In [1363]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

In [1364]:
```
df=pd.read_csv("sample_service.xlsx - Sheet1.csv")
df_orginal=df.copy()
```

In [1365]:
```
df
```

Out[1365]:

	ID	ID_status	active	count_reassign	count_opening	count_updated	ID_caller	op
0	INC0000045	New	True	0	0	0	Caller 2403	C
1	INC0000045	Resolved	True	0	0	2	Caller 2403	C
2	INC0000045	Resolved	True	0	0	3	Caller 2403	C
3	INC0000045	Closed	False	0	0	4	Caller 2403	C
4	INC0000047	New	True	0	0	0	Caller 2403	C
...	...	...	...	...	...	...	...	
344	INC0000134	New	True	0	0	2	Caller 2471	C
345	INC0000134	New	True	1	0	1	Caller 2471	C
346	INC0000134	Active	True	1	0	2	Caller 2471	C
347	INC0000134	Resolved	True	1	0	3	Caller 2471	C
348	INC0000134	Closed	False	1	0	4	Caller 2471	C

349 rows × 25 columns

In [1366]:
```
df.columns
```

Out[1366]:
```
Index(['ID', 'ID_status', 'active', 'count_reassign', 'count_opening',
       'count_updated', 'ID_caller', 'opened_by', 'opened_time', 'Created_b
y',
       'created_at', 'updated_by', 'updated_at', 'type_contact', 'location',
       'Category Id', 'user_symptom', 'Support_group', 'support_incharge',
       'Doc_knowledge', 'confirmation_check', 'impact', 'notify', 'problem_I
D',
       'change_request'],
      dtype='object')
```

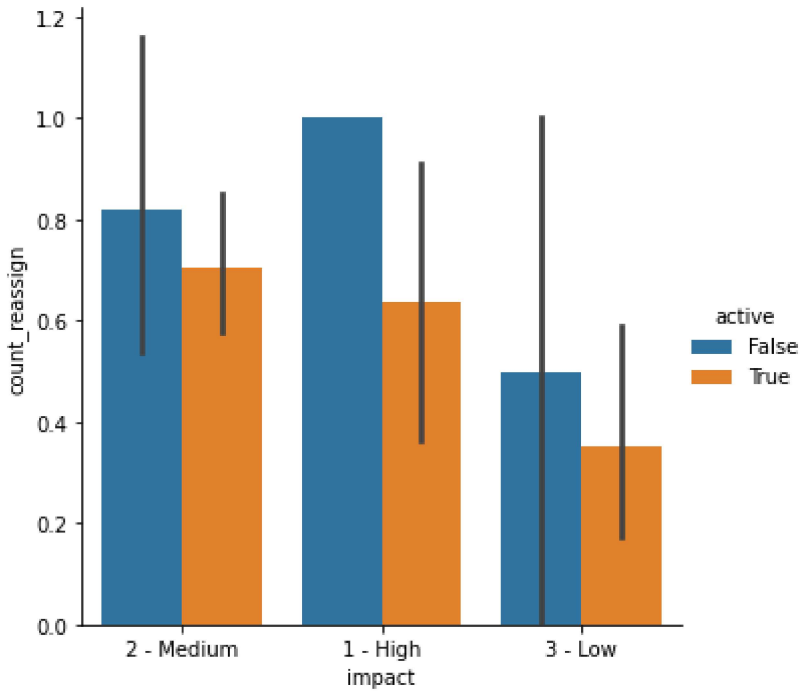
In [1367]:
```
# Import label encoder
from sklearn import preprocessing

# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()

# Encode labels in column 'species'.
df['impact']= label_encoder.fit_transform(df['impact'])
```

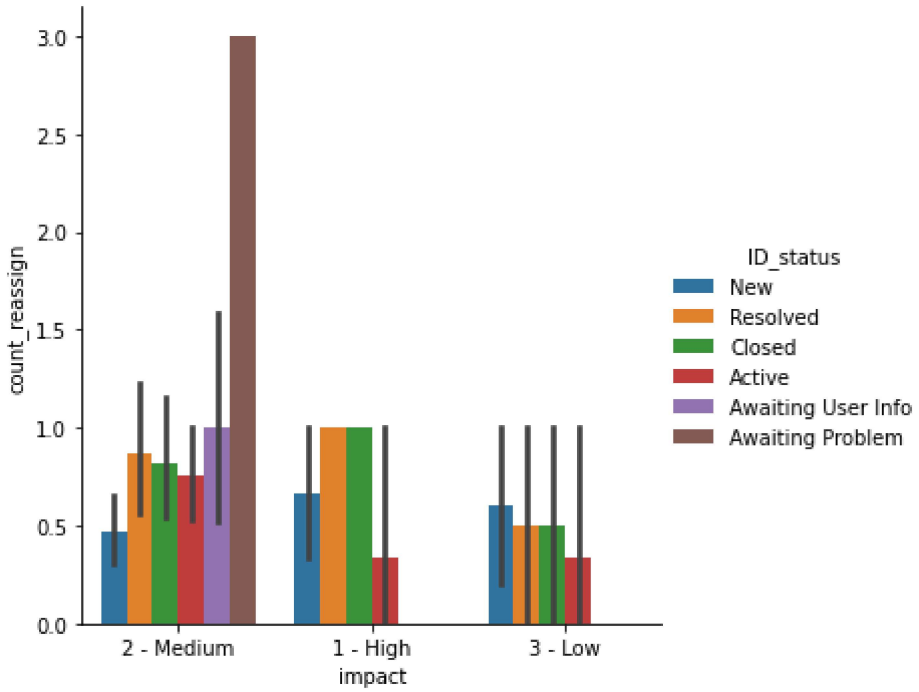
In [1368]:
```
sns.catplot(x="impact", y="count_reassign", hue="active", kind="bar", data=df_
orginal)
```

Out[1368]:
```
<seaborn.axisgrid.FacetGrid at 0x1e2b678ca90>
```



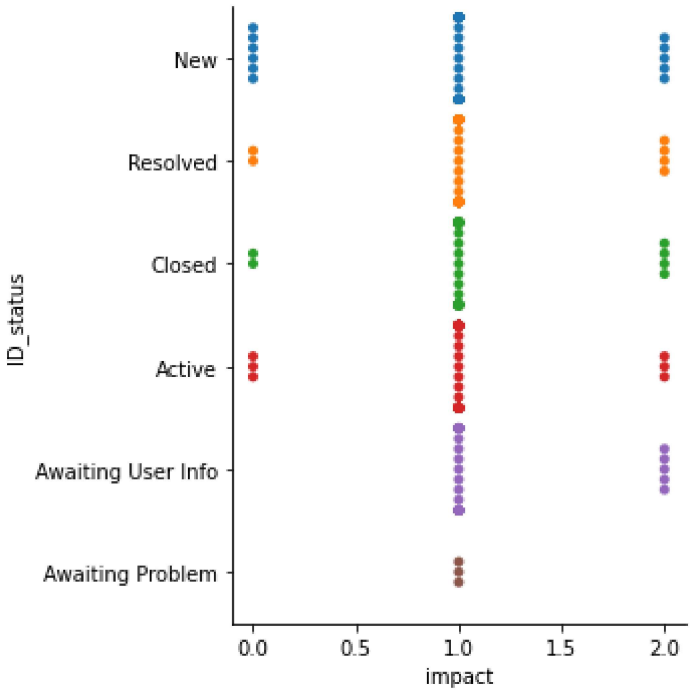
In [1369]:
```
sns.catplot(x="impact", y="count_reassign", hue="ID_status", kind="bar", data=df_orginal)
```

Out[1369]: <seaborn.axisgrid.FacetGrid at 0x1e2b6aed070>



In [1370]:
```
sns.catplot(x="impact", y="ID_status", kind="swarm", data=df)
```

Out[1370]: <seaborn.axisgrid.FacetGrid at 0x1e2b85f3160>



In [1371]:
```
df1=df.replace('?', np.nan) #replacing ? with nan
```

In [1372]:
```
df1
```

Out[1372]:

	ID	ID_status	active	count_reassign	count_opening	count_updated	ID_caller	op
0	INC0000045	New	True	0	0	0	Caller 2403	C
1	INC0000045	Resolved	True	0	0	2	Caller 2403	C
2	INC0000045	Resolved	True	0	0	3	Caller 2403	C
3	INC0000045	Closed	False	0	0	4	Caller 2403	C
4	INC0000047	New	True	0	0	0	Caller 2403	C
...	...	...	...	...	...	...	...	
344	INC0000134	New	True	0	0	2	Caller 2471	C
345	INC0000134	New	True	1	0	1	Caller 2471	C
346	INC0000134	Active	True	1	0	2	Caller 2471	C
347	INC0000134	Resolved	True	1	0	3	Caller 2471	C
348	INC0000134	Closed	False	1	0	4	Caller 2471	C

349 rows × 25 columns

In [1373]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 349 entries, 0 to 348
Data columns (total 25 columns):
ID                    349 non-null object
ID_status             349 non-null object
active                349 non-null bool
count_reassign        349 non-null int64
count_opening         349 non-null int64
count_updated         349 non-null int64
ID_caller             349 non-null object
opened_by             349 non-null object
opened_time           349 non-null object
Created_by            287 non-null object
created_at            287 non-null object
updated_by            349 non-null object
updated_at            349 non-null object
type_contact          349 non-null object
location              349 non-null object
Category Id           349 non-null object
user_symptom          304 non-null object
Support_group         349 non-null object
support_incharge      194 non-null object
Doc_knowledge         349 non-null bool
confirmation_check    349 non-null bool
impact                349 non-null int64
notify                349 non-null object
problem_ID            82 non-null object
change_request        6 non-null object
dtypes: bool(3), int64(4), object(18)
memory usage: 61.1+ KB
```

In [1374]:
```
df1_copy=df1.copy()
```

In [1375]:
```
df1.shape
```

Out[1375]:
```
(349, 25)
```

In [1376]:
```
df1=df1.iloc[:,:-2] #last 2 coloumn eliminated,because so manynull values
```

In [1377]:
```
df1.shape
```

Out[1377]:
```
(349, 23)
```

In [1378]:
```
df1.columns
```

Out[1378]:
```
Index(['ID', 'ID_status', 'active', 'count_reassign', 'count_opening',
       'count_updated', 'ID_caller', 'opened_by', 'opened_time', 'Created_b
y',
       'created_at', 'updated_by', 'updated_at', 'type_contact', 'location',
       'Category Id', 'user_symptom', 'Support_group', 'support_incharge',
       'Doc_knowledge', 'confirmation_check', 'impact', 'notify'],
      dtype='object')
```

In [1379]:
```
df1.isnull().sum()
```

Out[1379]:
```
ID                    0
ID_status             0
active                0
count_reassign        0
count_opening         0
count_updated         0
ID_caller             0
opened_by             0
opened_time           0
Created_by           62
created_at           62
updated_by            0
updated_at            0
type_contact          0
location              0
Category Id           0
user_symptom         45
Support_group         0
support_incharge    155
Doc_knowledge         0
confirmation_check    0
impact                0
notify                0
dtype: int64
```

In [1380]:
```
df1['Created_by'].fillna(method='ffill',inplace= True)
```

In [1381]:
```
df1['created_at'].fillna(method='ffill',inplace= True)
df1['user_symptom'].fillna(method='ffill',inplace= True)
```

In [1382]:
```
df1.isnull().sum()
```

Out[1382]:
```
ID                    0
ID_status             0
active                0
count_reassign        0
count_opening         0
count_updated         0
ID_caller             0
opened_by             0
opened_time           0
Created_by            0
created_at            0
updated_by            0
updated_at            0
type_contact          0
location              0
Category Id           0
user_symptom          0
Support_group         0
support_incharge    155
Doc_knowledge         0
confirmation_check    0
impact                0
notify                0
dtype: int64
```

In [1383]:
```
new=df1.copy()
```

In [1384]:
```
df1.drop('support_incharge',axis='columns', inplace=True)
```

In [1385]:
```
df1.isnull().sum()
```

Out[1385]:
```
ID                  0
ID_status           0
active              0
count_reassign      0
count_opening       0
count_updated       0
ID_caller           0
opened_by           0
opened_time         0
Created_by          0
created_at          0
updated_by          0
updated_at          0
type_contact        0
location            0
Category Id         0
user_symptom        0
Support_group       0
Doc_knowledge       0
confirmation_check  0
impact              0
notify              0
dtype: int64
```

In [1386]:
```
df2=df1.apply(label_encoder.fit_transform)
```

In [1387]:
```
x=pd.concat([df2,df1_copy['support_incharge']],axis=1,ignore_index=False)
x.reset_index(drop=True)
```

Out[1387]:

	ID	ID_status	active	count_reassign	count_opening	count_updated	ID_caller	opened_by
0	0	4	1	0	0	0	8	8
1	0	5	1	0	0	2	8	8
2	0	5	1	0	0	3	8	8
3	0	3	0	0	0	4	8	8
4	1	4	1	0	0	0	8	3
...	...	...	...	...	...	...	...	...
344	55	4	1	0	0	2	9	6
345	55	4	1	1	0	1	9	6
346	55	0	1	1	0	2	9	6
347	55	5	1	1	0	3	9	6
348	55	3	0	1	0	4	9	6

349 rows × 23 columns

In [ ]:

In [1388]:
```
test=x[x['support_incharge'].isna()]
test1=test.reset_index(drop=True)
test1
```

Out[1388]:

	ID	ID_status	active	count_reassign	count_opening	count_updated	ID_caller	opened_by
0	0	4	1	0	0	0	8	8
1	0	5	1	0	0	2	8	8
2	0	5	1	0	0	3	8	8
3	0	3	0	0	0	4	8	8
4	2	4	1	0	0	0	33	8
...	...	...	...	...	...	...	...	...
150	49	4	1	0	0	1	21	1
151	49	5	1	0	0	2	21	1
152	49	3	0	0	0	3	21	1
153	52	4	1	0	0	2	18	7
154	52	4	1	1	0	1	18	7

155 rows × 23 columns

In [1389]:
```
train=x[x['support_incharge'].notnull()]
train1=train.reset_index(drop=True)
train1
```

Out[1389]:

	ID	ID_status	active	count_reassign	count_opening	count_updated	ID_caller	opened_by
0	1	4	1	0	0	0	8	3
1	1	0	1	1	0	1	8	3
2	1	0	1	1	0	2	8	3
3	1	0	1	1	0	3	8	3
4	1	0	1	1	0	4	8	3
...	...	...	...	...	...	...	...	...
189	55	4	1	0	0	2	9	6
190	55	4	1	1	0	1	9	6
191	55	0	1	1	0	2	9	6
192	55	5	1	1	0	3	9	6
193	55	3	0	1	0	4	9	6

194 rows × 23 columns

In [1390]:
```
# Encode labels in column 'species'.
train1['support_incharge']= label_encoder.fit_transform(train1['support_incharge'])
```

In [1391]:
```
train_copy=train1.copy()
```

In [1392]:
```
train1.shape
```

Out[1392]: (194, 23)

In [1393]:
```
x_train=train1.iloc[:,0:22]
x_train
```

Out[1393]:

	ID	ID_status	active	count_reassign	count_opening	count_updated	ID_caller	opened_by
0	1	4	1	0	0	0	8	3
1	1	0	1	1	0	1	8	3
2	1	0	1	1	0	2	8	3
3	1	0	1	1	0	3	8	3
4	1	0	1	1	0	4	8	3
...	...	...	...	...	...	...	...	...
189	55	4	1	0	0	2	9	6
190	55	4	1	1	0	1	9	6
191	55	0	1	1	0	2	9	6
192	55	5	1	1	0	3	9	6
193	55	3	0	1	0	4	9	6

194 rows × 22 columns

In [1394]:
```
y_train=train1.iloc[:,-1:]
y_train
```

Out[1394]:

	support_incharge
0	21
1	17
2	17
3	17
4	17
...	...
189	12
190	12
191	9
192	9
193	9

194 rows × 1 columns

In [1395]:
```
x_test=test1.iloc[:,0:22]
x_test.reset_index(drop=True)
x_test
```

Out[1395]:

	ID	ID_status	active	count_reassign	count_opening	count_updated	ID_caller	opened_by
0	0	4	1	0	0	0	8	8
1	0	5	1	0	0	2	8	8
2	0	5	1	0	0	3	8	8
3	0	3	0	0	0	4	8	8
4	2	4	1	0	0	0	33	8
...	...	...	...	...	...	...	...	...
150	49	4	1	0	0	1	21	1
151	49	5	1	0	0	2	21	1
152	49	3	0	0	0	3	21	1
153	52	4	1	0	0	2	18	7
154	52	4	1	1	0	1	18	7

155 rows × 22 columns

In [ ]:
```
```

In [1396]:
```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression()
classifier.fit(x_train,y_train)
```

```
C:\Users\ashiq\anaconda3\lib\site-packages\sklearn\utils\validation.py:760: D
ataConversionWarning: A column-vector y was passed when a 1d array was expect
ed. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\ashiq\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
938: 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-regres
sion
  n_iter_i = _check_optimize_result(
```

Out[1396]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [1397]:
```
y_pred = classifier.predict(x_test)
y_pred
```

Out[1397]:
```
array([20, 20, 17, 20, 18, 18,  3, 18, 18, 10, 10, 10, 10, 10, 10, 10, 10,
       17, 17, 17, 17, 17, 17, 17, 17, 17,  8,  8,  8, 18, 14, 14, 14, 17,
       17,  5,  8, 17,  5,  5,  5, 17, 17, 17, 12, 12, 12, 12, 12,  5,  6,
        6,  6,  6,  6,  6,  6,  6,  6,  6, 10, 10, 10, 10, 11, 14, 10, 14,
       14,  6,  6,  6, 17, 17, 17, 17, 17, 17, 10, 10, 10, 10,  8,  8,  8,
       22, 16, 13, 13,  6,  6,  6,  7, 21, 22, 12, 12, 12, 21, 21, 21, 12,
       21, 21, 21, 12, 21, 21, 23, 23, 23, 23, 23, 23, 23, 23,  2,  2,  2,
       21, 10, 10, 10, 10, 21, 21, 21, 21, 21, 21, 21, 21, 21,  4,  4,  4,
        4, 13, 23, 13, 23, 23,  1,  1,  1, 21,  2, 21, 21,  1,  1,  4,  4,
        5,  9])
```

In [1398]:
```
y_pred.shape
```

Out[1398]:
```
(155,)
```

In [1399]:
```
y_test = pd.DataFrame(y_pred, columns = ['support_incharge'])
y_test.reset_index(drop=True)
y_test
```

Out[1399]:

	support_incharge
0	20
1	20
2	17
3	20
4	18
...	...
150	1
151	4
152	4
153	5
154	9

155 rows × 1 columns

In [1400]:
```
x_test
```

Out[1400]:

	ID	ID_status	active	count_reassign	count_opening	count_updated	ID_caller	opened_by
0	0	4	1	0	0	0	8	8
1	0	5	1	0	0	2	8	8
2	0	5	1	0	0	3	8	8
3	0	3	0	0	0	4	8	8
4	2	4	1	0	0	0	33	8
...	...	...	...	...	...	...	...	...
150	49	4	1	0	0	1	21	1
151	49	5	1	0	0	2	21	1
152	49	3	0	0	0	3	21	1
153	52	4	1	0	0	2	18	7
154	52	4	1	1	0	1	18	7

155 rows × 22 columns

In [1401]:
```
test_new=pd.concat([x_test,y_test], axis =1)
```

In [1402]:
```
test_new
```

Out[1402]:

	ID	ID_status	active	count_reassign	count_opening	count_updated	ID_caller	opened_by
0	0	4	1	0	0	0	8	8
1	0	5	1	0	0	2	8	8
2	0	5	1	0	0	3	8	8
3	0	3	0	0	0	4	8	8
4	2	4	1	0	0	0	33	8
...	...	...	...	...	...	...	...	...
150	49	4	1	0	0	1	21	1
151	49	5	1	0	0	2	21	1
152	49	3	0	0	0	3	21	1
153	52	4	1	0	0	2	18	7
154	52	4	1	1	0	1	18	7

155 rows × 23 columns

In [1403]:
```
train_copy
```

Out[1403]:

	ID	ID_status	active	count_reassign	count_opening	count_updated	ID_caller	opened_by
0	1	4	1	0	0	0	8	3
1	1	0	1	1	0	1	8	3
2	1	0	1	1	0	2	8	3
3	1	0	1	1	0	3	8	3
4	1	0	1	1	0	4	8	3
...	...	...	...	...	...	...	...	...
189	55	4	1	0	0	2	9	6
190	55	4	1	1	0	1	9	6
191	55	0	1	1	0	2	9	6
192	55	5	1	1	0	3	9	6
193	55	3	0	1	0	4	9	6

194 rows × 23 columns

In [1404]:
```
vertical_stack = pd.concat([train_copy, test_new], axis=0)
```

In [1405]:
```
vertical_stack
```

Out[1405]:

	ID	ID_status	active	count_reassign	count_opening	count_updated	ID_caller	opened_by
0	1	4	1	0	0	0	8	3
1	1	0	1	1	0	1	8	3
2	1	0	1	1	0	2	8	3
3	1	0	1	1	0	3	8	3
4	1	0	1	1	0	4	8	3
...	...	...	...	...	...	...	...	...
150	49	4	1	0	0	1	21	1
151	49	5	1	0	0	2	21	1
152	49	3	0	0	0	3	21	1
153	52	4	1	0	0	2	18	7
154	52	4	1	1	0	1	18	7

349 rows × 23 columns

In [1406]:
```
vertical_stack.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 349 entries, 0 to 154
Data columns (total 23 columns):
ID                    349 non-null int32
ID_status             349 non-null int32
active                349 non-null int64
count_reassign        349 non-null int64
count_opening         349 non-null int64
count_updated         349 non-null int64
ID_caller             349 non-null int32
opened_by             349 non-null int32
opened_time           349 non-null int32
Created_by            349 non-null int32
created_at            349 non-null int32
updated_by            349 non-null int32
updated_at            349 non-null int32
type_contact          349 non-null int32
location              349 non-null int32
Category Id           349 non-null int32
user_symptom          349 non-null int32
Support_group         349 non-null int32
Doc_knowledge         349 non-null int64
confirmation_check    349 non-null int64
impact                349 non-null int64
notify                349 non-null int32
support_incharge      349 non-null int32
dtypes: int32(16), int64(7)
memory usage: 43.6 KB
```

In [1407]:
```
df_new=vertical_stack.reset_index(drop=True)
```

In [1408]:
```
df_new
```

Out[1408]:

	ID	ID_status	active	count_reassign	count_opening	count_updated	ID_caller	opened_by
0	1	4	1	0	0	0	8	3
1	1	0	1	1	0	1	8	3
2	1	0	1	1	0	2	8	3
3	1	0	1	1	0	3	8	3
4	1	0	1	1	0	4	8	3
...	...	...	...	...	...	...	...	...
344	49	4	1	0	0	1	21	1
345	49	5	1	0	0	2	21	1
346	49	3	0	0	0	3	21	1
347	52	4	1	0	0	2	18	7
348	52	4	1	1	0	1	18	7

349 rows × 23 columns

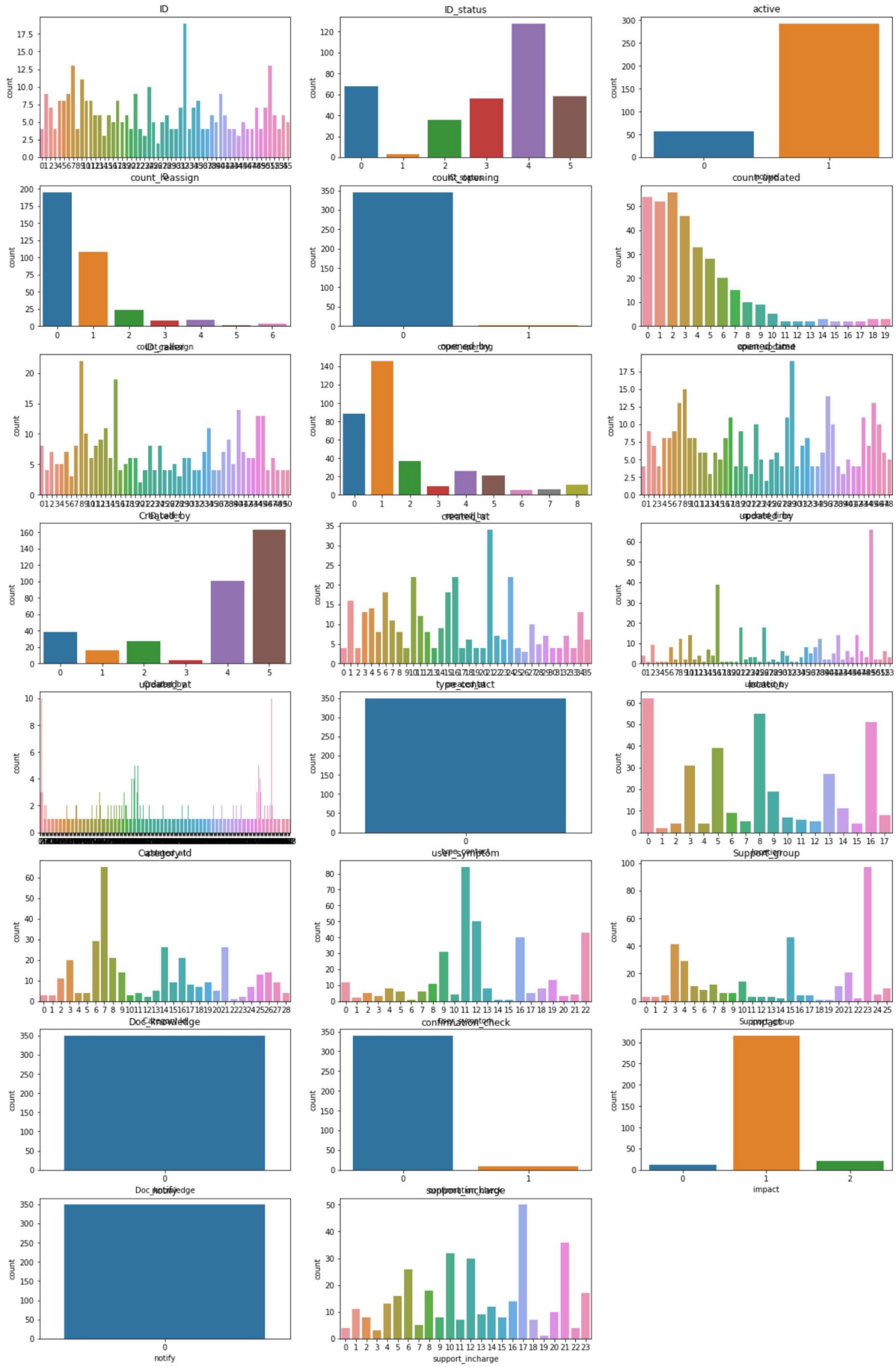
In [1409]:
```
#sns.pairplot(df_new)
```

In [1410]:
```
df_new.columns
```

Out[1410]:
```
Index(['ID', 'ID_status', 'active', 'count_reassign', 'count_opening',
       'count_updated', 'ID_caller', 'opened_by', 'opened_time', 'Created_b
y',
       'created_at', 'updated_by', 'updated_at', 'type_contact', 'location',
       'Category Id', 'user_symptom', 'Support_group', 'Doc_knowledge',
       'confirmation_check', 'impact', 'notify', 'support_incharge'],
      dtype='object')
```

In [1411]:
```
fig = plt.figure(figsize=(20,40))
z=['ID', 'ID_status', 'active', 'count_reassign', 'count_opening',
       'count_updated', 'ID_caller', 'opened_by', 'opened_time', 'Created_by',
       'created_at', 'updated_by', 'updated_at', 'type_contact', 'location',
       'Category Id', 'user_symptom', 'Support_group', 'Doc_knowledge',
       'confirmation_check', 'impact', 'notify', 'support_incharge']
a = 10  # number of rows
b = 3  # number of columns
c = 1 # initialize plot counter
for i in z: #it will take column names
    plt.subplot(a, b, c)
    plt.title('{}'.format(i, a, b, c))
    plt.xlabel(i)
    sns.countplot(df_new[i])
    c = c + 1

plt.show()
```


ID
ID_status
active
count_reassign
count_opening
count_updated
opened_by
created_by
created_at
updated_by
updated_at
type_contact
location
category_id
user_symptom
Support_group
Doc_knowledge
confirmation_check
impact
notify
support_incharge
count

In [1412]:
```
df_new.columns
```

Out[1412]:
```
Index(['ID', 'ID_status', 'active', 'count_reassign', 'count_opening',
       'count_updated', 'ID_caller', 'opened_by', 'opened_time', 'Created_b
y',
       'created_at', 'updated_by', 'updated_at', 'type_contact', 'location',
       'Category Id', 'user_symptom', 'Support_group', 'Doc_knowledge',
       'confirmation_check', 'impact', 'notify', 'support_incharge'],
      dtype='object')
```

Univariate Feature Selection

In [1413]:
```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from numpy import set_printoptions
```

In [1414]:
```
data=df_new.copy()
data
```

Out[1414]:

	ID	ID_status	active	count_reassign	count_opening	count_updated	ID_caller	opened_by
0	1	4	1	0	0	0	8	3
1	1	0	1	1	0	1	8	3
2	1	0	1	1	0	2	8	3
3	1	0	1	1	0	3	8	3
4	1	0	1	1	0	4	8	3
...	...	...	...	...	...	...	...	...
344	49	4	1	0	0	1	21	1
345	49	5	1	0	0	2	21	1
346	49	3	0	0	0	3	21	1
347	52	4	1	0	0	2	18	7
348	52	4	1	1	0	1	18	7

349 rows × 23 columns

In [1415]:
```
X = data[['ID', 'ID_status', 'active', 'count_reassign', 'count_opening',
       'count_updated', 'ID_caller', 'opened_by', 'opened_time', 'Created_by',
       'created_at', 'updated_by', 'updated_at', 'type_contact', 'location',
       'Category Id', 'user_symptom', 'Support_group', 'Doc_knowledge',
       'confirmation_check',  'notify', 'support_incharge']]
X
```

Out[1415]:

	ID	ID_status	active	count_reassign	count_opening	count_updated	ID_caller	opened_by
0	1	4	1	0	0	0	8	3
1	1	0	1	1	0	1	8	3
2	1	0	1	1	0	2	8	3
3	1	0	1	1	0	3	8	3
4	1	0	1	1	0	4	8	3
...	...	...	...	...	...	...	...	...
344	49	4	1	0	0	1	21	1
345	49	5	1	0	0	2	21	1
346	49	3	0	0	0	3	21	1
347	52	4	1	0	0	2	18	7
348	52	4	1	1	0	1	18	7

349 rows × 22 columns

In [1416]:
```
Y = data[['impact']]
```

In [1417]:
```
Y
```

Out[1417]:

	impact
0	1
1	1
2	1
3	1
4	1
...	...
344	1
345	1
346	1
347	1
348	1

349 rows × 1 columns

In [1418]:
```
# feature extraction
test = SelectKBest(score_func=chi2, k=15)
fit = test.fit(X, Y)
# summarize scores
set_printoptions(precision=3)
q1=fit.scores_
print(fit.scores_)
features = fit.transform(X)
```

```
[4.243e+01 4.201e-02 2.434e-02 3.298e+00 3.238e-01 4.263e+00 3.673e+01
 6.890e+01 3.497e+01 7.744e+00 2.185e+01 3.747e+01 1.913e+02       nan
 1.073e+02 2.006e+01 9.117e+01 4.099e+01       nan 1.137e+01       nan
 4.306e+01]
```

In [1419]:
```
q2=['ID', 'ID_status', 'active', 'count_reassign', 'count_opening',
       'count_updated', 'ID_caller', 'opened_by', 'opened_time', 'Created_by',
       'created_at', 'updated_by', 'updated_at', 'type_contact', 'location',
       'Category Id', 'user_symptom', 'Support_group', 'Doc_knowledge',
       'confirmation_check',  'notify', 'support_incharge']
q1 = pd.DataFrame(q1, columns = ['scores_'])
```

In [1420]:
```
q2=pd.DataFrame(q2, columns = ['attributes'])
```

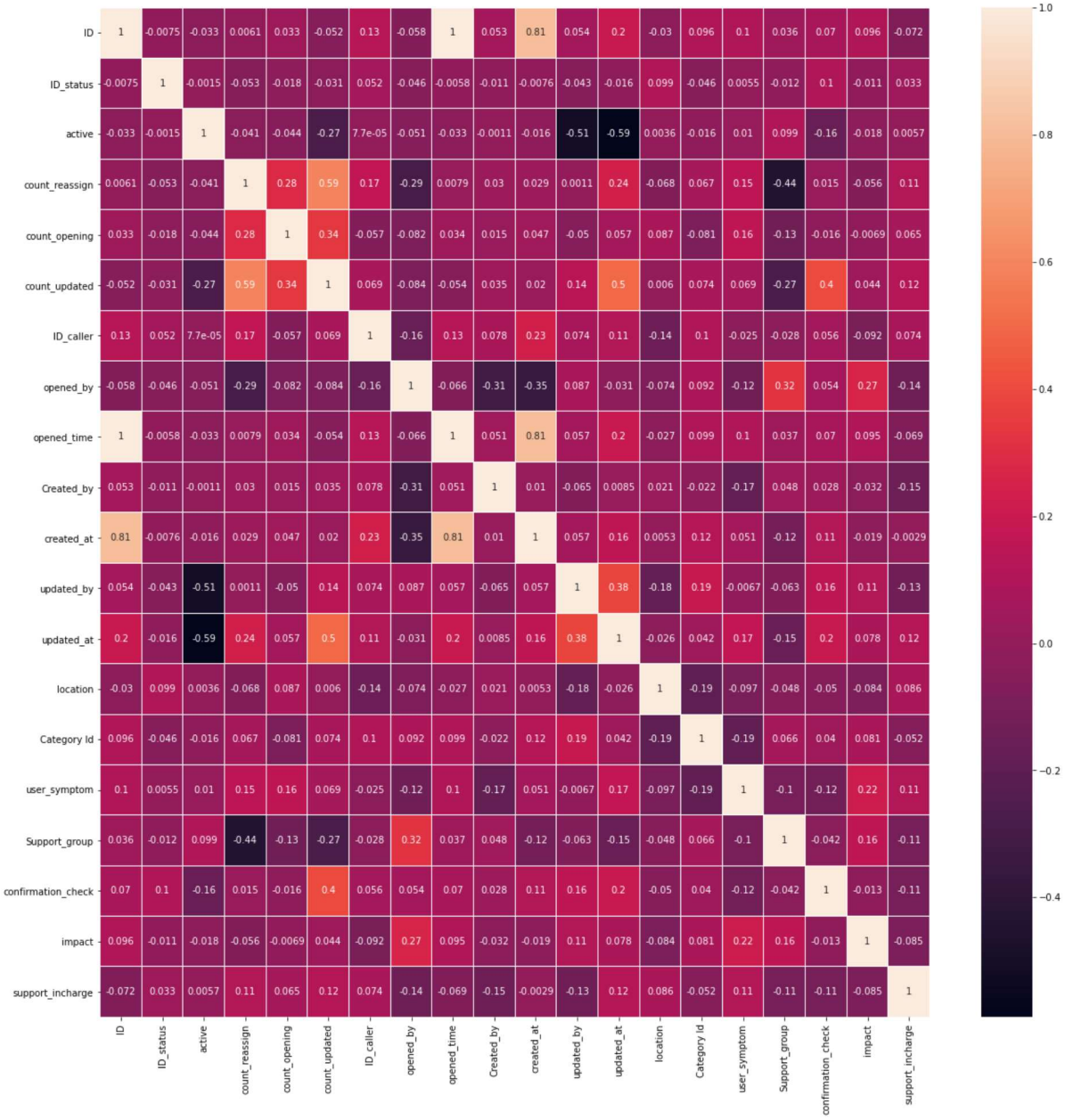
In [1421]:
```
q=pd.concat([q1,q2],axis=1,ignore_index=False)
q.sort_values('scores_',ascending = False)
```

Out[1421]:

	scores_	attributes
12	191.297636	updated_at
14	107.293447	location
16	91.167660	user_symptom
7	68.895413	opened_by
21	43.062580	support_incharge
0	42.425670	ID
17	40.993456	Support_group
11	37.468671	updated_by
6	36.726189	ID_caller
8	34.973865	opened_time
10	21.847590	created_at
15	20.061076	Category Id
19	11.374652	confirmation_check
9	7.743826	Created_by
5	4.262502	count_updated
3	3.298485	count_reassign
4	0.323810	count_opening
1	0.042007	ID_status
2	0.024337	active
13	NaN	type_contact
18	NaN	Doc_knowledge
20	NaN	notify

In [1422]:
```
df_new1=df_new.drop(columns=['type_contact', 'Doc_knowledge','notify'])
```

In [1423]:
```
sns.heatmap(df_new1.corr(), annot=True, linewidth=.02)
fig=plt.gcf()
fig.set_size_inches(20,20)
plt.show()
```



Recursive Feature Elimination

In [1424]:
```
Y
```

Out[1424]:

	impact
0	1
1	1
2	1
3	1
4	1
...	...
344	1
345	1
346	1
347	1
348	1

349 rows × 1 columns

In [1425]:
```
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression

# feature extraction
model = LogisticRegression(max_iter=400)
rfe = RFE(model, 3)
fit = rfe.fit(X, Y)
```

```
C:\Users\ashiq\anaconda3\lib\site-packages\sklearn\utils\validation.py:760: D
ataConversionWarning: A column-vector y was passed when a 1d array was expect
ed. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\ashiq\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
938: 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-regres
sion
  n_iter_i = _check_optimize_result(
C:\Users\ashiq\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
938: 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-regres
sion
  n_iter_i = _check_optimize_result(
C:\Users\ashiq\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
938: 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-regres
sion
  n_iter_i = _check_optimize_result(
C:\Users\ashiq\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
938: 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-regres
sion
  n_iter_i = _check_optimize_result(
C:\Users\ashiq\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
938: 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-regres
sion
  n_iter_i = _check_optimize_result(
C:\Users\ashiq\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
938: 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-regres
sion
  n_iter_i = _check_optimize_result(
C:\Users\ashiq\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
938: 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-regres
sion
  n_iter_i = _check_optimize_result(
C:\Users\ashiq\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
938: 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-regres
sion
  n_iter_i = _check_optimize_result(
C:\Users\ashiq\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
938: 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-regres
sion
  n_iter_i = _check_optimize_result(
C:\Users\ashiq\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
938: 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-regres
sion
  n_iter_i = _check_optimize_result(
C:\Users\ashiq\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
938: 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-regres
sion
  n_iter_i = _check_optimize_result(
```

```
C:\Users\ashiq\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
938: 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-regres
sion
  n_iter_i = _check_optimize_result(
C:\Users\ashiq\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
938: 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-regres
sion
  n_iter_i = _check_optimize_result(
C:\Users\ashiq\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
938: 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-regres
sion
  n_iter_i = _check_optimize_result(
```

In [1426]:

```
fit.support_
```

Out[1426]:

```
array([False, False, False,  True, False, False, False,  True, False,
        True, False, False, False, False, False, False, False, False,
       False, False, False, False])
```

In [1427]:

```
# Feature Ranking:
q3=fit.ranking_
q3 = pd.DataFrame(q3, columns = ['rankings_'])
```

In [1428]:
```
q=pd.concat([q3,q2],axis=1,ignore_index=False)
q.sort_values('rankings_',ascending = False)
```

Out[1428]:

	rankings_	attributes
18	20	Doc_knowledge
13	19	type_contact
20	18	notify
12	17	updated_at
11	16	updated_by
17	15	Support_group
10	14	created_at
4	13	count_opening
0	12	ID
8	11	opened_time
19	10	confirmation_check
6	9	ID_caller
15	8	Category Id
5	7	count_updated
21	6	support_incharge
16	5	user_symptom
2	4	active
1	3	ID_status
14	2	location
9	1	Created_by
7	1	opened_by
3	1	count_reassign

In [1429]:
```
from sklearn.tree import  DecisionTreeClassifier
model = DecisionTreeClassifier()
model.fit(X, Y)
q4=model.feature_importances_
```

In [1430]:
```
q4 = pd.DataFrame(q4, columns = ['importance'])
```

In [1431]:
```
q_new=pd.concat([q4,q2],axis=1,ignore_index=False)
q_new.sort_values('importance',ascending = False)
```

Out[1431]:

	importance	attributes
21	0.344949	support_incharge
16	0.208183	user_symptom
8	0.145381	opened_time
0	0.141092	ID
6	0.110082	ID_caller
5	0.050313	count_updated
4	0.000000	count_opening
14	0.000000	location
20	0.000000	notify
19	0.000000	confirmation_check
18	0.000000	Doc_knowledge
17	0.000000	Support_group
2	0.000000	active
15	0.000000	Category Id
13	0.000000	type_contact
12	0.000000	updated_at
1	0.000000	ID_status
10	0.000000	created_at
9	0.000000	Created_by
3	0.000000	count_reassign
7	0.000000	opened_by
11	0.000000	updated_by

In [1432]:
```
df_latest=df_new.copy()
```

In [1433]:
```
df_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 349 entries, 0 to 348
Data columns (total 23 columns):
ID                    349 non-null int32
ID_status             349 non-null int32
active                349 non-null int64
count_reassign        349 non-null int64
count_opening         349 non-null int64
count_updated         349 non-null int64
ID_caller             349 non-null int32
opened_by             349 non-null int32
opened_time           349 non-null int32
Created_by            349 non-null int32
created_at            349 non-null int32
updated_by            349 non-null int32
updated_at            349 non-null int32
type_contact          349 non-null int32
location              349 non-null int32
Category Id           349 non-null int32
user_symptom          349 non-null int32
Support_group         349 non-null int32
Doc_knowledge         349 non-null int64
confirmation_check    349 non-null int64
impact                349 non-null int64
notify                349 non-null int32
support_incharge      349 non-null int32
dtypes: int32(16), int64(7)
memory usage: 41.0 KB
```

In [1434]:
```
o_p=df_new['impact']
```

In [1435]:
```
o_p
```

Out[1435]:
```
0      1
1      1
2      1
3      1
4      1
      ..
344    1
345    1
346    1
347    1
348    1
Name: impact, Length: 349, dtype: int64
```

In [1436]:
```
i_o=df_new.drop(columns=['impact'])
i_o
```

Out[1436]:

	ID	ID_status	active	count_reassign	count_opening	count_updated	ID_caller	opened_by
0	1	4	1	0	0	0	8	3
1	1	0	1	1	0	1	8	3
2	1	0	1	1	0	2	8	3
3	1	0	1	1	0	3	8	3
4	1	0	1	1	0	4	8	3
...	...	...	...	...	...	...	...	...
344	49	4	1	0	0	1	21	1
345	49	5	1	0	0	2	21	1
346	49	3	0	0	0	3	21	1
347	52	4	1	0	0	2	18	7
348	52	4	1	1	0	1	18	7

349 rows × 22 columns

In [1437]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split( i_o, o_p, test_size=0.4,
random_state=13)
```

In [1438]:
```
X_train.shape
```

Out[1438]:
```
(209, 22)
```

logistic regression

In [1439]:
```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_
score, classification_report, confusion_matrix
from sklearn.metrics import plot_confusion_matrix
```

In [1440]:
```
classifier = LogisticRegression()
classifier.fit(X_train,y_train)
```

```
C:\Users\ashiq\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
938: 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-regres
sion
  n_iter_i = _check_optimize_result(
```

Out[1440]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [1441]:
```
y_pred = classifier.predict(X_test)
```

In [1442]:
```
np.mean(y_pred==y_test.values)
```

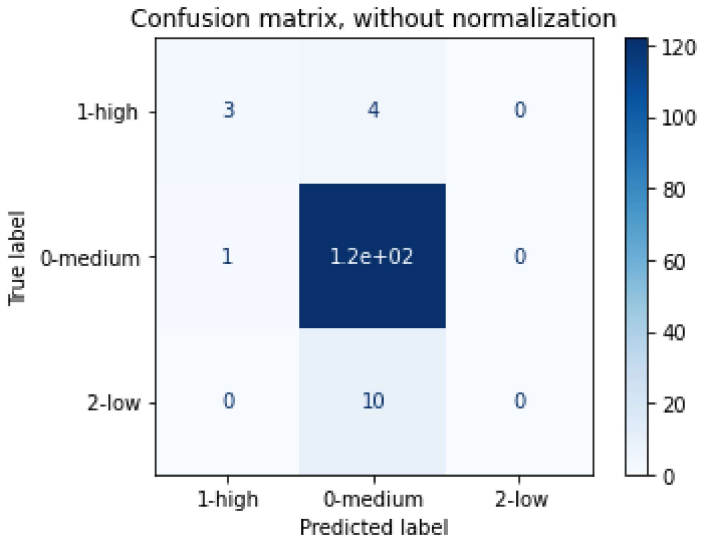
Out[1442]:
```
0.8928571428571429
```

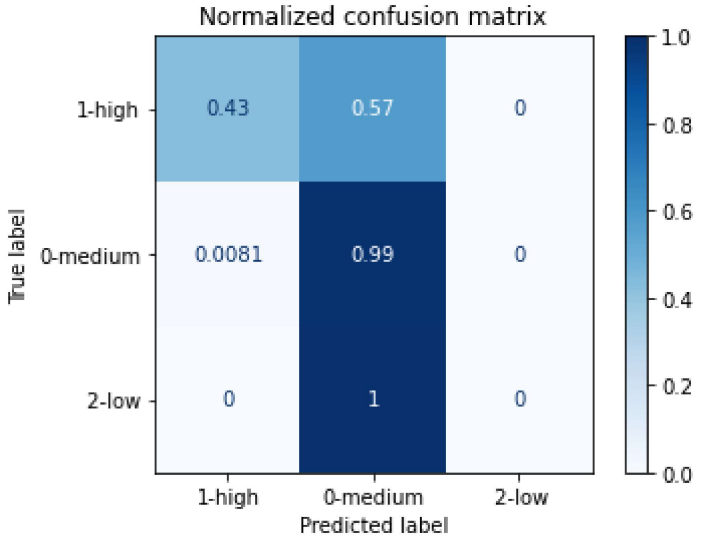
In [1443]:
```
titles_options = [("Confusion matrix, without normalization", None),
                  ("Normalized confusion matrix", 'true')]
class_names =['1-high', '0-medium',' 2-low']
for title, normalize in titles_options:
    disp = plot_confusion_matrix(classifier, X_test, y_test,
                                 display_labels=class_names,
                                 cmap=plt.cm.Blues,
                                 normalize=normalize)
    disp.ax_.set_title(title)

    print(title)
    print(disp.confusion_matrix)

plt.show()
```

```
Confusion matrix, without normalization
[[  3   4   0]
 [  1 122   0]
 [  0  10   0]]
Normalized confusion matrix
[[0.429 0.571 0.   ]
 [0.008 0.992 0.   ]
 [0.    1.    0.   ]]
```





In [1444]:
```
print(f1_score(y_test, y_pred, average="weighted"))
print(precision_score(y_test, y_pred, average="weighted"))
print(recall_score(y_test, y_pred, average="weighted"))
```

```
0.8549616406759264
0.8256302521008404
0.8928571428571429

C:\Users\ashiq\anaconda3\lib\site-packages\sklearn\metrics\_classification.p
y:1272: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0
in labels with no predicted samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [1445]:
```
from sklearn.naive_bayes import MultinomialNB as MB
from sklearn.naive_bayes import GaussianNB as GB
import numpy as np
```

In [1446]:
```
# Multinomial Naive Bayes
classifier_mb = MB()
classifier_mb.fit(X_train,y_train)
```

Out[1446]:
```
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

In [1447]:
```
y_pred_na = classifier_mb.predict(X_test)
accuracy_naive = np.mean(y_pred_na==y_test.values)
accuracy_naive
```

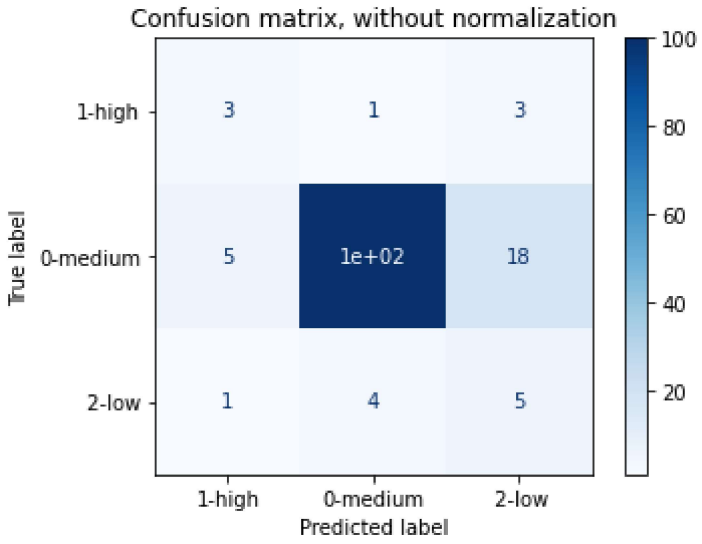
Out[1447]:
```
0.7714285714285715
```

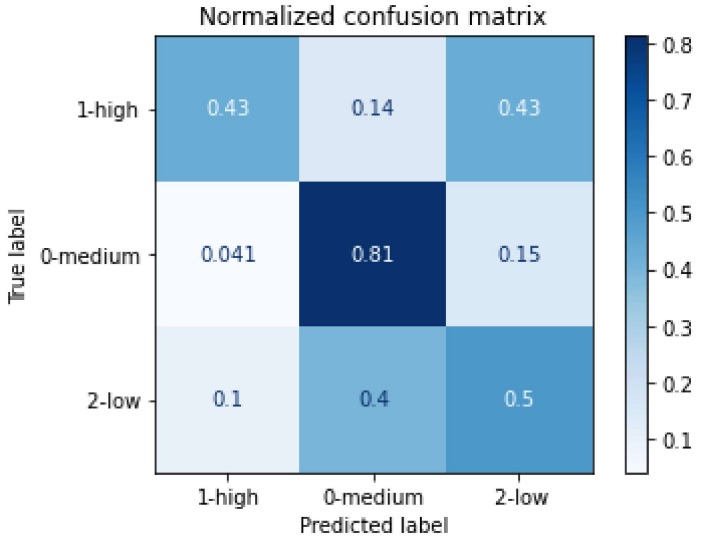
In [1448]:
```
titles_options = [("Confusion matrix, without normalization", None),
                  ("Normalized confusion matrix", 'true')]
class_names =['1-high', '0-medium',' 2-low']
for title, normalize in titles_options:
    disp = plot_confusion_matrix(classifier_mb, X_test, y_test,
                                 display_labels=class_names,
                                 cmap=plt.cm.Blues,
                                 normalize=normalize)
    disp.ax_.set_title(title)

    print(title)
    print(disp.confusion_matrix)

plt.show()
```

```
Confusion matrix, without normalization
[[  3   1   3]
 [  5 100  18]
 [  1   4   5]]
Normalized confusion matrix
[[0.429 0.143 0.429]
 [0.041 0.813 0.146]
 [0.1   0.4   0.5  ]]
```





In [1449]:
```
print(f1_score(y_test, y_pred_na, average="weighted"))
print(precision_score(y_test, y_pred_na, average="weighted"))
print(recall_score(y_test, y_pred_na, average="weighted"))
```

```
0.8092679615705931
0.8671376242804814
0.7714285714285715
```

adaboost classifier

In [1450]:
```
from sklearn.ensemble import AdaBoostClassifier
# Create adaboost classifer object
clf =AdaBoostClassifier(n_estimators=50,random_state=0)
# Train Adaboost Classifer
model_ada = clf.fit(X_train, y_train)
#Predict the response for test dataset
y_pred_adaboost = model_ada.predict(X_test)
y_pred_adaboost
```

Out[1450]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1], dtype=int64)
```

In [1451]:
```
print("Accuracy_ada boost:",metrics.accuracy_score(y_test, y_pred_adaboost))
```

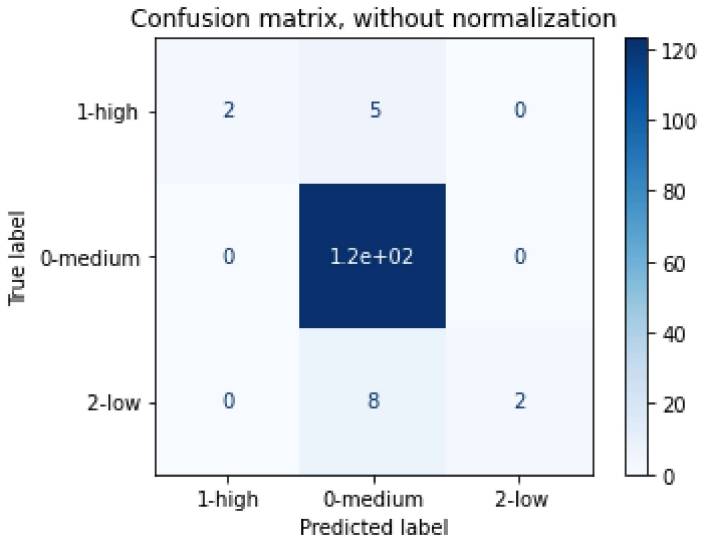
```
Accuracy_ada boost: 0.9071428571428571
```

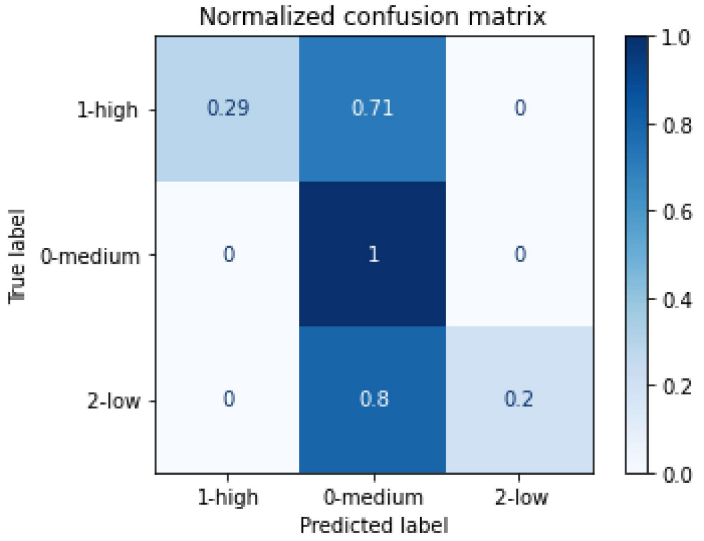
In [1452]:
```
titles_options = [("Confusion matrix, without normalization", None),
                  ("Normalized confusion matrix", 'true')]
class_names =['1-high', '0-medium',' 2-low']
for title, normalize in titles_options:
    disp = plot_confusion_matrix(model_ada, X_test, y_test,
                                 display_labels=class_names,
                                 cmap=plt.cm.Blues,
                                 normalize=normalize)
    disp.ax_.set_title(title)

    print(title)
    print(disp.confusion_matrix)

plt.show()
```

```
Confusion matrix, without normalization
[[  2   5   0]
 [  0 123   0]
 [  0   8   2]]
Normalized confusion matrix
[[0.286 0.714 0.   ]
 [0.    1.    0.   ]
 [0.    0.8   0.2  ]]
```





In [1453]:
```
print(f1_score(y_test, y_pred_adaboost, average="weighted"))
print(precision_score(y_test, y_pred_adaboost, average="weighted"))
print(recall_score(y_test, y_pred_adaboost, average="weighted"))
```

```
0.8805049947907092
0.9160189075630253
0.9071428571428571
```

gradiant boosting

In [1455]:
```
from sklearn.ensemble import GradientBoostingClassifier
```

In [1456]:
```
model_gradiant = GradientBoostingClassifier()
model_gradiant.fit(X_train, y_train)
y_pred_gradiant = model_gradiant.predict(X_test)
y_pred_gradiant
```

Out[1456]:
```
array([1, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 0, 1,
       1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 2, 1, 2, 1, 1,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1], dtype=int64)
```

In [1457]:
```
accuracy_gradiant = np.mean(y_pred_gradiant==y_test.values)
accuracy_gradiant
```

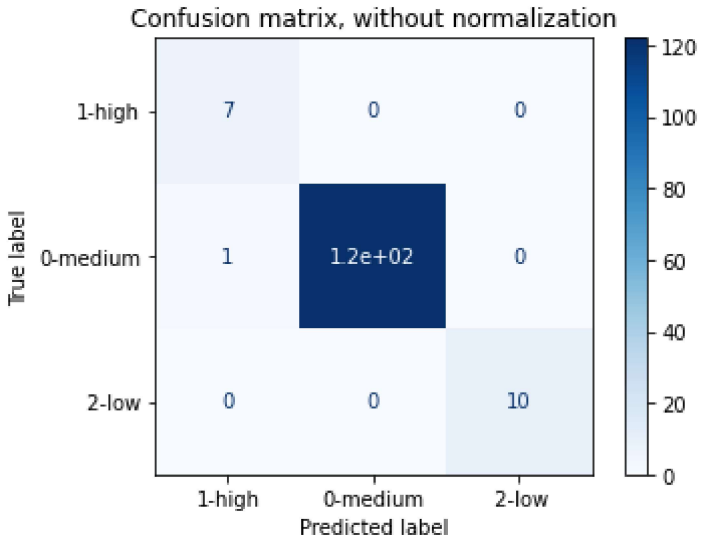
Out[1457]:
```
0.9928571428571429
```

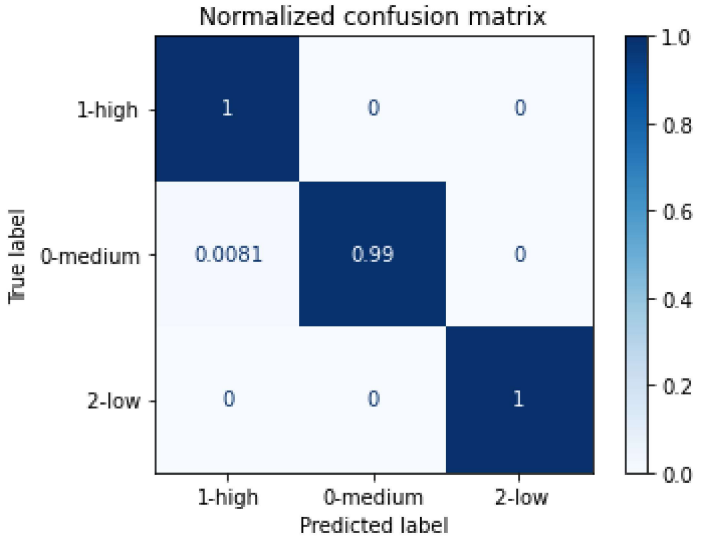
In [1458]:
```
titles_options = [("Confusion matrix, without normalization", None),
                  ("Normalized confusion matrix", 'true')]
class_names =['1-high', '0-medium',' 2-low']
for title, normalize in titles_options:
    disp = plot_confusion_matrix(model_gradiant, X_test, y_test,
                                 display_labels=class_names,
                                 cmap=plt.cm.Blues,
                                 normalize=normalize)
    disp.ax_.set_title(title)

    print(title)
    print(disp.confusion_matrix)

plt.show()
```

```
Confusion matrix, without normalization
[[  7   0   0]
 [  1 122   0]
 [  0   0  10]]
Normalized confusion matrix
[[1.    0.    0.   ]
 [0.008 0.992 0.   ]
 [0.    0.    1.   ]]
```





In [1459]:
```
print(f1_score(y_test, y_pred_gradiant, average="weighted"))
print(precision_score(y_test, y_pred_gradiant, average="weighted"))
print(recall_score(y_test, y_pred_gradiant, average="weighted"))
```

```
0.993080660835763
0.99375
0.9928571428571429
```

In [ ]:
```

```