In [1]:

```
import pandas as pd
import numpy as np
import  matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:

```
vaccinetrain=pd.read_csv("training_set_features.csv")
```

In [3]:

```
vaccinetest=pd.read_csv("test_set_features.csv")
```

In [4]:

```
vaccine_labels=pd.read_csv("training_set_labels.csv")
```

In [5]:

```
print(vaccinetrain.shape)
print(vaccinetest.shape)
print(vaccine_labels.shape)
```

```
(26707, 36)
(26708, 36)
(26707, 3)
```

In [6]:

```
vaccinetrain
```

Out[6]:

	respondent_id	h1n1_concern	h1n1_knowledge	behavioral_antiviral_meds	behavioral_avoida
0	0	1.0	0.0	0.0	
1	1	3.0	2.0	0.0	
2	2	1.0	1.0	0.0	
3	3	1.0	1.0	0.0	
4	4	2.0	1.0	0.0	
...	...	...	...	...	
26702	26702	2.0	0.0	0.0	
26703	26703	1.0	2.0	0.0	
26704	26704	2.0	2.0	0.0	
26705	26705	1.0	1.0	0.0	
26706	26706	0.0	0.0	0.0	

26707 rows × 36 columns

In [7]:

```
vaccinetest
```

Out[7]:

	respondent_id	h1n1_concern	h1n1_knowledge	behavioral_antiviral_meds	behavioral_avoida
0	26707	2.0	2.0	0.0	
1	26708	1.0	1.0	0.0	
2	26709	2.0	2.0	0.0	
3	26710	1.0	1.0	0.0	
4	26711	3.0	1.0	1.0	
...	...	...	...	...	
26703	53410	1.0	1.0	0.0	
26704	53411	3.0	1.0	0.0	
26705	53412	0.0	1.0	0.0	
26706	53413	3.0	1.0	0.0	
26707	53414	2.0	1.0	0.0	

26708 rows × 36 columns

In [8]:

```
vaccinetrain['id']='train'
vaccinetest['id']='test'
```

In [9]:

```
combinedf=pd.concat([vaccinetrain,vaccinetest],axis=0)
```

In [10]:

```
combinedf.shape
```

Out[10]:

```
(53415, 37)
```

In [11]:

```
combinedf.isnull().sum().sort_values(ascending=False)
```

Out[11]:

```
employment_occupation          26896
employment_industry            26605
health_insurance               24502
income_poverty                  8920
doctor_recc_h1n1                4320
doctor_recc_seasonal            4320
rent_or_own                     4078
employment_status               2934
marital_status                  2850
education                       2814
chronic_med_condition           1903
child_under_6_months            1633
health_worker                   1593
opinion_seas_sick_from_vacc     1058
opinion_seas_risk               1013
opinion_seas_vacc_effective      914
opinion_h1n1_vacc_effective      789
opinion_h1n1_sick_from_vacc      770
opinion_h1n1_risk                768
household_children               474
household_adults                 474
behavioral_avoidance             421
behavioral_touch_face            256
h1n1_knowledge                   238
h1n1_concern                     177
behavioral_outside_home          164
behavioral_large_gatherings      159
behavioral_antiviral_meds        150
behavioral_wash_hands             82
behavioral_face_mask              38
census_msa                         0
respondent_id                      0
hhs_geo_region                     0
sex                                0
race                               0
age_group                          0
id                                 0
dtype: int64
```

In [12]:
```
for col in combinedf.drop(['respondent_id','id'],axis=1).columns:
    freq=combinedf[col].value_counts(dropna=False)
    print(freq)
```

```
2.0    21318
1.0    16311
3.0     9114
0.0     6495
NaN      177
Name: h1n1_concern, dtype: int64
1.0    29227
2.0    19002
0.0     4948
NaN      238
Name: h1n1_knowledge, dtype: int64
0.0    50642
1.0     2623
NaN      150
Name: behavioral_antiviral_meds, dtype: int64
1.0    38564
0.0    14430
NaN      421
Name: behavioral_avoidance, dtype: int64
```

In [13]:
```
#non-parametric algorithms means algorithms which has no prior assumptions
#ex : KNN (K-nearest neighbours)
#missing values imputation (KNNimputer)
#label encoding must be done before imputation
#label encoder will keep the missing values as it is
```

In [14]:
```
from sklearn.preprocessing import LabelEncoder
```

In [15]:
```
stringcols=combinedf.drop(['respondent_id','id'],axis=1)
```

In [16]:
```
stringcols=stringcols.astype(str).apply(lambda series:pd.Series(
LabelEncoder().fit_transform(series[series.notnull()]),
index=series[series.notnull()].index))
```

In [17]:
```
from sklearn.impute import KNNImputer
```

In [18]:
```
imputer=KNNImputer()
```

In [19]:
```
stringcolsimp=imputer.fit_transform(stringcols)
```

In [20]:
```
stringcols.isnull().sum().sort_values(ascending=False)
```

Out[20]:
```
h1n1_concern                   0
marital_status                 0
opinion_seas_sick_from_vacc    0
age_group                      0
education                      0
race                           0
sex                            0
income_poverty                 0
rent_or_own                    0
opinion_seas_vacc_effective    0
employment_status              0
hhs_geo_region                 0
census_msa                     0
household_adults               0
household_children             0
employment_industry            0
opinion_seas_risk              0
opinion_h1n1_sick_from_vacc    0
h1n1_knowledge                 0
behavioral_touch_face          0
behavioral_antiviral_meds      0
behavioral_avoidance           0
behavioral_face_mask           0
behavioral_wash_hands          0
behavioral_large_gatherings    0
behavioral_outside_home        0
doctor_recc_h1n1               0
opinion_h1n1_risk              0
doctor_recc_seasonal           0
chronic_med_condition          0
child_under_6_months           0
health_worker                  0
health_insurance               0
opinion_h1n1_vacc_effective    0
employment_occupation          0
dtype: int64
```

In [21]:
```
stringcolsimp=pd.DataFrame(stringcolsimp,columns=stringcols.columns)
```

In [22]:
```
stringcolsimp
```

Out[22]:

	h1n1_concern	h1n1_knowledge	behavioral_antiviral_meds	behavioral_avoidance	behavioral_
0	1.0	0.0	0.0	0.0	
1	3.0	2.0	0.0	1.0	
2	1.0	1.0	0.0	1.0	
3	1.0	1.0	0.0	1.0	
4	2.0	1.0	0.0	1.0	
...	...	...	...	...	
53410	1.0	1.0	0.0	1.0	
53411	3.0	1.0	0.0	1.0	
53412	0.0	1.0	0.0	0.0	
53413	3.0	1.0	0.0	1.0	
53414	2.0	1.0	0.0	0.0	

53415 rows × 35 columns

In [23]:
```
for col in stringcolsimp.columns:
  freq=stringcolsimp[col].value_counts(dropna=False)
  print(freq)
```

```
2.0    21318
1.0    16311
3.0     9114
0.0     6495
4.0      177
Name: h1n1_concern, dtype: int64
1.0    29227
2.0    19002
0.0     4948
3.0      238
Name: h1n1_knowledge, dtype: int64
0.0    50642
1.0     2623
2.0      150
Name: behavioral_antiviral_meds, dtype: int64
1.0    38564
0.0    14430
2.0      421
Name: behavioral_avoidance, dtype: int64
0.0    40687
```

In [24]:
```
vaccinetraindf=stringcolsimp.loc[0:26706]
```

In [25]:
```
vaccinetestdf=stringcolsimp.loc[26707:53415]
```

In [26]:

```
print(vaccinetraindf.shape)
print(vaccinetestdf.shape)
```

```
(26707, 35)
(26708, 35)
```

LogisticRegression

In [27]:

```
from sklearn.linear_model import LogisticRegression
```

In [28]:

```
logreg=LogisticRegression()
```

In [29]:

```
X=vaccinetraindf
y=vaccine_labels.h1n1_vaccine
```

In [30]:

```
logregmodel=logreg.fit(X,y)
```

```
C:\Users\admin\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:76
3: 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 (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-regressi
on (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on)
  n_iter_i = _check_optimize_result(
```

In [31]:

```
logregmodel.score(X,y)
```

Out[31]:

```
0.8280975025274273
```

In [32]:

```
logregpredict=logregmodel.predict(X)
```

In [33]:

```
from sklearn.metrics import classification_report
```

```
In [34]: print(classification_report(y,logregpredict))
```

```
              precision    recall  f1-score   support

           0       0.85      0.95      0.90     21033
           1       0.67      0.37      0.48      5674

    accuracy                           0.83     26707
   macro avg       0.76      0.66      0.69     26707
weighted avg       0.81      0.83      0.81     26707
```

```
In [35]: y.value_counts()
```

```
Out[35]: 0    21033
         1     5674
         Name: h1n1_vaccine, dtype: int64
```

CategoricalNaiveBayes

```
In [36]: from sklearn.naive_bayes import CategoricalNB
```

```
In [37]: catnb=CategoricalNB()
```

```
In [38]: catnb=CategoricalNB()
```

```
In [39]: catNBmodel.score(X,y)
```

```
Out[39]: 0.8092260456060208
```

```
In [40]: catNBpredict=catNBmodel.predict(X)
```

```
In [41]: print(classification_report(y,catNBpredict))
```

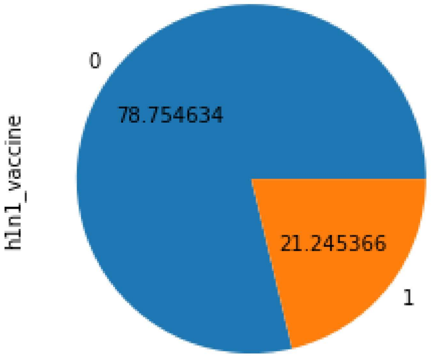
```
              precision    recall  f1-score   support

           0       0.89      0.86      0.88     21033
           1       0.55      0.61      0.58      5674

    accuracy                           0.81     26707
   macro avg       0.72      0.74      0.73     26707
weighted avg       0.82      0.81      0.81     26707
```

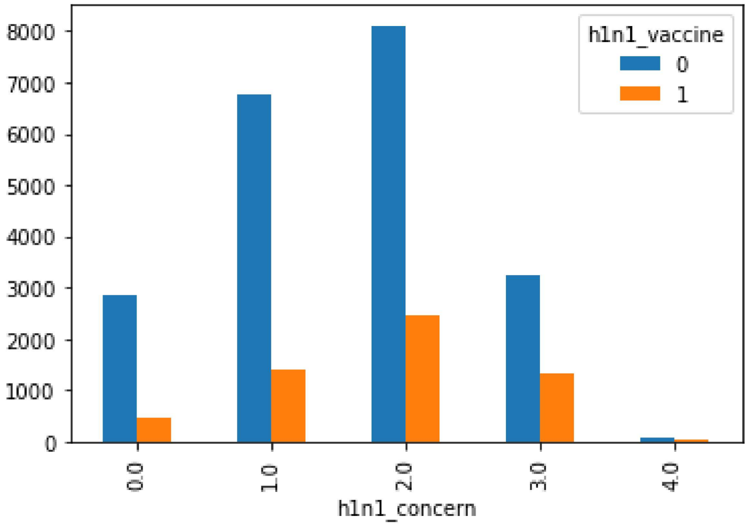
In [42]:
```
pd.value_counts(y).plot(kind="pie",autopct="%f")
```

Out[42]:
```
<AxesSubplot:ylabel='h1n1_vaccine'>
```



In [43]:
```
pd.crosstab(X.h1n1_concern,y).plot(kind='bar',stacked=False)
```

Out[43]:
```
<AxesSubplot:xlabel='h1n1_concern'>
```



In [44]:
```
combinedf.columns
```

Out[44]:
```
Index(['respondent_id', 'h1n1_concern', 'h1n1_knowledge',
       'behavioral_antiviral_meds', 'behavioral_avoidance',
       'behavioral_face_mask', 'behavioral_wash_hands',
       'behavioral_large_gatherings', 'behavioral_outside_home',
       'behavioral_touch_face', 'doctor_recc_h1n1', 'doctor_recc_seasonal',
       'chronic_med_condition', 'child_under_6_months', 'health_worker',
       'health_insurance', 'opinion_h1n1_vacc_effective', 'opinion_h1n1_risk',
       'opinion_h1n1_sick_from_vacc', 'opinion_seas_vacc_effective',
       'opinion_seas_risk', 'opinion_seas_sick_from_vacc', 'age_group',
       'education', 'race', 'sex', 'income_poverty', 'marital_status',
       'rent_or_own', 'employment_status', 'hhs_geo_region', 'census_msa',
       'household_adults', 'household_children', 'employment_industry',
       'employment_occupation', 'id'],
      dtype='object')
```

In [45]:
```
combinedf.head()
```

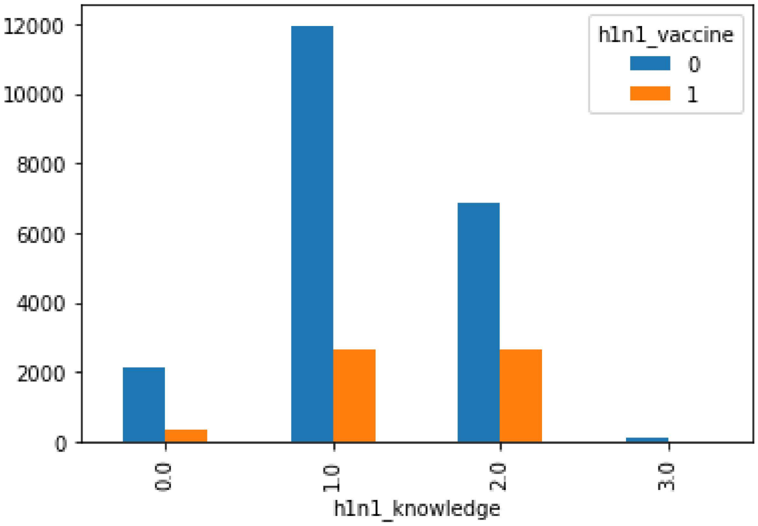
Out[45]:

	respondent_id	h1n1_concern	h1n1_knowledge	behavioral_antiviral_meds	behavioral_avoidance
0	0	1.0	0.0	0.0	0.0
1	1	3.0	2.0	0.0	1.0
2	2	1.0	1.0	0.0	1.0
3	3	1.0	1.0	0.0	1.0
4	4	2.0	1.0	0.0	1.0

5 rows × 37 columns

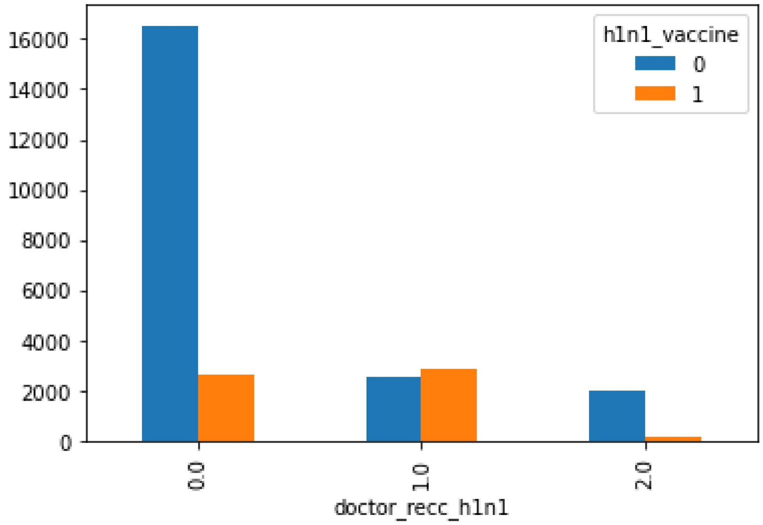
In [46]:
```
pd.crosstab(X.h1n1_knowledge,y).plot(kind='bar',stacked=False)
```

Out[46]:
```
<AxesSubplot:xlabel='h1n1_knowledge'>
```



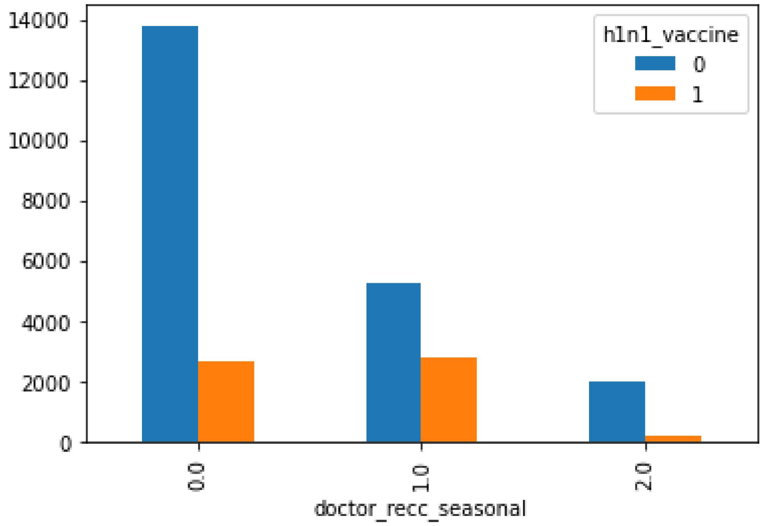
In [47]:
```
pd.crosstab(X.doctor_recc_h1n1,y).plot(kind='bar',stacked=False)
```

Out[47]:
```
<AxesSubplot:xlabel='doctor_recc_h1n1'>
```



In [48]:
```
pd.crosstab(X.doctor_recc_seasonal,y).plot(kind='bar',stacked=False)
```

Out[48]:
```
<AxesSubplot:xlabel='doctor_recc_seasonal'>
```



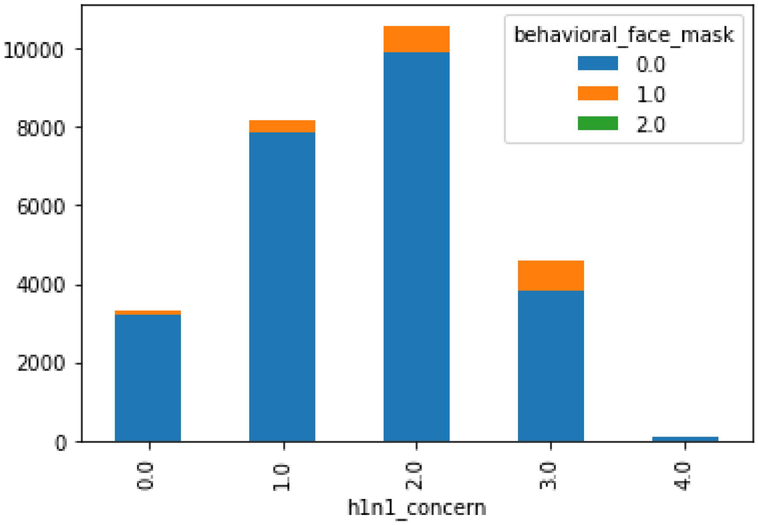
In [49]:
```
pd.crosstab(X.h1n1_concern,y)
```

Out[49]:

h1n1_vaccine	0	1
h1n1_concern		
0.0	2849	447
1.0	6756	1397
2.0	8102	2473
3.0	3250	1341
4.0	76	16

```
In [50]: pd.crosstab(X.h1n1_concern,X.behavioral_face_mask).plot(kind='bar',stacked=True)

Out[50]: <AxesSubplot:xlabel='h1n1_concern'>
```



MultiLevelPerceptron

```
In [51]: from sklearn.neural_network import MLPClassifier   #Multi-Level Perceptron()
```

```
In [54]: nn=MLPClassifier(hidden_layer_sizes=(50,100,50),activation='logistic',max_iter=1
```

```
In [55]: nnmodel=nn.fit(X,y)
```

```
In [56]: nnmodel.score(X,y) #h1n1 vaccine

Out[56]: 0.8757629086007414
```