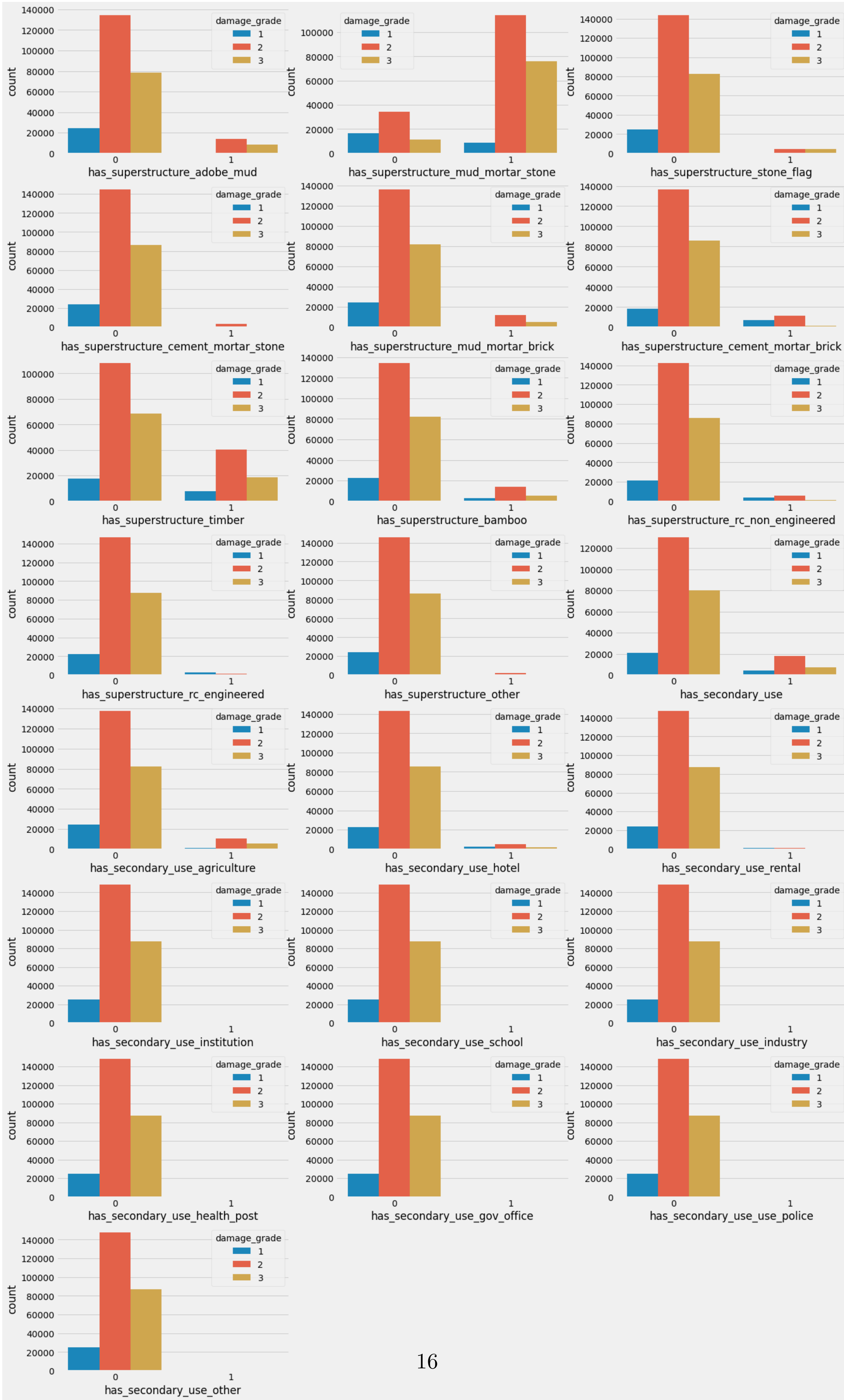
[29]:
```
hist = df.hist('damage_grade',bins=5)
plt.show()
```



[6]:
```
continous_values =␣
↪['geo_level_1_id','geo_level_2_id','geo_level_3_id','age','area_percentage','height_percent

def densityPlot(continous_values):
  fig = plt.figure(figsize=(18,16))
  plt.style.use('fivethirtyeight')
  for i,txt in enumerate(continous_values):
    ax = fig.add_subplot(3,2,i+1)
    sns.kdeplot(df.loc[df['damage_grade'] == 1, txt], ax=ax,␣
↪label='damage_grade==1')
    sns.kdeplot(df.loc[df['damage_grade'] == 2, txt], ax=ax,␣
↪label='damage_grade==2')
    sns.kdeplot(df.loc[df['damage_grade'] == 3, txt], ax=ax,␣
↪label='damage_grade==3')
  plt.show()
densityPlot(continous_values)
```

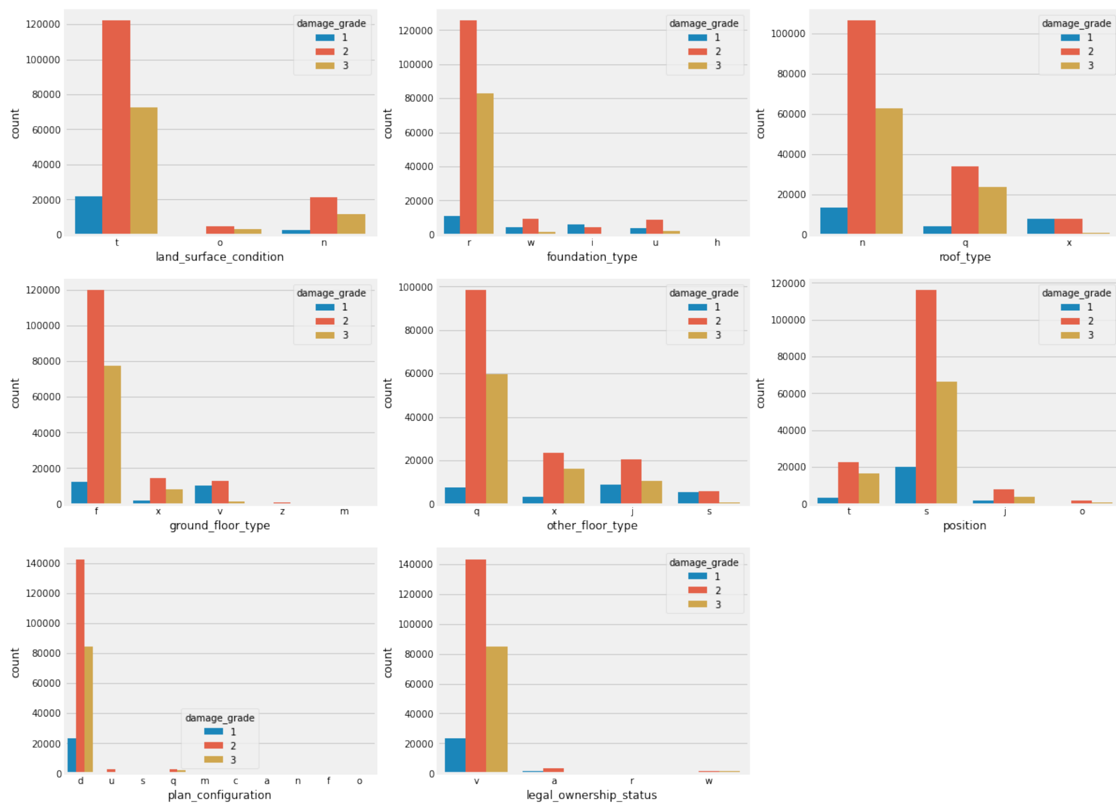


[7]:
```
binary_features = df.columns[df.columns.str.startswith('has')]

def countPlot(binary_features):
  plt.rcParams['font.size'] = 18
  plt.style.use('fivethirtyeight')
  fig = plt.figure(figsize=(20,37))
  for i,txt in enumerate(binary_features):
    ax = fig.add_subplot(8,3,i+1)
    sns.countplot(x=df[txt], ax=ax, hue=df['damage_grade'])
  plt.show()
countPlot(binary_features)
```


damage_grade
1
2
3
count
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

Except has_superstructure_cement_mortar_stone other binary features have more zero than 1 and some columns have only zero values

[6]:
```python
categorical_features = df.select_dtypes(include=object).columns

def catPlot(categorical_features):
  plt.rcParams['font.size'] = 18
  plt.style.use('fivethirtyeight')
  fig = plt.figure(figsize=(18,15))
  for i,txt in enumerate(categorical_features):
    ax = fig.add_subplot(3,3,i+1)
    sns.countplot(x=df[txt], ax=ax, hue=df['damage_grade'])
  plt.show()
catPlot(categorical_features)
```



[ ]:
```python

```

[31]:
```python
values_raw
```

[31]:

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

	age	area_percentage	height_percentage	land_surface_condition	\
0	30	6	5	t	
1	10	8	7	o	
2	10	5	5	t	
3	10	6	5	t	
4	30	8	9	t	
…	…	…	…	…	
260596	55	6	3	n	
260597	0	6	5	t	
260598	55	6	7	t	
260599	10	14	6	t	
260600	10	7	6	n	

	foundation_type	roof_type	…	has_secondary_use_agriculture	\
0	r	n	…	0	
1	r	n	…	0	
2	r	n	…	0	
3	r	n	…	0	
4	r	n	…	0	
…	…	…	…	…	
260596	r	n	…	0	
260597	r	n	…	0	
260598	r	q	…	0	
260599	r	x	…	0	
260600	r	n	…	0	

	has_secondary_use_hotel	has_secondary_use_rental	\
0	0	0	
1	0	0	
2	0	0	
3	0	0	
4	0	0	
…	…	…	
260596	0	0	

```
260597                          0                        0
260598                          0                        0
260599                          0                        0
260600                          0                        0

       has_secondary_use_institution  has_secondary_use_school  \
0                                  0                         0
1                                  0                         0
2                                  0                         0
3                                  0                         0
4                                  0                         0
…                                  …                         …
260596                             0                         0
260597                             0                         0
260598                             0                         0
260599                             0                         0
260600                             0                         0

        has_secondary_use_industry  has_secondary_use_health_post  \
0                                0                              0
1                                0                              0
2                                0                              0
3                                0                              0
4                                0                              0
…                                …                              …
260596                           0                              0
260597                           0                              0
260598                           0                              0
260599                           0                              0
260600                           0                              0

        has_secondary_use_gov_office  has_secondary_use_use_police  \
0                                  0                             0
1                                  0                             0
2                                  0                             0
3                                  0                             0
4                                  0                             0
…                                  …                             …
260596                             0                             0
260597                             0                             0
260598                             0                             0
260599                             0                             0
260600                             0                             0

        has_secondary_use_other
0                             0
1                             0
```

```
2                              0
3                              0
4                              0
…                         …
260596                         0
260597                         0
260598                         0
260599                         0
260600                         0

[260601 rows x 38 columns]
```

MODELS

[7]:
```
from sklearn.preprocessing import OneHotEncoder
from sklearn.feature_selection import SelectKBest, chi2,f_classif
values_onehot = pd.get_dummies(values_raw, columns=categorical_features,␣
 ↪prefix=categorical_features)

encoder = LabelEncoder()
for i in categorical_features:
    values[i] = encoder.fit_transform(values[i])
```

[62]:
```
values_onehot
```

[62]:
```
        geo_level_1_id  geo_level_2_id  geo_level_3_id  count_floors_pre_eq  \
0                    6             487           12198                    2
1                    8             900            2812                    2
2                   21             363            8973                    2
3                   22             418           10694                    2
4                   11             131            1488                    3
…                  …             …             …                   …
260596              25            1335            1621                    1
260597              17             715            2060                    2
260598              17              51            8163                    3
260599              26              39            1851                    2
260600              21               9            9101                    3

        age  area_percentage  height_percentage  has_superstructure_adobe_mud  \
0        30                6                  5                             1
1        10                8                  7                             0
2        10                5                  5                             0
3        10                6                  5                             0
4        30                8                  9                             1
…      …              …                 …                           …
260596   55                6                  3                             0
260597    0                6                  5                             0
```

```
260598   55            6                7                         0
260599   10           14                6                         0
260600   10            7                6                         0

        has_superstructure_mud_mortar_stone  has_superstructure_stone_flag  \
0                                         1                              0
1                                         1                              0
2                                         1                              0
3                                         1                              0
4                                         0                              0
…                                       …                            …
260596                                    1                              0
260597                                    1                              0
260598                                    1                              0
260599                                    0                              0
260600                                    1                              0

        …  plan_configuration_m  plan_configuration_n  plan_configuration_o  \
0       …                     0                     0                     0
1       …                     0                     0                     0
2       …                     0                     0                     0
3       …                     0                     0                     0
4       …                     0                     0                     0
…     …                     …                     …                     …
260596  …                     0                     0                     0
260597  …                     0                     0                     0
260598  …                     0                     0                     0
260599  …                     0                     0                     0
260600  …                     0                     0                     0

        plan_configuration_q  plan_configuration_s  plan_configuration_u  \
0                          0                     0                     0
1                          0                     0                     0
2                          0                     0                     0
3                          0                     0                     0
4                          0                     0                     0
…                        …                     …                     …
260596                     1                     0                     0
260597                     0                     0                     0
260598                     0                     0                     0
260599                     0                     0                     0
260600                     0                     0                     0

        legal_ownership_status_a  legal_ownership_status_r  \
0                              0                         0
1                              0                         0
2                              0                         0
```

```
3                                  0                           0
4                                  0                           0
…                               …                          …
260596                             0                           0
260597                             0                           0
260598                             0                           0
260599                             0                           0
260600                             0                           0

        legal_ownership_status_v  legal_ownership_status_w
0                              1                         0
1                              1                         0
2                              1                         0
3                              1                         0
4                              1                         0
…                           …                      …
260596                         1                         0
260597                         1                         0
260598                         1                         0
260599                         1                         0
260600                         1                         0

[260601 rows x 68 columns]
```

[8]:
```
#maxabs = preprocessing.MaxAbsScaler()
#x_scaled=x_scaled.set_axis(values.columns,axis=1)

#x_scaled = pd.DataFrame(standard_scaler.fit_transform(values))

#values_raw_scaled = pd.DataFrame(maxabs.fit_transform(values_onehot))

#values=x_scaled.set_axis(columns[values.columns],axis=1)

#values = pd.get_dummies(values,drop_first = True)
#standard_scaler = preprocessing.StandardScaler()
#x_scaled = pd.DataFrame(min_max_scaler.fit_transform(values))
#print(x_scaled.shape)

min_max_scaler = preprocessing.MinMaxScaler()

x_scaled_onehot = pd.DataFrame(min_max_scaler.fit_transform(values_onehot))

print(x_scaled_onehot.shape)
```

```
(260601, 39)
(260601, 68)
```

[37]:
```
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(2)
#x_scaled_generated = pd.DataFrame(poly.fit_transform(x_scaled))
#values = pd.DataFrame(SelectKBest(f_classif, k=20).fit_transform(x_scaled,␣
 ↪labels))
#values_onehot = pd.DataFrame(SelectKBest(f_classif, k=20).
 ↪fit_transform(values_raw_scaled, labels))
values_onehot = pd.DataFrame(SelectKBest(f_classif, k=20).
 ↪fit_transform(x_scaled_onehot, labels))
```

[43]:
```
x_scaled_onehot
```

[43]:
```
               0         1         2      3         4         5         6   \
0       0.200000  0.341275  0.970637  0.125  0.030151  0.050505  0.100000
1       0.266667  0.630694  0.223761  0.125  0.010050  0.070707  0.166667
2       0.700000  0.254380  0.714013  0.125  0.010050  0.040404  0.100000
3       0.733333  0.292922  0.850959  0.125  0.010050  0.050505  0.100000
4       0.366667  0.091801  0.118405  0.250  0.030151  0.070707  0.233333
…            …         …         …      …         …         …         …
260596  0.833333  0.935529  0.128989  0.000  0.055276  0.050505  0.033333
260597  0.566667  0.501051  0.163921  0.125  0.000000  0.050505  0.100000
260598  0.566667  0.035739  0.649558  0.250  0.055276  0.050505  0.166667
260599  0.866667  0.027330  0.147291  0.125  0.010050  0.131313  0.133333
260600  0.700000  0.006307  0.724198  0.250  0.010050  0.060606  0.133333

         7    8    9  …   58   59   60   61   62   63   64   65   66   67
0       1.0  1.0  0.0  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0  0.0
1       0.0  1.0  0.0  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0  0.0
2       0.0  1.0  0.0  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0  0.0
3       0.0  1.0  0.0  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0  0.0
4       1.0  0.0  0.0  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0  0.0
…       …    …    …  …  …    …    …    …    …    …    …    …    …    …
260596  0.0  1.0  0.0  …  0.0  0.0  0.0  1.0  0.0  0.0  0.0  0.0  1.0  0.0
260597  0.0  1.0  0.0  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0  0.0
260598  0.0  1.0  0.0  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0  0.0
260599  0.0  0.0  0.0  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0  0.0
260600  0.0  1.0  0.0  …  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0  0.0

[260601 rows x 68 columns]
```

[9]:
```
#X = x_scaled
X = x_scaled_onehot
#X = values_onehot
y = labels
#X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.
 ↪2,random_state=1453)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=1)
#X_train_under, y_train_under = undersample.fit_resample(X_train, y_train)


#X_train_SMOTE, y_train_SMOTE = SMOTE.fit_resample(X_train, y_train)

#pipeline_optimizer = tpot.TPOTClassifier(generations=5, #number of iterations␣
 ↪to run the training
                                         #population_size=20, #number of␣
 ↪individuals to train
                                         #cv=5) #number of folds in␣
 ↪StratifiedKFold
#pipeline_optimizer.fit(X_train, y_train) #fit the pipeline optimizer - can␣
 ↪take a long time
#print(pipeline_optimizer.score(X_test, y_test)) #print scoring for the pipeline
#pipeline_optimizer.export('tpot_exported_pipeline.py')
```

[34]:
```
weight1 = (Counter(y_train)[1]/sum(Counter(y_train).values()))
weight2 = (Counter(y_train)[2]/sum(Counter(y_train).values()))
weight3 = (Counter(y_train)[3]/sum(Counter(y_train).values()))
print(weight1)
print(weight2)
print(weight3)
```

```
0.09653683806600154
0.5683278971603991
0.3351352647735994
```

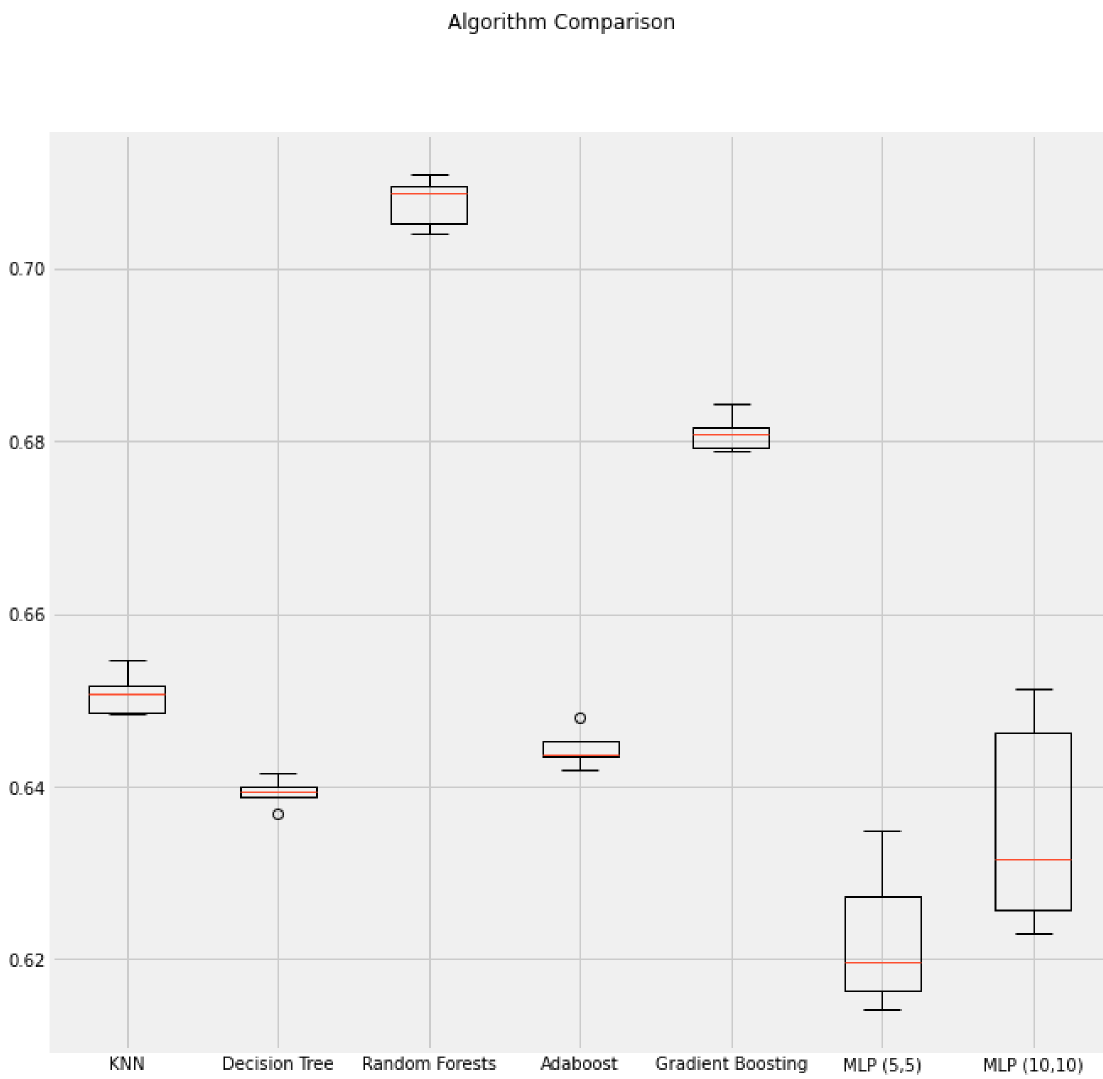
[13]:
```
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn import model_selection



classifiers = {'KNN': KNeighborsClassifier(3),
               'Decision Tree Classifier':DecisionTreeClassifier(max_features =␣
 ↪None,
                            max_depth = 45,
                            min_samples_split = 3,
                            min_samples_leaf = 30,
                            random_state=42,class_weight='balanced'),
               'Random Forests Classifier':RandomForestClassifier(criterion=␣
 ↪'entropy', max_features='sqrt', n_estimators=280,class_weight='balanced'),
               'Adaboost Classifier':AdaBoostClassifier(),
               'Gradient Boosting Classifier':GradientBoostingClassifier(),
               'MLP (5,5)':␣
 ↪MLPClassifier(solver='lbfgs',max_iter=500,hidden_layer_sizes=(5, 5),␣
 ↪random_state=1),
```

```
             'MLP (10,10)':␣
↪MLPClassifier(solver='lbfgs',max_iter=500,hidden_layer_sizes=(10, 10),␣
↪random_state=1)}
models = []
names = []
results = []
names = []
scoring = 'accuracy'
last = 0
kf = KFold(n_splits=5)
kf.get_n_splits(X_train)
X_train_temp = X_train
y_train_temp = y_train
for name, model in classifiers.items():
    kfold = model_selection.KFold(n_splits=5)
    cv_results = model_selection.cross_val_score(model, X_train_temp,␣
↪y_train_temp, cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
    #for train_index, val_index in kf.split(X_train_temp):
    #    kfold = model_selection.KFold(n_splits=10, random_state=1452)
    #    #print("TRAIN:", train_index, "TEST:", val_index)
    #    X_train, X_val = X_train_temp.iloc[train_index], X_train_temp.
↪iloc[val_index]
    #    y_train, y_val = y_train_temp.iloc[train_index], y_train_temp.
↪iloc[val_index]
    #    print("Model Name:",name)
    #    model.fit(X_train, y_train)
    #    y_pred = model.predict(X_val)
    #    f1 = f1_score(y_val, y_pred, average="macro")
    #    recall_scores =recall_score(y_val, y_pred, average="macro")
    #    prec_scores =precision_score(y_val, y_pred, average="macro")
    #    print("training:",model.score(X_train_temp, y_train_temp))
    #    print("test:",model.score(X_val, y_val))
    #    print("f1:",f1)
    #    print("recall score:",recall_scores)
    #    print("precision score:",prec_scores)
    #    conf_mat = confusion_matrix(y_val, y_pred)
    #    print(conf_mat)
    #    print()
    #    if f1 > last:
    #        names.append(name)
    #        models.append(model)
    #        last = f1
#print("test")
```

```
#name = names[-1]
#print("Model Name:",name)
#model = models[-1]
#y_pred = model.predict(X_test)
#f1 = f1_score(y_test, y_pred, average="macro")
#recall_scores =recall_score(y_test, y_pred, average="macro")
#prec_scores =precision_score(y_test, y_pred, average="macro")
#print("test:",model.score(X_test, y_test))
#print("f1:",f1)
#print("recall score:",recall_scores)
#print("precision score:",prec_scores)
#conf_mat = confusion_matrix(y_test, y_pred)
#disp = ConfusionMatrixDisplay(confusion_matrix=conf_mat, display_labels=["Low␣
↪Damage 1","Medium Damage 2","High Damage 3"])
#disp.plot(cmap=plt.cm.Blues)
#print(conf_mat)
#print()
```

```
KNN: 0.650844 (0.002296)
Decision Tree Classifier: 0.639332 (0.001559)
Random Forests Classifier: 0.707708 (0.002534)
Adaboost Classifier: 0.644474 (0.002046)
Gradient Boosting Classifier: 0.680991 (0.001955)
MLP (5,5): 0.622563 (0.007627)
MLP (10,10): 0.635557 (0.011238)
```

[24]:
```
names = ['KNN','Decision Tree','Random Forests','Adaboost','Gradient␣
↪Boosting','MLP (5,5)', 'MLP (10,10)']
fig = plt.figure(figsize = (10,10))
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```



[26]:

```
#value='{'gamma': 0, 'learning_rate': 0.3, 'max_depth': 5, 'n_estimators': 500,␣
↪'subsample': 0.8}'
parameters = {
        'n_estimators': [100, 500],
        'subsample': [0.8, 1.0],
        'gamma' : [0,1,5],
        'max_depth': [3, 4, 5],
        'learning_rate': [0.1, 0.3]}

xgboost = XGBClassifier()
xgboost_cv = RandomizedSearchCV(xgboost, parameters, cv = 3, n_jobs = -1,␣
↪verbose = 2)
xgboost_cv.fit(X_train_temp, y_train_temp)
```

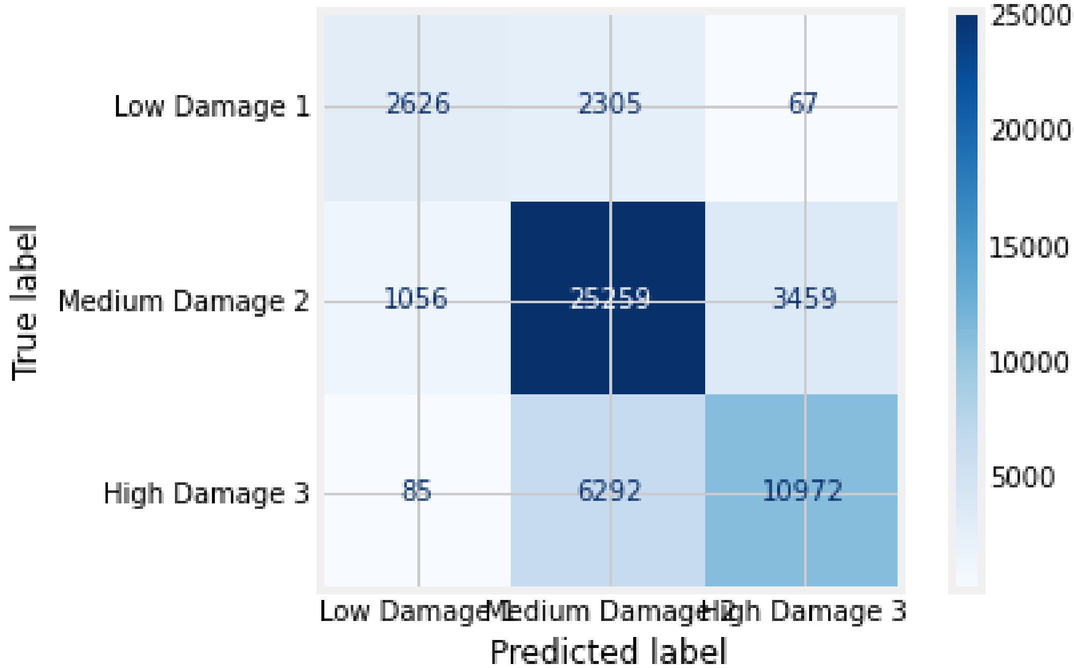
```
best = xgboost_cv.best_params_
xgboost = XGBClassifier(**best)
#xgboost = XGBClassifier(eval_metric='rmse',learning_rate= 0.1, n_estimators=␣
↪250,gamma= 0, max_depth=3) #eval_metric='auc',learning_rate= 0.01,␣
↪n_estimators= 900, min_child_weight= 9,gamma= 0.1, reg_lambda= 1, subsample=␣
↪0.6
xgb_tuned =  xgboost.fit(X_train_temp,y_train_temp)
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
```

[30]:
```
print("Model Name: XGBTUNED")
y_pred = xgb_tuned.predict(X_test)
f1 = f1_score(y_test, y_pred, average="macro")
recall_scores =recall_score(y_test, y_pred, average="macro")
prec_scores =precision_score(y_test, y_pred, average="macro")
print("training:",xgb_tuned.score(X_train_temp, y_train_temp))
print("test:",xgb_tuned.score(X_test, y_test))
print("f1:",f1)
print("recall score:",recall_scores)
print("precision score:",prec_scores)
conf_mat = confusion_matrix(y_test, y_pred)
print(conf_mat)
print()
disp = ConfusionMatrixDisplay(confusion_matrix=conf_mat, display_labels=["Low␣
↪Damage 1","Medium Damage 2","High Damage 3"])
disp.plot(cmap=plt.cm.Blues)
```

```
Model Name: XGBTUNED
training: 0.7826170376055257
test: 0.7455152433759905
f1: 0.6940598473835476
recall score: 0.6687320578550456
precision score: 0.7333240295557952
[[ 2626  2305    67]
 [ 1056 25259  3459]
 [   85  6292 10972]]
```

[30]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x185255792b0>
```



[89]:
```
rf =RandomForestClassifier(max_features = None,
                           max_depth = 45,
                           min_samples_split = 3,
                           min_samples_leaf = 30,
                           random_state=42)
#rf.fit(X_train, y_train)
rf.fit(X_train, y_train)
results = list(zip(X, rf.feature_importances_))
importance = pd.DataFrame(results, columns = ["Feature", "Importance"])
importance = importance.sort_values(by="Importance", ascending=False)
y_pred = rf.predict(X_test)
f1 = f1_score(y_test, y_pred, average="macro")
recall_scores =recall_score(y_test, y_pred, average="macro")
prec_scores =precision_score(y_test, y_pred, average="macro")
print("training:",rf.score(X_train, y_train))
print("test:",rf.score(X_test, y_test))
print("f1:",f1)
print("recall score:",recall_scores)
print("precision score:",prec_scores)
conf_mat = confusion_matrix(y_test, y_pred)
print(conf_mat)
print()
importance
```

```
training: 0.694459900230238
```

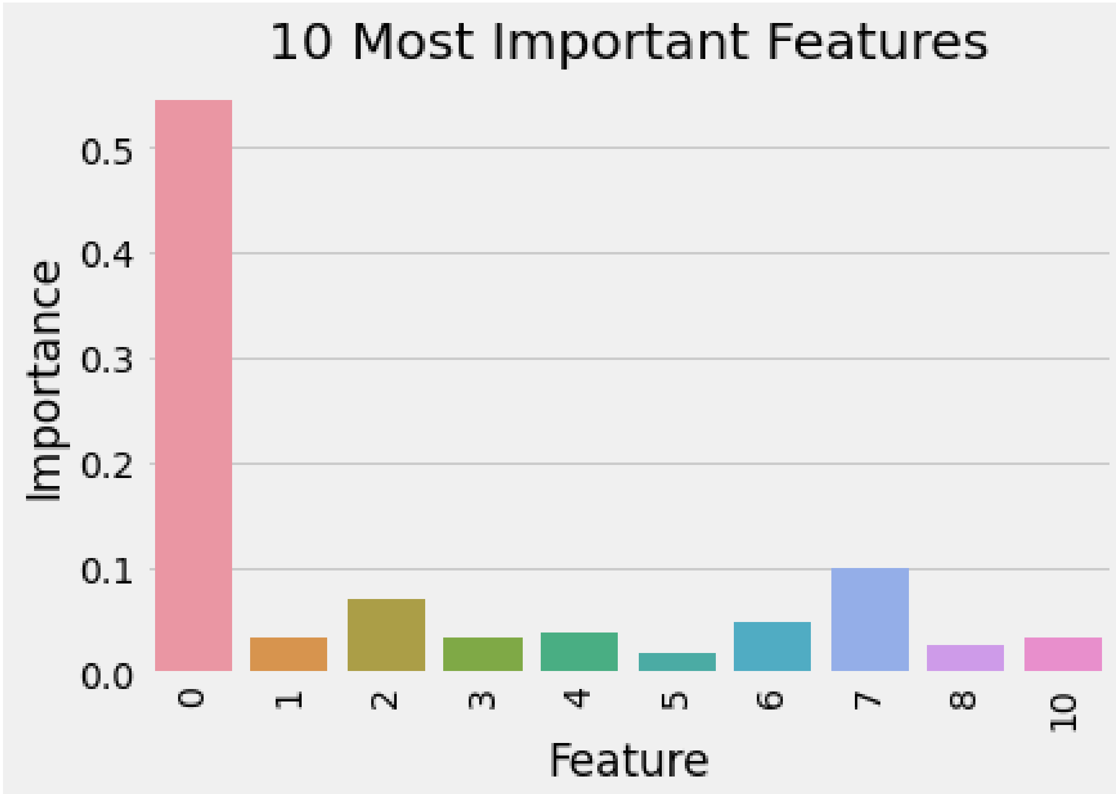
```
test: 0.6844458087910823
f1: 0.6091764654319819
recall score: 0.5816783493966118
precision score: 0.6638350970685539
[[ 1929  2995    81]
 [ 1120 24335  4231]
 [   84  7936  9410]]
```

[89]:

	Feature	Importance
0	0	0.543381
7	7	0.098000
2	2	0.070475
6	6	0.047209
4	4	0.037765
10	10	0.033927
1	1	0.033095
3	3	0.032515
8	8	0.026631
5	5	0.018038
12	12	0.017877
17	17	0.011083
11	11	0.008808
13	13	0.005925
16	16	0.004434
14	14	0.004069
15	15	0.002311
9	9	0.002152
18	18	0.001863
19	19	0.000442

[90]:

```
importance_10 = importance.head(10)
plot = sns.barplot(x=columns[importance_10["Feature"]], y=importance_10["Importance"])
plot.set_xticklabels(plot.get_xticklabels(), rotation=90)
plt.title("10 Most Important Features")
plt.show()
```


10 Most Important Features
Importance
0.5
0.4
0.3
0.2
0.1
0.0
0
1
2
3
4
5
6
7
8
10
Feature