

Logistic Regression-1 & 2

Data: [h1n1_vaccine_prediction.csv](#)

Libraries

```
# Jesus is my Saviour!
import os
os.chdir('C:\\Users\\Dr Vinod\\Desktop\\WD_python')
import pandas as pd
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from statsmodels.formula.api import ols
from scipy.stats import chi2_contingency
from sklearn.preprocessing import LabelEncoder

df = pd.read_csv('h1n1_vaccine_prediction.csv') #26707; 34 , 1st is unique id
df.info()
```

```
df = pd.read_csv('h1n1_vaccine_prediction.csv') #26707; 34 , 1st is unique id
df.info()
```

```
In [3]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 26707 entries, 0 to 26706
```

```
Data columns (total 34 columns):
```

#	Column	Non-Null Count	Dtype
0	unique_id	26707 non-null	int64
1	h1n1_worry	26615 non-null	float64
2	h1n1_awareness	26591 non-null	float64
3	antiviral_medication	26636 non-null	float64
4	contact_avoidance	26499 non-null	float64
5	bought_face_mask	26688 non-null	float64
6	wash_hands_frequently	26665 non-null	float64
7	avoid_large_gatherings	26620 non-null	float64
8	reduced_outside_home_cont	26625 non-null	float64
9	avoid_touch_face	26579 non-null	float64
10	dr_recc_h1n1_vacc	24547 non-null	float64
11	dr_recc_seasonal_vacc	24547 non-null	float64
12	chronic_medication	25736 non-null	float64
13	cont_child_undr_6_mnth	25887 non-null	float64
14	is_health_worker	25903 non-null	float64

15	has_health_insur	14433 non-null	float64
16	is_h1n1_vacc_effective	26316 non-null	float64
17	is_h1n1_risky	26319 non-null	float64
18	sick_from_h1n1_vacc	26312 non-null	float64
19	is_seas_vacc_effective	26245 non-null	float64
20	is_seas_risky	26193 non-null	float64
21	sick_from_seas_vacc	26170 non-null	float64
22	age_bracket	26707 non-null	object
23	qualification	25300 non-null	object
24	race	26707 non-null	object
25	sex	26707 non-null	object
26	income_level	22284 non-null	object
27	marital_status	25299 non-null	object
28	housing_status	24665 non-null	object
29	employment	25244 non-null	object
30	census_msa	26707 non-null	object
31	no_of_adults	26458 non-null	float64
32	no_of_children	26458 non-null	float64
33	h1n1_vaccine	26707 non-null	int64

```
dtypes: float64(23), int64(2), object(9)
```

```
memory usage: 6.9+ MB
```

Drop

```
# Lets drop 1st unique_id
df33 = df.drop(['unique_id'], axis = 1)
df33.info() # now 33 columns
# dropping all will give 11794 rows only 50% values
dfnomissing = df33.dropna()
dfnomissing.info() # # 11,794 rows; 33 columns
# its not a good idea to carry with 11,794 rows
## note that index identification remains same as in
# the original file
'''
so, we will remove the column- "has_health_insur" (14433 rows only)
and create a new file name 'df32'. This is shown below.
'''
df32 = df33.drop(['has_health_insur'], axis = 1)
```

```
32 df32.info() # observe many missing values in many columns, now 32 columns
```

```
In [5]: df32.info() # observe many missing values in many columns, now 32 columns
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 26707 entries, 0 to 26706
```

```
Data columns (total 32 columns):
```

#	Column	Non-Null Count	Dtype
0	h1n1_worry	26615 non-null	float64
1	h1n1_awareness	26591 non-null	float64
2	antiviral_medication	26636 non-null	float64
3	contact_avoidance	26499 non-null	float64
4	bought_face_mask	26688 non-null	float64
5	wash_hands_frequently	26665 non-null	float64
6	avoid_large_gatherings	26620 non-null	float64
7	reduced_outside_home_cont	26625 non-null	float64
8	avoid_touch_face	26579 non-null	float64
9	dr_recc_h1n1_vacc	24547 non-null	float64
10	dr_recc_seasonal_vacc	24547 non-null	float64
11	chronic_medical_condition	25736 non-null	float64
12	cont_child_undr_6_mnth	25887 non-null	float64
13	is_health_worker	25903 non-null	float64
14	is_h1n1_vacc_effective	26316 non-null	float64
15	is_h1n1_risky	26319 non-null	float64
16	sick_from_h1n1_vacc	26312 non-null	float64
17	is_seas_vacc_effective	26245 non-null	float64
18	is_seas_risky	26193 non-null	float64
19	sick_from_seas_vacc	26170 non-null	float64
20	age_bracket	26707 non-null	object
21	qualification	25300 non-null	object
22	race	26707 non-null	object
23	sex	26707 non-null	object
24	income_level	22284 non-null	object
25	marital_status	25299 non-null	object
26	housing_status	24665 non-null	object
27	employment	25244 non-null	object
28	census_msa	26707 non-null	object
29	no_of_adults	26458 non-null	float64
30	no_of_children	26458 non-null	float64
31	h1n1_vaccine	26707 non-null	int64

dtypes: float64(22), int64(1), object(9)
memory usage: 6.5+ MB

```
# Lets remove all missing values from df32
df_vac = df32.dropna() # 19642 , 32
df_vac.info()
## now , 19642 is a good no to go with!
```

```
In [6]: df_vac = df32.dropna() # 19642 , 32
```

```
In [7]: df_vac.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
Int64Index: 19642 entries, 0 to 26706
```

```
Data columns (total 32 columns):
```

#	Column	Non-Null Count	Dtype
0	h1n1_worry	19642 non-null	float64
1	h1n1_awareness	19642 non-null	float64
2	antiviral_medication	19642 non-null	float64
3	contact_avoidance	19642 non-null	float64
4	bought_face_mask	19642 non-null	float64
5	wash_hands_frequently	19642 non-null	float64
6	avoid_large_gatherings	19642 non-null	float64
7	reduced_outside_home_cont	19642 non-null	float64
8	avoid_touch_face	19642 non-null	float64
9	dr_recc_h1n1_vacc	19642 non-null	float64
10	dr_recc_seasonal_vacc	19642 non-null	float64
11	chronic_medical_condition	19642 non-null	float64
12	cont_child_undr_6_mnths	19642 non-null	float64
13	is_health_worker	19642 non-null	float64
14	is_h1n1_vacc_effective	19642 non-null	float64
15	is_h1n1_risky	19642 non-null	float64

16	sick_from_h1n1_vacc	19642 non-null	float64
17	is_seas_vacc_effective	19642 non-null	float64
18	is_seas_risky	19642 non-null	float64
19	sick_from_seas_vacc	19642 non-null	float64
20	age_bracket	19642 non-null	object
21	qualification	19642 non-null	object
22	race	19642 non-null	object
23	sex	19642 non-null	object
24	income_level	19642 non-null	object
25	marital_status	19642 non-null	object
26	housing_status	19642 non-null	object
27	employment	19642 non-null	object
28	census_msa	19642 non-null	object
29	no_of_adults	19642 non-null	float64
30	no_of_children	19642 non-null	float64
31	h1n1_vaccine	19642 non-null	int64

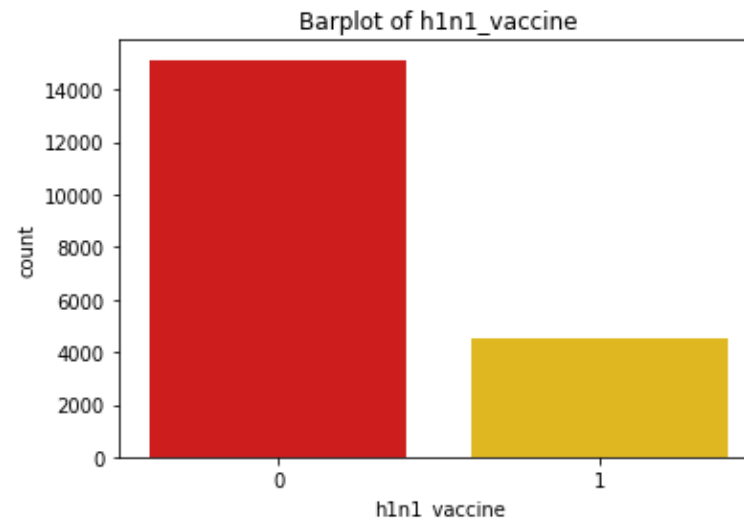
```
dtypes: float64(22), int64(1), object(9)
```

```
memory usage: 4.9+ MB
```

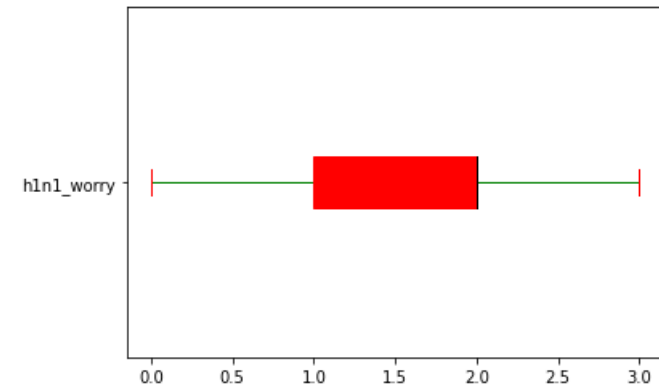
```
###_____1 ALWAYS START WITH TARGET VARIABLE
# 1 h1n1_vaccine - Target Variable
```

```
###_____1 ALWAYS START WITH TARGET VARIABLE
# 1 h1n1_vaccine - Target Variable
df_vac.h1n1_vaccine.isnull().sum() #No Missing values
df_vac.h1n1_vaccine.value_counts()
'''
0      15128
1       4514
Name: h1n1_vaccine, dtype: int64
'''

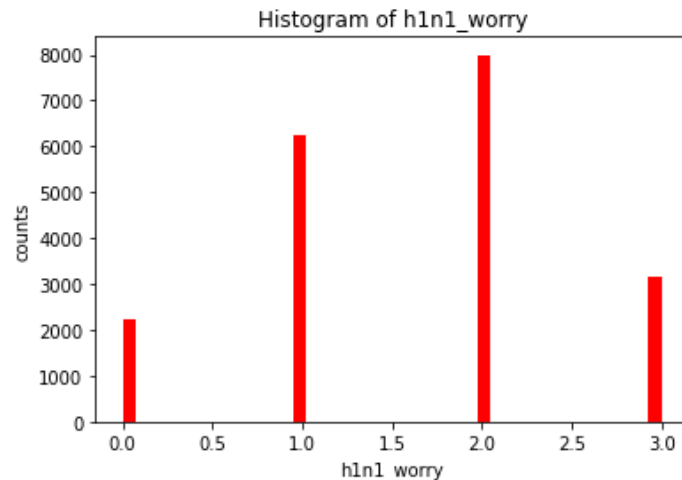
# Bar Plot
sns.countplot(x = 'h1n1_vaccine', data = df_vac , palette = 'hot')
plt.title('Barplot of h1n1_vaccine')
```




```
#_____ 2 h1n1_worry [0,1,2,3] ordered
#_____ histogram
#_run in block
plt.hist(df_vac.h1n1_worry, bins = 'auto', facecolor = 'red')
plt.xlabel('h1n1_worry')
plt.ylabel('counts')
plt.title('Histogram of h1n1_worry')
```



```
#_____ boxplot
props2 = dict(boxes = 'red', whiskers = 'green', medians = 'black', caps = 'red')
df_vac['h1n1_worry'].plot.box(color=props2, patch_artist = True, vert = False) #No outliers
```



```
In [13]: df_vac.h1n1_worry.isnull().sum() #0 Missing values
Out[13]: 0

In [14]: df_vac.h1n1_worry.value_counts()
Out[14]:
2.0    7989
1.0    6229
3.0    3175
0.0    2249
Name: h1n1_worry, dtype: int64
```

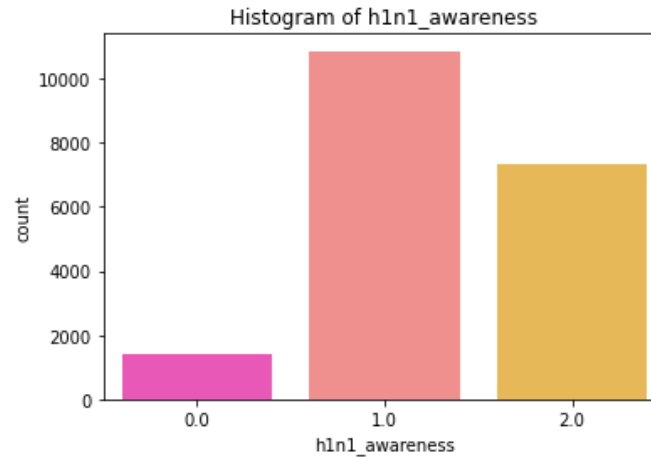
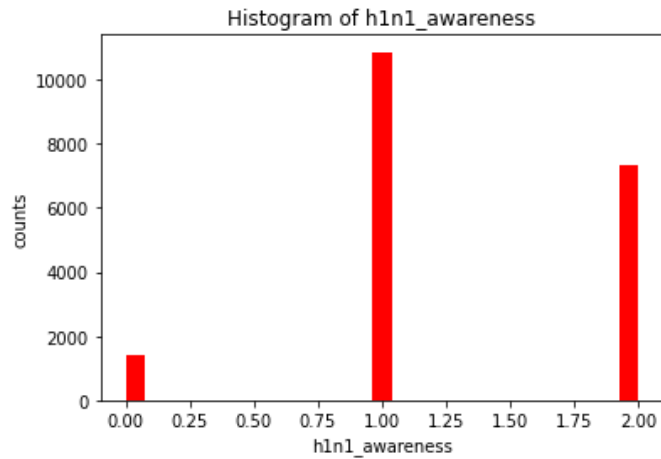

Is h1n1_worry a good predictor?

```
#Hypothesis Testing
import statsmodels.api as sm
from statsmodels.formula.api import ols
mod = ols('h1n1_worry ~ h1n1_vaccine', data = df_vac).fit()
aov_table = sm.stats.anova_lm(mod)
print(aov_table)
# 1.564e-79 ie p_value is <0.05; Ho Reject; Good Predictor
```

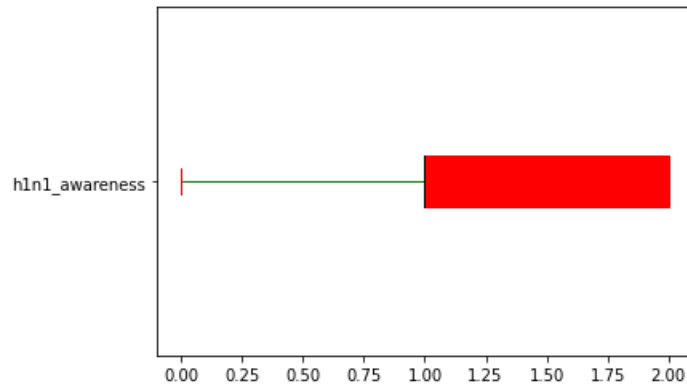
```
In [15]: import statsmodels.api as sm
...: from statsmodels.formula.api import ols
...: mod = ols('h1n1_worry ~ h1n1_vaccine', data = df_vac).fit()
...: aov_table = sm.stats.anova_lm(mod)
...: print(aov_table)
```

	df	sum_sq	mean_sq	F	PR(>F)
h1n1_vaccine	1.0	278.817902	278.817902	359.846071	1.564062e-79
Residual	19640.0	15217.572282	0.774825	NaN	NaN

#_____# 3 h1n1_awareness [0,1,2] ordered



```
In [23]: df_vac.h1n1_awareness.value_counts()
Out[23]:
1.0      10861
2.0       7362
0.0       1419
Name: h1n1_awareness, dtype: int64
```

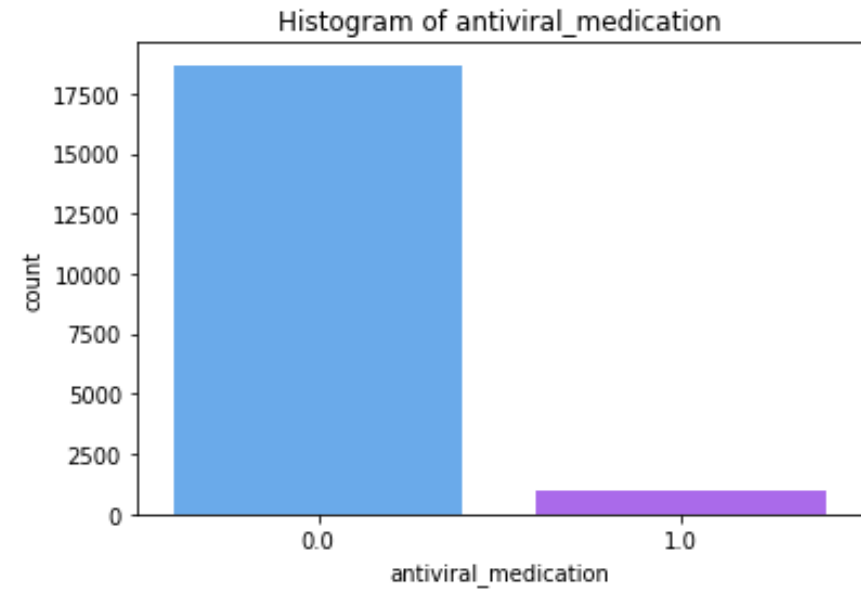


#Hypothesis Testing

```
import statsmodels.api as sm
from statsmodels.formula.api import ols
mod = ols('h1n1_awareness ~ h1n1_vaccine', data = df_vac).fit()
aov_table = sm.stats.anova_lm(mod)
print(aov_table)
#3.442e-70 ie p_value is <0.05; Ho Reject; Good Predictor
```

```
#_____ 4 antiviral_medication [0 and 1]
```

```
In [27]: df_vac.antiviral_medication.value_counts()  
Out[27]:  
0.0    18671  
1.0     971  
Name: antiviral_medication, dtype: int64
```



#Hypothesis Testing

```
from scipy.stats import chi2_contingency  
ct_antiviral = pd.crosstab(df_vac.h1n1_vaccine, df_vac.antiviral_medication)  
chi2_contingency(ct_antiviral, correction = False)  
# p_val = 3.9e-7, Ho reject, hence association exists, good predictor
```

```
#_____ 5 contact_avoidance [0 and 1]
```

```
df_vac.contact_avoidance.isnull().sum() #0 Missing values
```

```
df_vac.contact_avoidance.value_counts()
```

```
'''
```

```
1.0    14544
```

```
0.0     5098'''
```

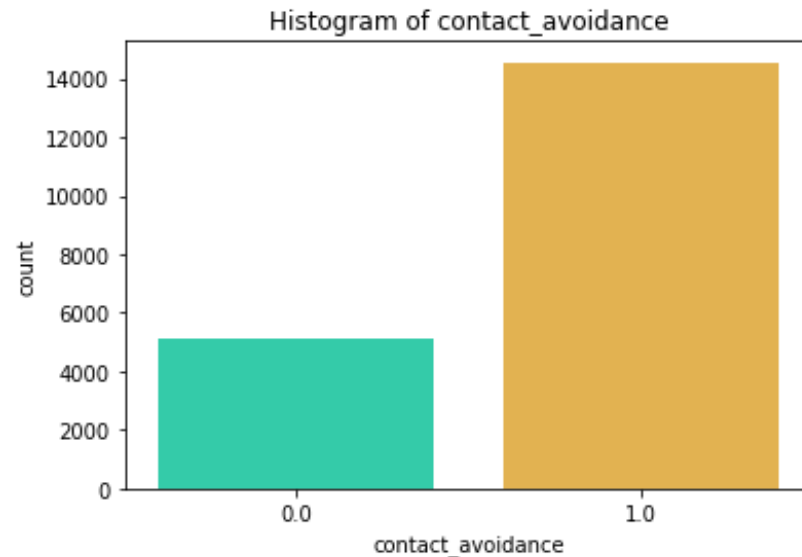
```
#Hypothesis Testing
```

```
from scipy.stats import chi2_contingency
```

```
ct_avoid = pd.crosstab(df_vac.h1n1_vaccine, df_vac.contact_avoidance)
```

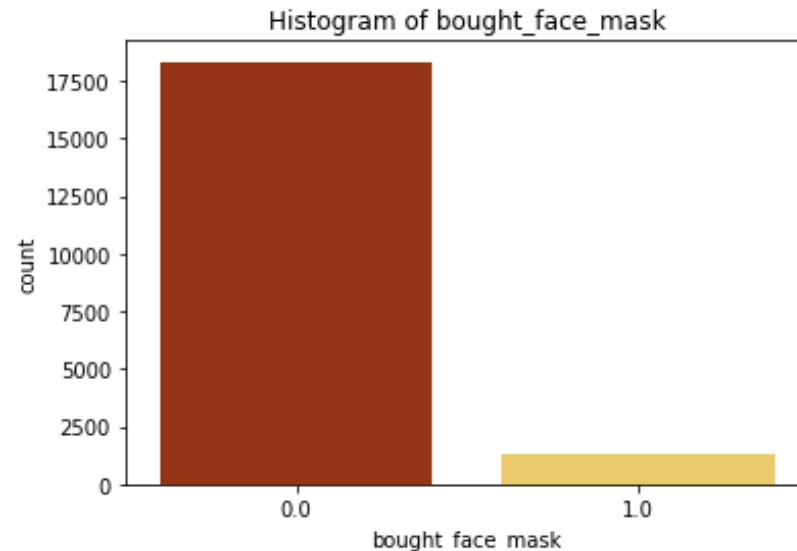
```
chi2_contingency(ct_avoid, correction = False)
```

```
# p_val = 6.6 e-10, Ho reject, hence association exists, good predictor
```



```
# _____ 6 bought_face_mask [0 and 1]

df_vac.bought_face_mask.isnull().sum() #0 Missing values
df_vac.bought_face_mask.value_counts()
'''
0.0    18312
1.0     1330'''
# Bar Plot
sns.countplot(x = 'bought_face_mask', data = df_vac , palette = 'afmhot')
plt.title('Histogram of bought_face_mask')
#Hypothesis Testing
ct_mask = pd.crosstab(df_vac.h1n1_vaccine, df_vac.bought_face_mask)
chi2_contingency(ct_mask, correction = False)
# p_val = 4.9 e-26, Ho reject, hence association exists, good predictor
```



```
#_____ 7 wash_hands_frequently [0 and 1]

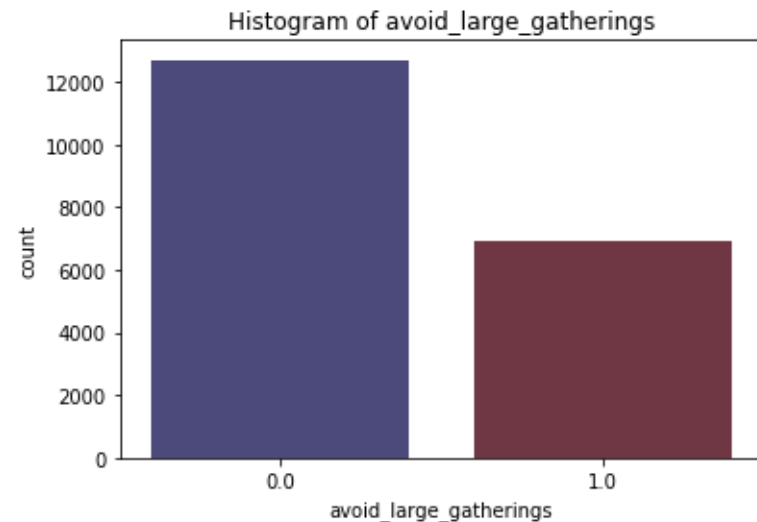
df_vac.wash_hands_frequently.isnull().sum() #0 Missing values
df_vac.wash_hands_frequently.value_counts()
'''
1.0      16399
0.0       3243'''
# Bar Plot
sns.countplot(x = 'wash_hands_frequently', data = df_vac , palette = 'rainbow')
plt.title('Histogram of wash_hands_frequently')

#Hypothesis Testing
ct_wash = pd.crosstab(df_vac.h1n1_vaccine, df_vac.wash_hands_frequently)
chi2_contingency(ct_wash, correction = False)
# p_val = 4.3 e-26, Ho reject, hence association exists, good predictor
```



```
#_____ 8 avoid_large_gatherings [0 and 1]

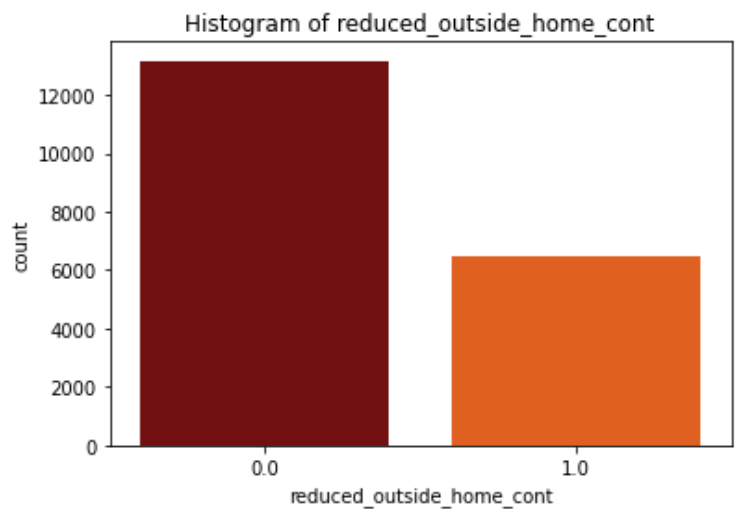
df_vac.avoid_large_gatherings.isnull().sum() #0 Missing values
df_vac.avoid_large_gatherings.value_counts()
'''
0.0    12703
1.0     6939'''
# Bar Plot
sns.countplot(x = 'avoid_large_gatherings', data = df_vac , palette = 'icefire')
plt.title('Histogram of avoid_large_gatherings')
#Hypothesis Testing
ct_gath = pd.crosstab(df_vac.h1n1_vaccine, df_vac.avoid_large_gatherings)
chi2_contingency(ct_gath, correction = False)
# p_val = 0.004, Ho reject, hence association exists, good predictor
```




```

#_____ 9 reduced_outside_home_cont [0 and 1]
df_vac.reduced_outside_home_cont.isnull().sum() #0 Missing values
df_vac.reduced_outside_home_cont.value_counts()
'''
0.0    13159
1.0     6483'''
# Bar Plot
sns.countplot(x = 'reduced_outside_home_cont', data = df_vac , palette = 'gist_heat')
plt.title('Histogram of reduced_outside_home_cont')
#Hypothesis Testing
ct_outside = pd.crosstab(df_vac.h1n1_vaccine, df_vac.reduced_outside_home_cont)
chi2_contingency(ct_outside, correction = False)
# p_val = 0.015, Ho reject, hence association exists, good predictor

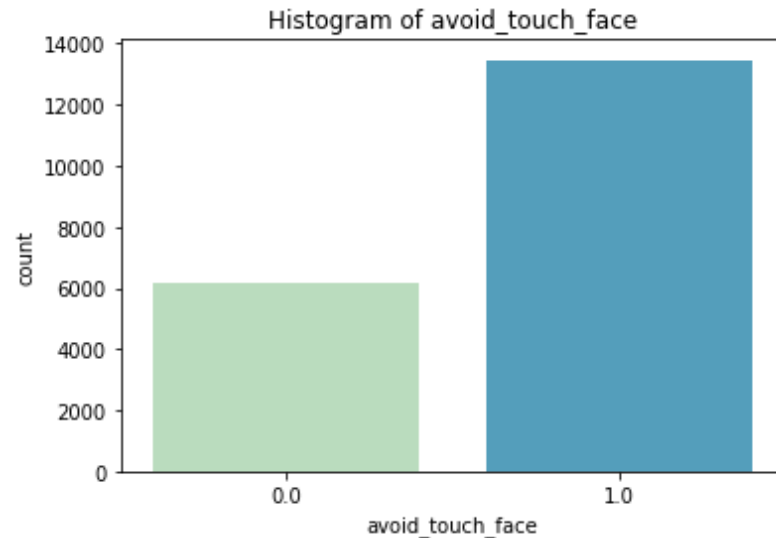
```



```

#_____10 avoid_touch_face [0 and 1]
df_vac.avoid_touch_face.isnull().sum() #0 Missing values
df_vac.avoid_touch_face.value_counts()
'''
1.0    13455
0.0     6187'''
# Bar Plot
sns.countplot(x = 'avoid_touch_face', data = df_vac , palette = 'GnBu')
plt.title('Histogram of avoid_touch_face')
#Hypothesis Testing
ct_face = pd.crosstab(df_vac.h1n1_vaccine, df_vac.avoid_touch_face)
chi2_contingency(ct_face, correction = False)
# p_val = 1.5e-23, Ho reject, hence association exists, good predictor

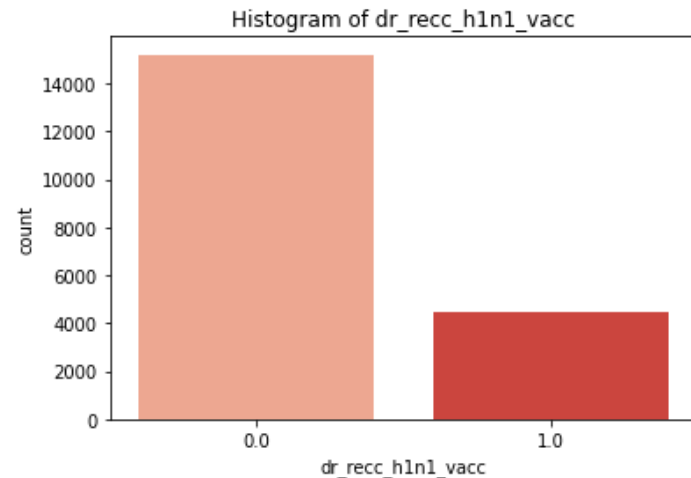
```



```

# 11 dr_recc_h1n1_vacc [0 and 1]
df_vac.dr_recc_h1n1_vacc.isnull().sum() #0 Missing values
df_vac.dr_recc_h1n1_vacc.value_counts()
'''
0.0    15203
1.0     4439'''
# Bar Plot
sns.countplot(x = 'dr_recc_h1n1_vacc', data = df_vac , palette = 'Reds')
plt.title('Histogram of dr_recc_h1n1_vacc')
#Hypothesis Testing
ct_drrec = pd.crosstab(df_vac.h1n1_vaccine, df_vac.dr_recc_h1n1_vacc)
chi2_contingency(ct_drrec, correction = False)
# p_val = 0, Ho reject, hence association exists, good predictor

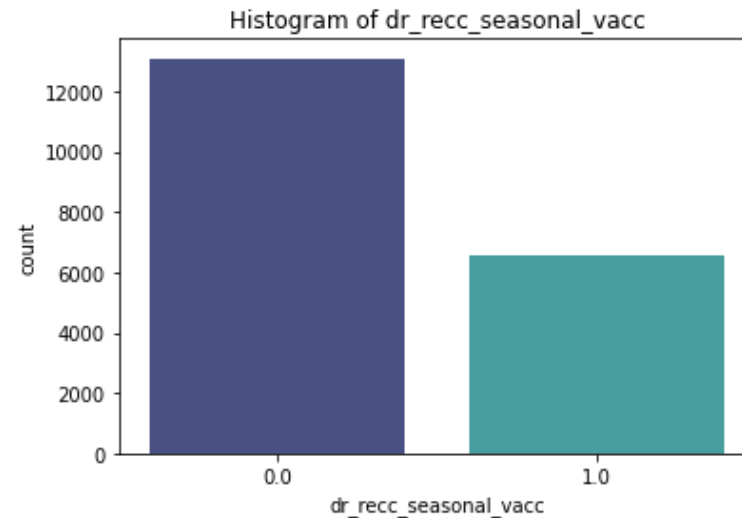
```



```

#_____ 12 dr_recc_seasonal_vacc [0 and 1]
df_vac.dr_recc_seasonal_vacc.isnull().sum() #0 Missing values
df_vac.dr_recc_seasonal_vacc.value_counts()
'''
0.0    13091
1.0     6551'''
# Bar Plot
sns.countplot(x = 'dr_recc_seasonal_vacc', data = df_vac , palette = 'mako')
plt.title('Histogram of dr_recc_seasonal_vacc')
#Hypothesis Testing
ct_drseason = pd.crosstab(df_vac.h1n1_vaccine, df_vac.dr_recc_seasonal_vacc)
chi2_contingency(ct_drseason, correction = False)
# p_val = 2.2e-192, Ho reject, hence association exists, good predictor

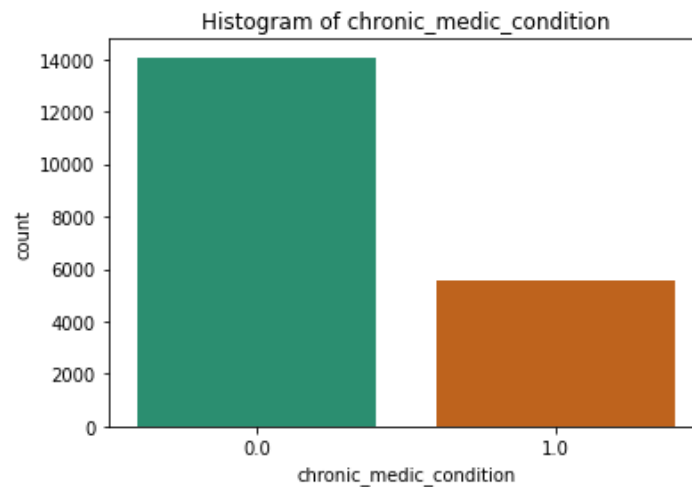
```



```

# 13 chronic_medical_condition [0 and 1]
df_vac.chronic_medical_condition.isnull().sum() #0 Missing values
df_vac.chronic_medical_condition.value_counts()
'''
0.0    14066
1.0     5576'''
# Bar Plot
sns.countplot(x = 'chronic_medical_condition', data = df_vac , palette = 'Dark2')
plt.title('Histogram of chronic_medical_condition')
#Hypothesis Testing
ct_chronic = pd.crosstab(df_vac.h1n1_vaccine, df_vac.chronic_medical_condition)
chi2_contingency(ct_chronic, correction = False)
# p_val = 1.3e-49, Ho reject, hence association exists, good predictor

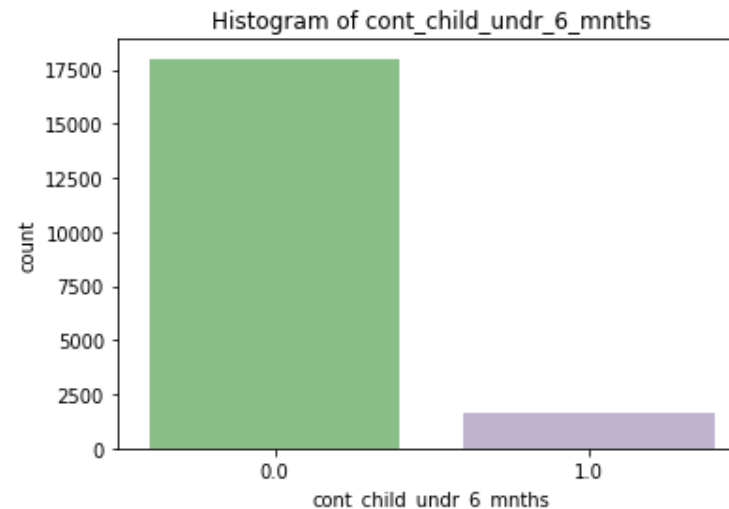
```



```

#_____ 14 cont_child_undr_6_mnth [0 and 1]
df_vac.cont_child_undr_6_mnth.isnull().sum() #0 Missing values
df_vac.cont_child_undr_6_mnth.value_counts()
'''
0.0    17995
1.0     1647'''
# Bar Plot
sns.countplot(x = 'cont_child_undr_6_mnth', data = df_vac , palette = 'Accent')
plt.title('Histogram of cont_child_undr_6_mnth')
#Hypothesis Testing
ct_child = pd.crosstab(df_vac.h1n1_vaccine, df_vac.cont_child_undr_6_mnth)
chi2_contingency(ct_child, correction = False)
# p_val = 9.2e-26, Ho reject, hence association exists, good predictor

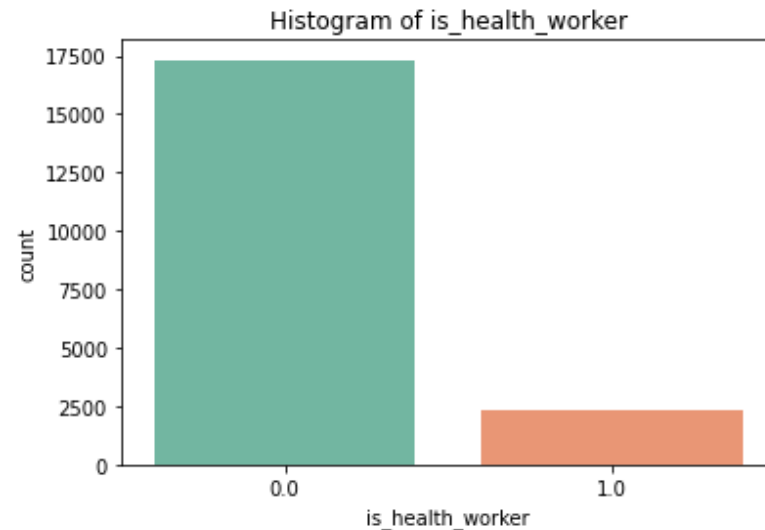
```



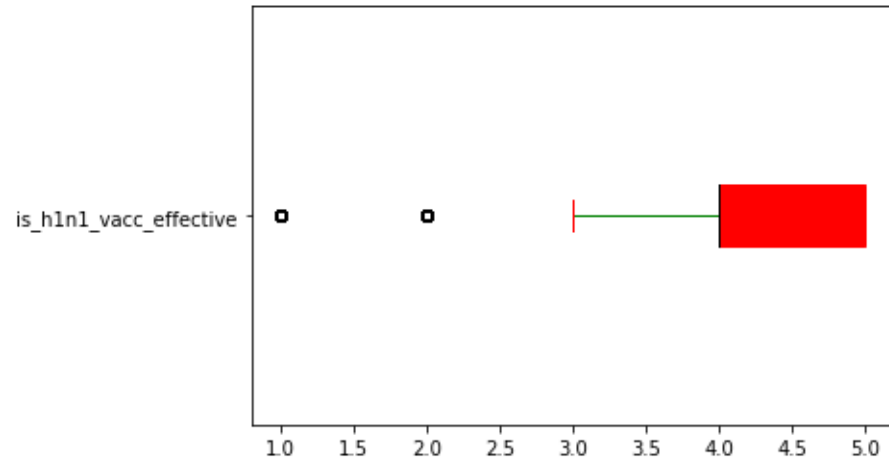
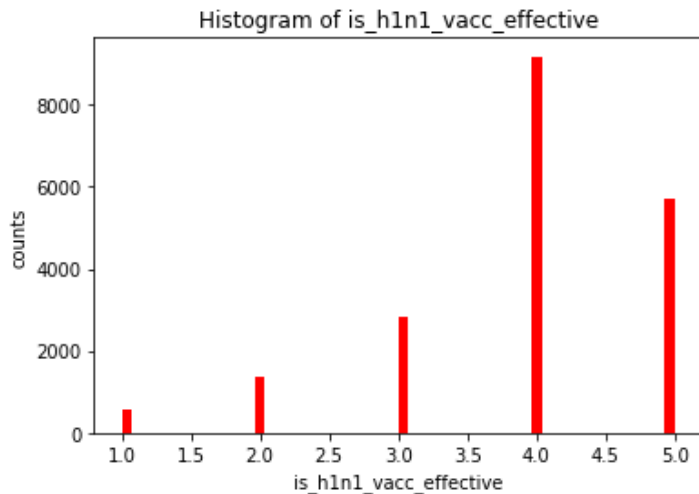
```

# 15 is_health_worker [0 and 1]
df_vac.is_health_worker.isnull().sum() #0 Missing values
df_vac.is_health_worker.value_counts()
'''
0.0    17310
1.0     2332'''
# Bar Plot
sns.countplot(x = 'is_health_worker', data = df_vac , palette = 'Set2')
plt.title('Histogram of is_health_worker')
#Hypothesis Testing
ct_hw = pd.crosstab(df_vac.h1n1_vaccine, df_vac.is_health_worker)
chi2_contingency(ct_hw, correction = False)
# p_val = 4e-152, Ho reject, hence association exists, good predictor

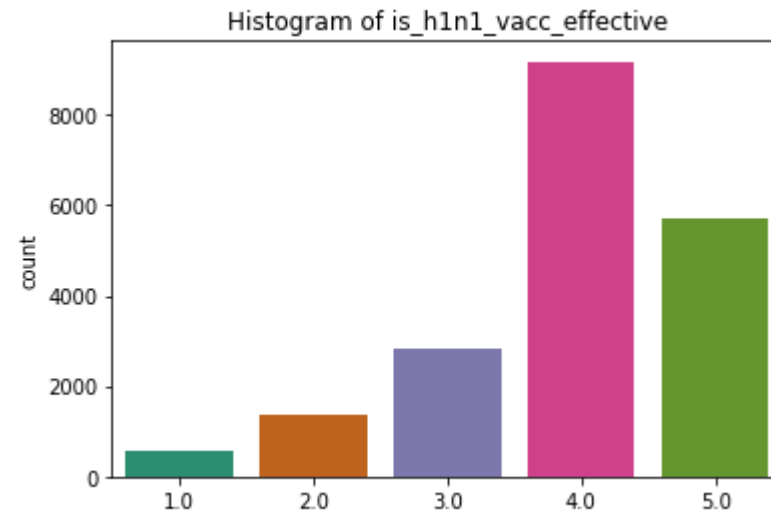
```



#_____ 16 is_h1n1_vacc_effective [1,2,3,4,5] ordered



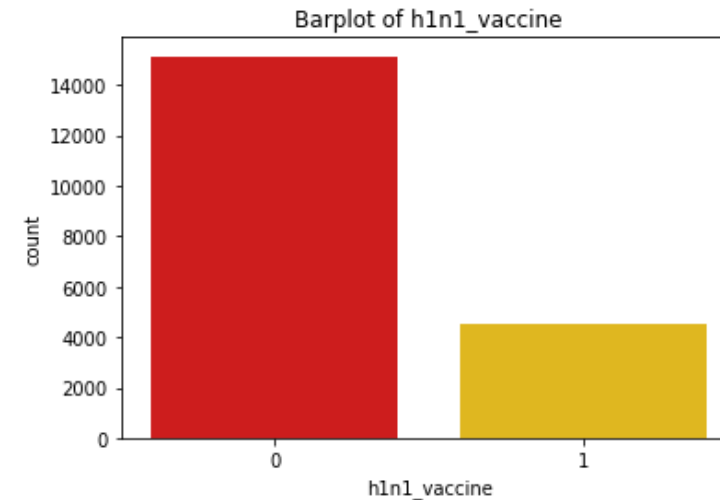
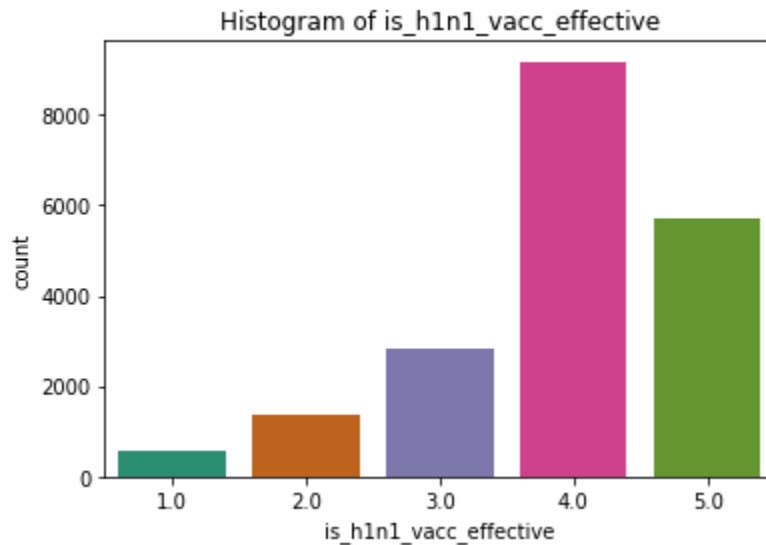
```
#_____ 16 is_h1n1_vacc_effective [1,2,3,4,5] ordered
#_____ histogram
#_run in block
plt.hist(df_vac.is_h1n1_vacc_effective, bins = 'auto', facecolor = 'red')
plt.xlabel('is_h1n1_vacc_effective')
plt.ylabel('counts')
plt.title('Histogram of is_h1n1_vacc_effective')
#_____ boxplot
props2 = dict(boxes = 'red', whiskers = 'green', medians = 'black', caps = 'red')
df_vac['is_h1n1_vacc_effective'].plot.box(color=props2, patch_artist = True, vert = False)
# few outliers on lower side; IGNORE!
df_vac.is_h1n1_vacc_effective.isnull().sum() #0 Missing values
df_vac.is_h1n1_vacc_effective.value_counts()
'''
4.0    9172
5.0    5715
3.0    2838
2.0    1347
1.0     570'''
```



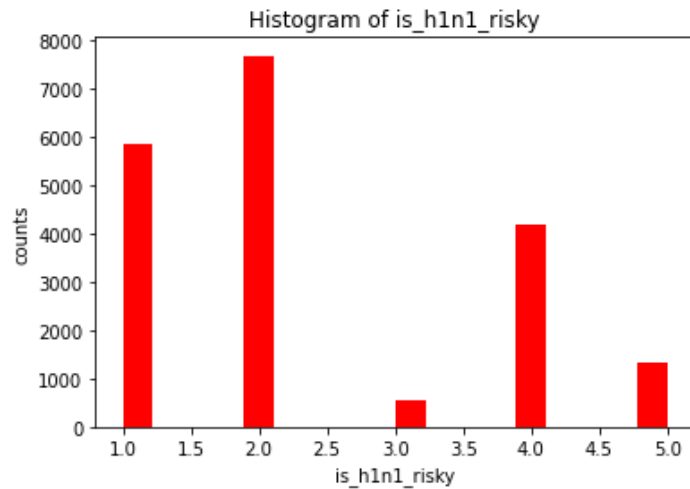
```
# Bar Plot
sns.countplot(x = 'is_h1n1_vacc_effective', data = df_vac , palette = 'Dark2')
plt.title('Histogram of is_h1n1_vacc_effective')
```

Is this a good predictor?

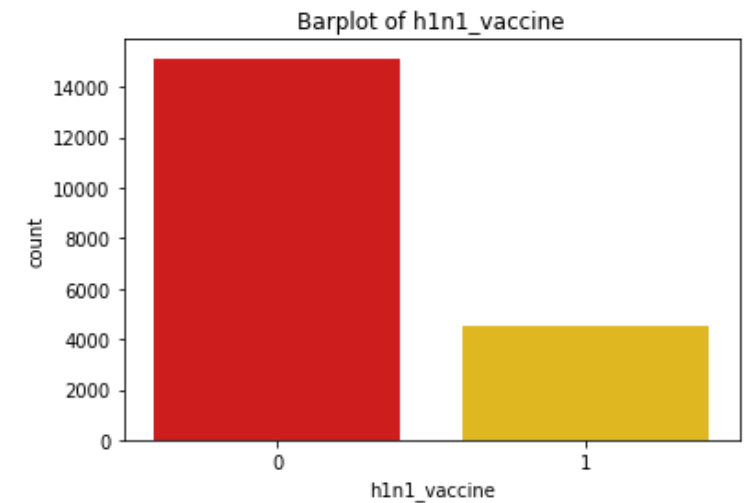
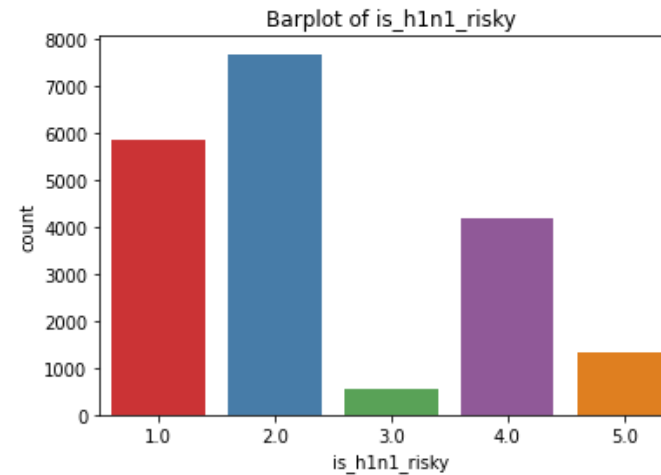
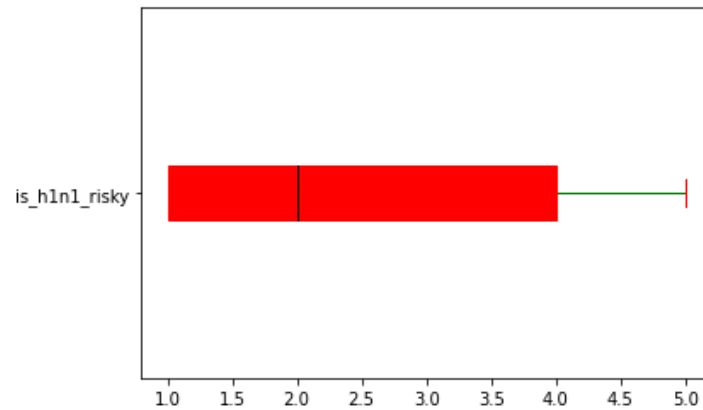
```
#Hypothesis Testing  
import statsmodels.api as sm  
from statsmodels.formula.api import ols  
mod = ols('is_h1n1_vacc_effective~ h1n1_vaccine', data = df_vac).fit()  
aov_table = sm.stats.anova_lm(mod)  
print(aov_table)  
#0.0 ie p_value which is <0.05; Ho Reject; Good Predictor
```



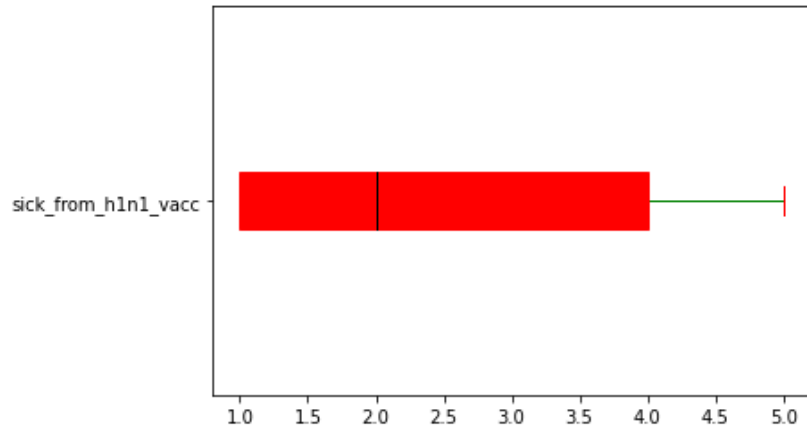
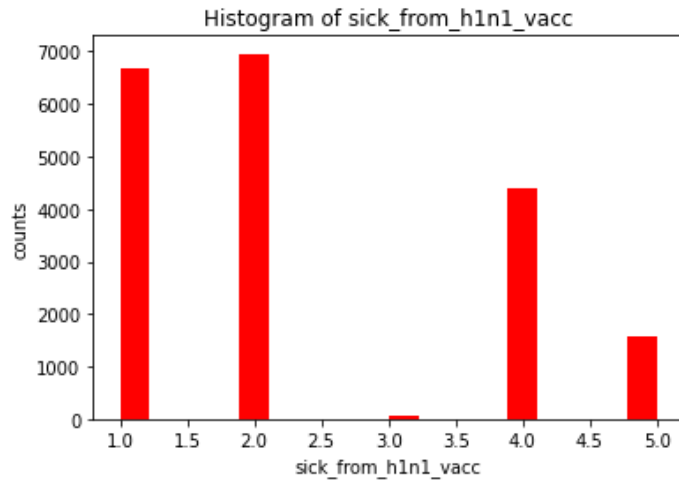
```
# _____ 17 is_h1n1_risky [1,2,3,4,5] ordered
```



```
#Hypothesis Testing
import statsmodels.api as sm
from statsmodels.formula.api import ols
mod = ols('is_h1n1_risky ~ h1n1_vaccine', data = df_vac).fit()
aov_table = sm.stats.anova_lm(mod)
print(aov_table)
#0.0 ie p_value is <0.05; Ho Reject; Good Predictor
```

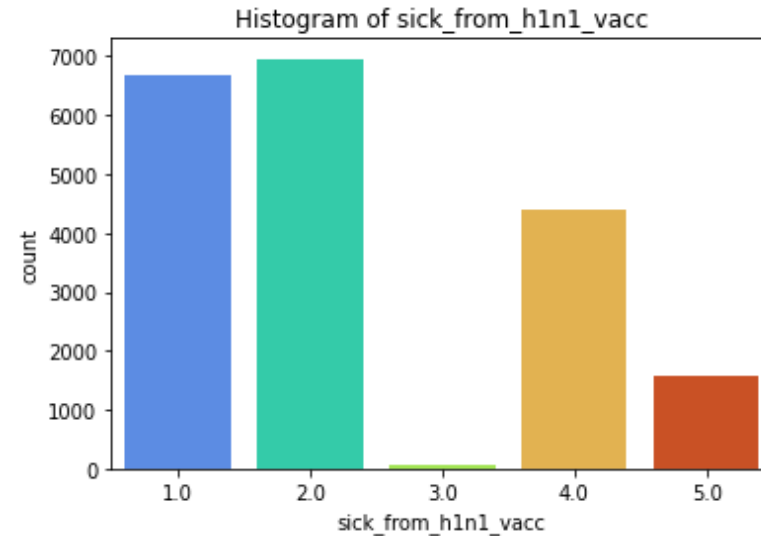


_____ 18 sick_from_h1n1_vacc [1,2,3,4,5] ordered



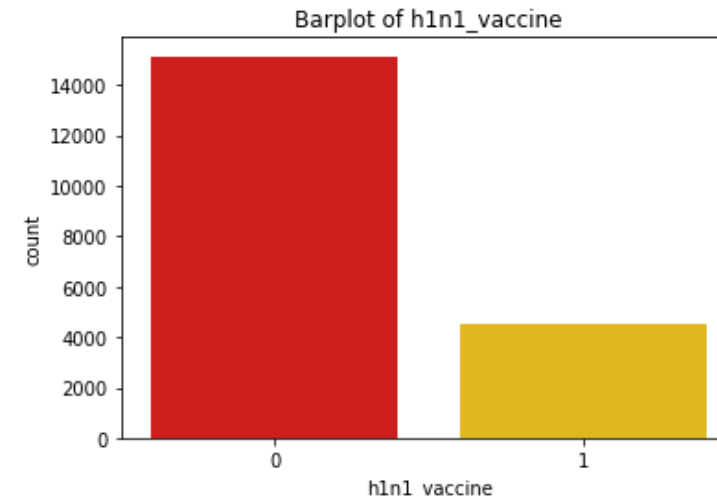
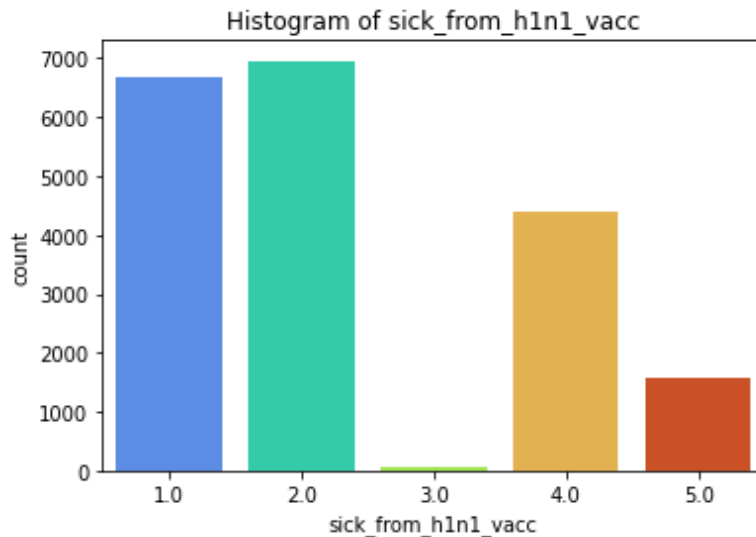
```
# _____ 18 sick_from_h1n1_vacc [1,2,3,4,5] ordered
# _____ histogram
#_run in block
plt.hist(df_vac.sick_from_h1n1_vacc, bins = 'auto', facecolor = 'red')
plt.xlabel('sick_from_h1n1_vacc')
plt.ylabel('counts')
plt.title('Histogram of sick_from_h1n1_vacc')
#_boxplot
props2 = dict(boxes = 'red', whiskers = 'green', medians = 'black', caps = 'red')
df_vac['sick_from_h1n1_vacc'].plot.box(color=props2, patch_artist = True, vert = False)

df_vac.sick_from_h1n1_vacc.isnull().sum() #0 Missing values
df_vac.sick_from_h1n1_vacc.value_counts()
'''
2.0    6956
1.0    6684
4.0    4390
5.0    1560
3.0      52'''
```



Is this a good predictor?

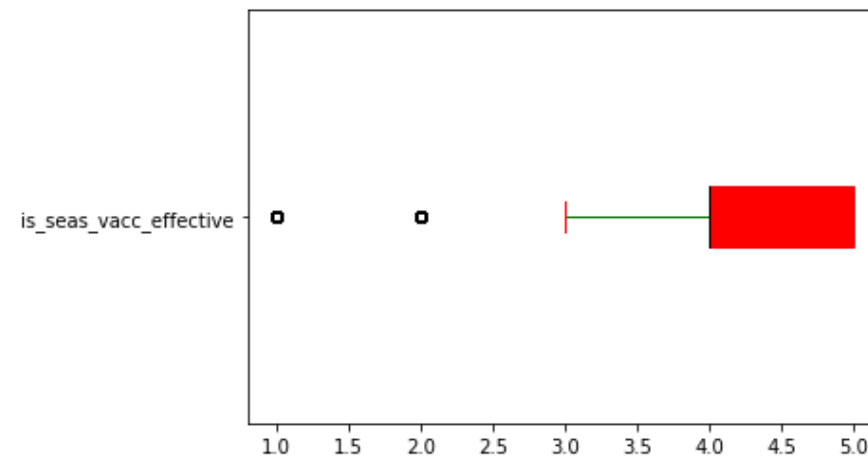
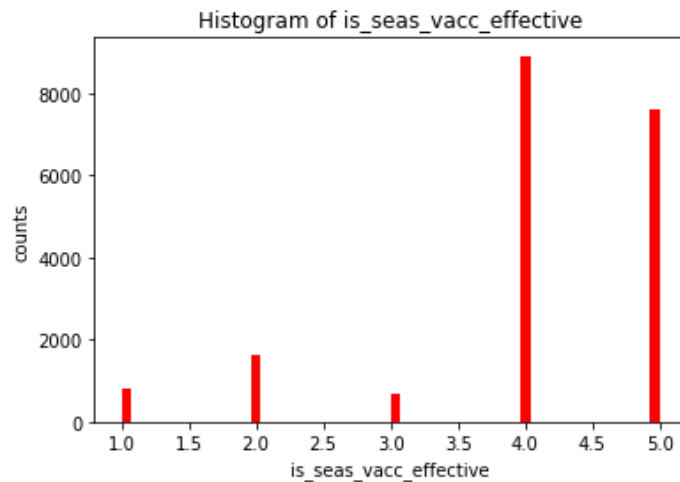
```
#Hypothesis Testing  
import statsmodels.api as sm  
from statsmodels.formula.api import ols  
mod = ols('sick_from_h1n1_vacc ~ h1n1_vaccine', data = df_vac).fit()  
aov_table = sm.stats.anova_lm(mod)  
print(aov_table)  
#3.62e-31 ie p_value is <0.05; Ho Reject; Good Predictor
```



```
#_____ 19 is_seas_vacc_effective [1,2,3,4,5] ordered
#_____ histogram
#_run in block
plt.hist(df_vac.is_seas_vacc_effective, bins = 'auto', facecolor = 'red')
plt.xlabel('is_seas_vacc_effective')
plt.ylabel('counts')
plt.title('Histogram of is_seas_vacc_effective')
#_____ boxplot
props2 = dict(boxes = 'red', whiskers = 'green', medians = 'black', caps = 'red')
df_vac['is_seas_vacc_effective'].plot.box(color=props2, patch_artist = True, vert = False)
# few are on lower side; Ignore outliers
```

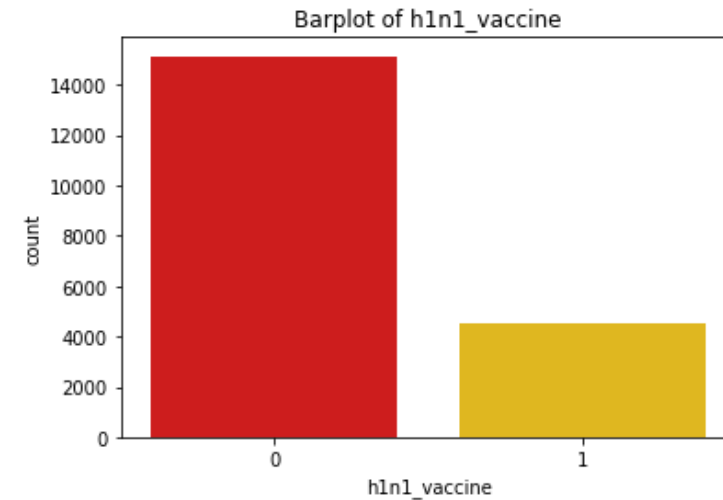
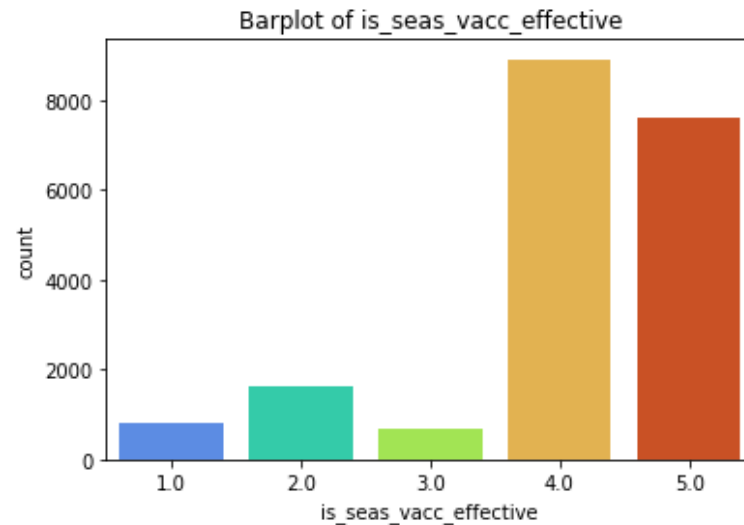
```
df_vac.is_seas_vacc_effective.isnull().sum() #0 Missing values
df_vac.is_seas_vacc_effective.value_counts()
```

```
'''
4.0    8906
5.0    7603
2.0    1638
1.0     822
3.0     673'''
```

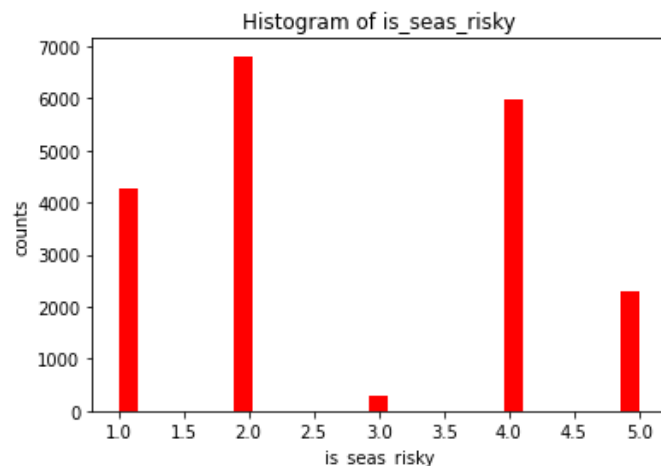


Is this a good predictor?

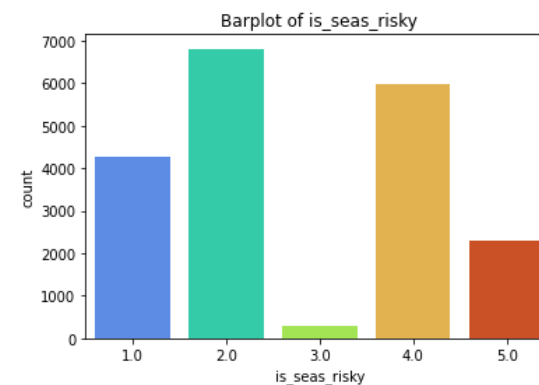
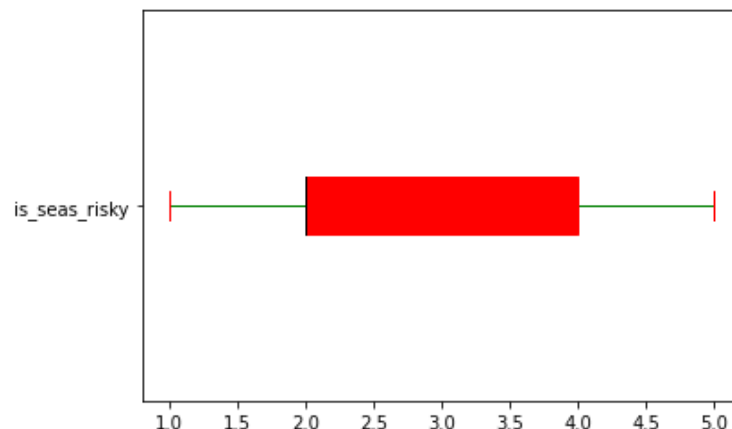
```
#Hypothesis Testing  
import statsmodels.api as sm  
from statsmodels.formula.api import ols  
mod = ols('is_seas_vacc_effective ~ h1n1_vaccine', data = df_vac).fit()  
aov_table = sm.stats.anova_lm(mod)  
print(aov_table)  
#9.2e-152 ie p_value is <0.05; Ho Reject; Good Predictor
```



_____ 20 is_seas_risky [1,2,3,4,5] ordered

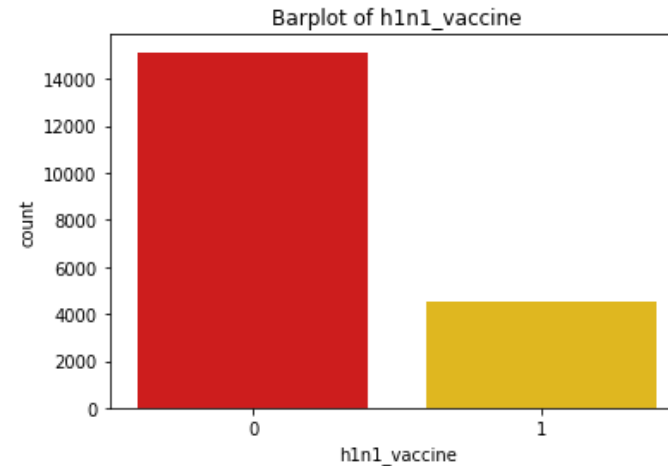
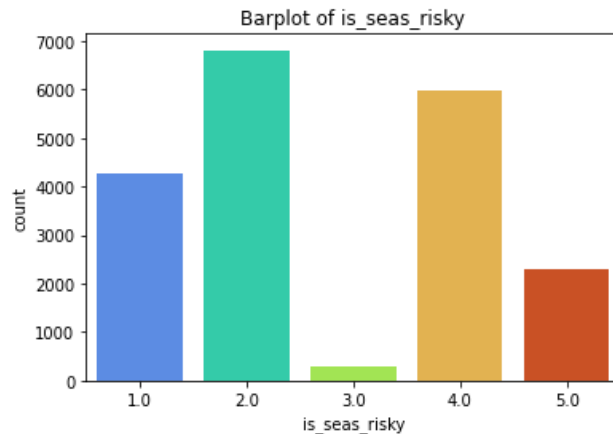


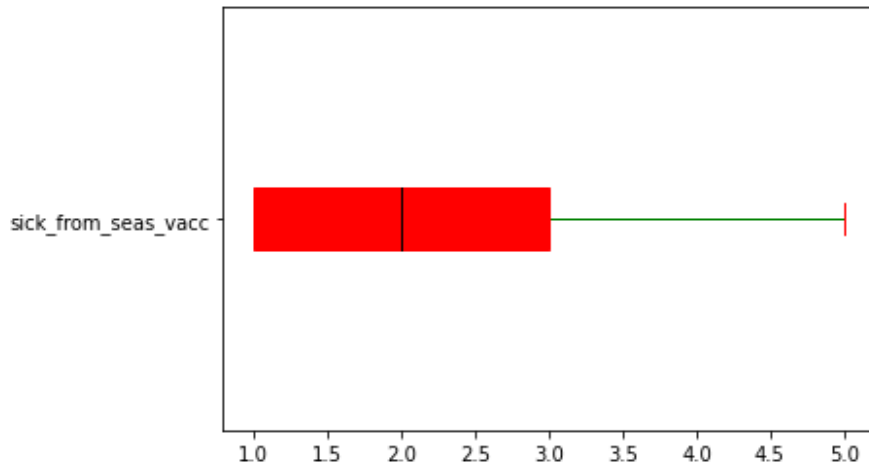
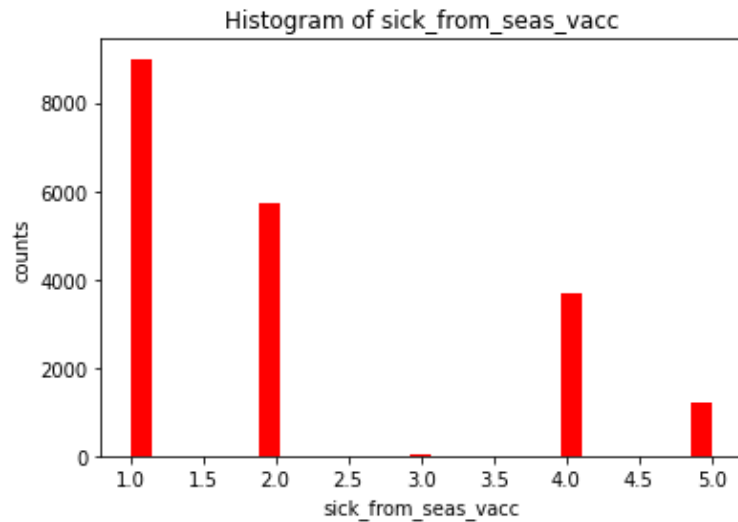
```
#_____histogram
#_run in block
plt.hist(df_vac.is_seas_risky, bins = 'auto', facecolor = 'red')
plt.xlabel('is_seas_risky')
plt.ylabel('counts')
plt.title('Histogram of is_seas_risky')
#_____boxplot
props2 = dict(boxes = 'red', whiskers = 'green', medians = 'black', caps = 'red')
df_vac['is_seas_risky'].plot.box(color=props2, patch_artist = True, vert = False)
# no outliers
df_vac.is_seas_risky.isnull().sum() #0 Missing values
df_vac.is_seas_risky.value_counts()
'''
2.0    6811
4.0    5984
1.0    4258
5.0    2286
3.0     303'''
# Bar Plot
sns.countplot(x = 'is_seas_risky', data = df_vac , palette = 'turbo')
plt.title('Barplot of is_seas_risky')
```



Is this a good predictor?

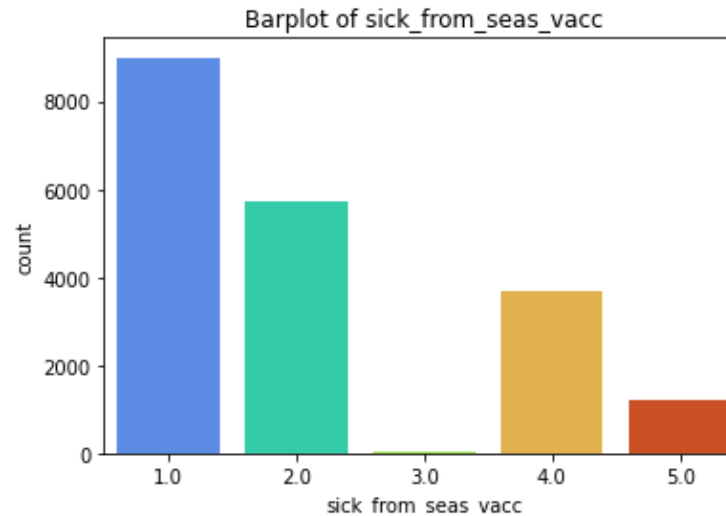
```
#Hypothesis Testing  
import statsmodels.api as sm  
from statsmodels.formula.api import ols  
mod = ols('is_seas_risky ~ h1n1_vaccine', data = df_vac).fit()  
aov_table = sm.stats.anova_lm(mod)  
print(aov_table)  
#0.0 ie p_value which is <0.05; Ho Reject; Good Predictor
```





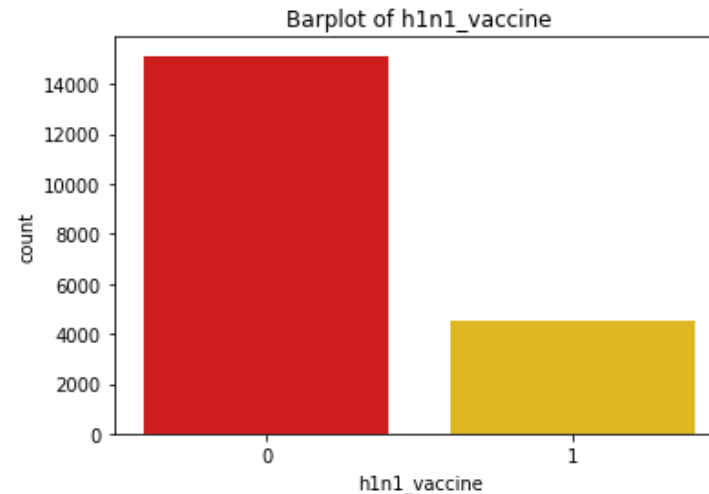
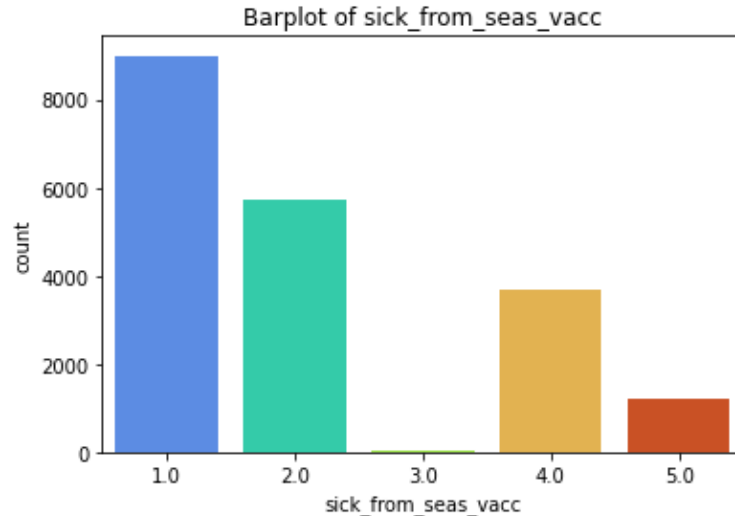
```
#_____ 21 sick_from_seas_vacc [1,2,3,4,5] ordered
#_____ histogram
#_run in block
plt.hist(df_vac.sick_from_seas_vacc, bins = 'auto', facecolor = 'red')
plt.xlabel('sick_from_seas_vacc')
plt.ylabel('counts')
plt.title('Histogram of sick_from_seas_vacc')
#_____ boxplot
props2 = dict(boxes = 'red', whiskers = 'green', medians = 'black', caps = 'red')
df_vac['sick_from_seas_vacc'].plot.box(color=props2, patch_artist = True, vert = False)

df_vac.sick_from_seas_vacc.isnull().sum() #0 Missing values
df_vac.sick_from_seas_vacc.value_counts()
'''
1.0    8996
2.0    5713
4.0    3683
5.0    1221
3.0      29'''
# Bar Plot
sns.countplot(x = 'sick_from_seas_vacc', data = df_vac , palette = 'turbo')
plt.title('Barplot of sick_from_seas_vacc')
```

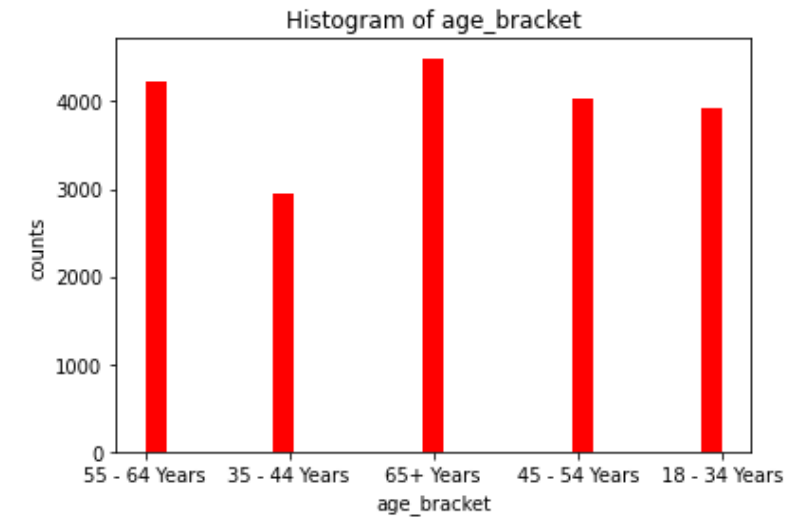


Is this a good predictor?

```
#Hypothesis Testing  
import statsmodels.api as sm  
from statsmodels.formula.api import ols  
mod = ols('sick_from_seas_vacc ~ h1n1_vaccine', data = df_vac).fit()  
aov_table = sm.stats.anova_lm(mod)  
print(aov_table)  
#0.36 ie p_value is >0.05; Ho accepted; Bad Predictor
```



```
#_____ 22 age_bracket [actually ordered]
```



```
#_____boxplot
props2 = dict(boxes = 'red', whiskers = 'green', medians = 'black', caps = 'red')
df_vac['age_bracket'].plot.box(color=props2, patch_artist = True, vert = False)
## oops, its object, lets change to 1,2,3 like; first see value counts
df_vac.age_bracket.isnull().sum() #0 Missing values
df_vac.age_bracket.value_counts()
'''
65+ Years          4491
55 - 64 Years      4234
45 - 54 Years      4038
18 - 34 Years      3925
35 - 44 Years      2954
'''
```

```
# Let categories be in order
```

```
df_vac['age_bracket'] =df_vac.get('age_bracket').replace('65+ Years', 5)  
df_vac['age_bracket'] =df_vac.get('age_bracket').replace('55 - 64 Years', 4)  
df_vac['age_bracket'] =df_vac.get('age_bracket').replace('45 - 54 Years', 3)  
df_vac['age_bracket'] =df_vac.get('age_bracket').replace('18 - 34 Years', 1)  
df_vac['age_bracket'] =df_vac.get('age_bracket').replace('35 - 44 Years', 2)
```

```
# ignore warnings !
```

```
df_vac.age_bracket.isnull().sum() #0 Missing values
```

```
df_vac.age_bracket.value_counts()
```

```
'''
```

```
5      4491
```

```
4      4234
```

```
3      4038
```

```
1      3925
```

```
2      2954
```

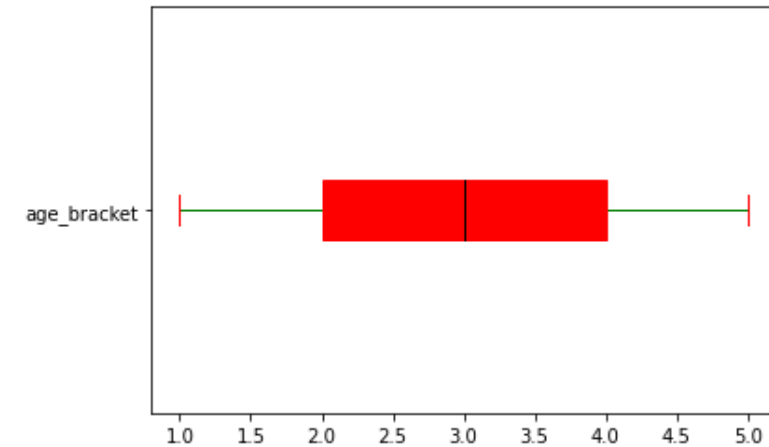
```
Name: age_bracket, dtype: int64
```

```
'''
```

```
# now boxplot will come
```

```
props2 = dict(boxes = 'red', whiskers = 'green', medians = 'black', caps = 'red')
```

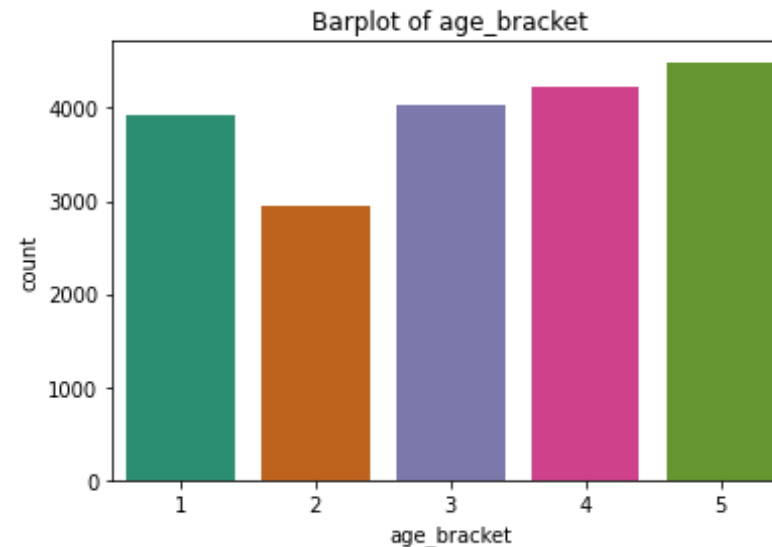
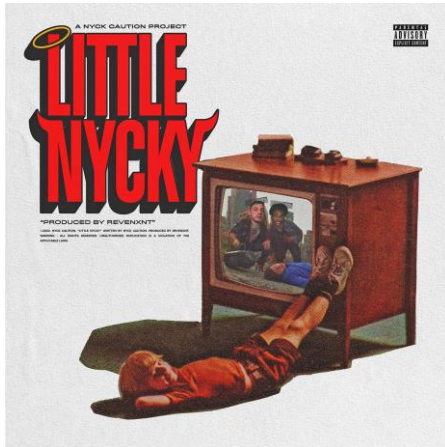
```
df_vac['age_bracket'].plot.box(color=props2, patch_artist = True, vert = False) #No outliers
```



Caution!

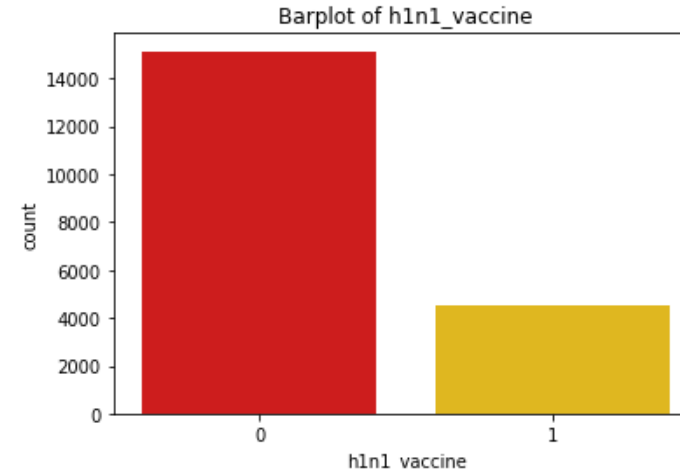
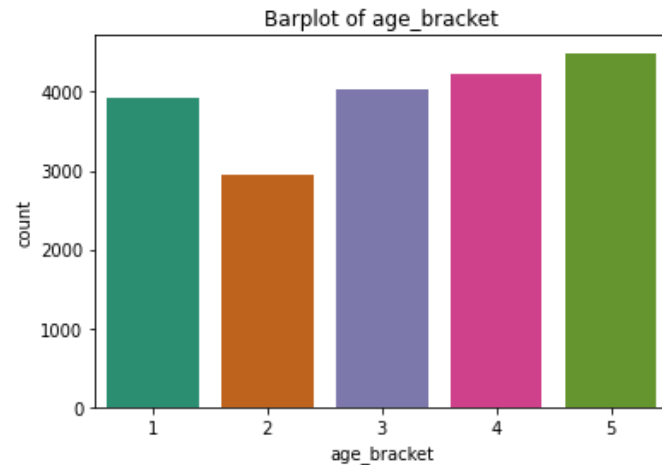
```
# _____ we could have followed a more easier way!  
# Label encoding the data ; its good for nominal data , not good for ordered data  
# Like in our present case!  
# DO NOT TRY AS IT HAS ALREADY BEING DONE !!!!  
from sklearn.preprocessing import LabelEncoder  
le = LabelEncoder()
```

```
# Bar Plot  
sns.countplot(x = 'age_bracket', data = df_vac , palette = 'Dark2')  
plt.title('Barplot of age_bracket')
```



Is this a good predictor?

```
#Hypothesis Testing  
import statsmodels.api as sm  
from statsmodels.formula.api import ols  
mod = ols('age_bracket ~ h1n1_vaccine', data = df_vac).fit()  
aov_table = sm.stats.anova_lm(mod)  
print(aov_table)  
#1.5e-10 ie p_value is <0.05; Ho rejected; Good Predictor
```



```
# 23 qualification - object, Actually ordered! 3 levels
```

```
df_vac.qualification.isnull().sum()  
df_vac.qualification.value_counts()  
'''
```

```
College Graduate      8165  
Some College         5570  
12 Years             4287  
< 12 Years          1620'''
```

```
# Let's put them in order
```

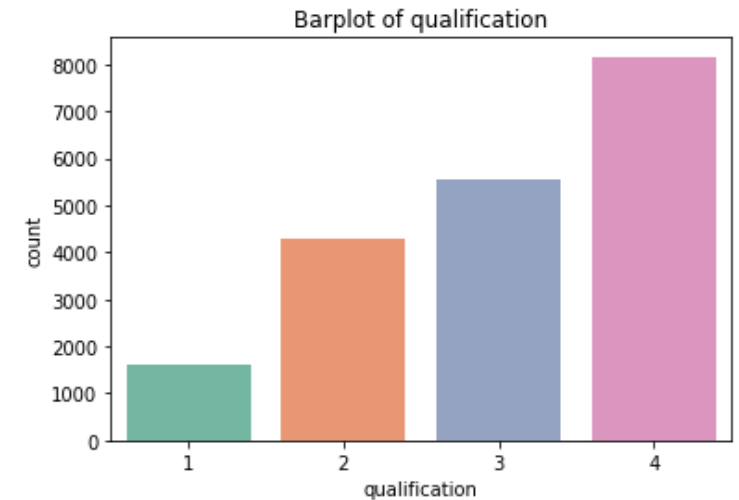
```
df_vac['qualification'] =df_vac.get('qualification').replace('College Graduate', 4)  
df_vac['qualification'] =df_vac.get('qualification').replace('Some College', 3)  
df_vac['qualification'] =df_vac.get('qualification').replace('12 Years', 2)  
df_vac['qualification'] =df_vac.get('qualification').replace('< 12 Years', 1)
```

```
df_vac.qualification.value_counts()  
'''
```

```
4      8165  
3      5570  
2      4287  
1      1620  
'''
```

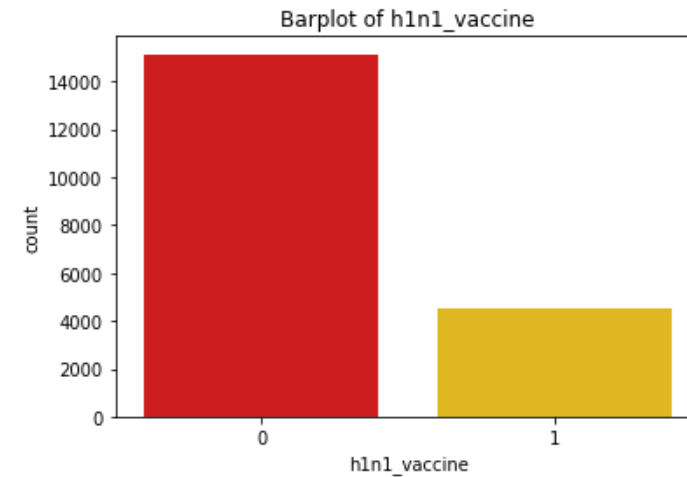
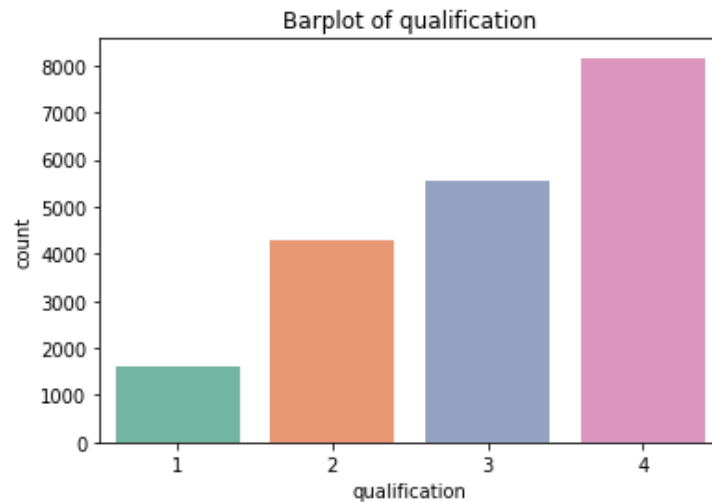
```
# Bar Plot
```

```
sns.countplot(x = 'qualification', data = df_vac , palette = 'Set2')  
plt.title('Barplot of qualification')
```



Is this a good predictor?

```
#Hypothesis Testing  
import statsmodels.api as sm  
from statsmodels.formula.api import ols  
mod = ols('qualification ~ h1n1_vaccine', data = df_vac).fit()  
aov_table = sm.stats.anova_lm(mod)  
print(aov_table)  
#1.57e-22 ie p_value is <0.05; Ho rejected; Good Predictor
```



```

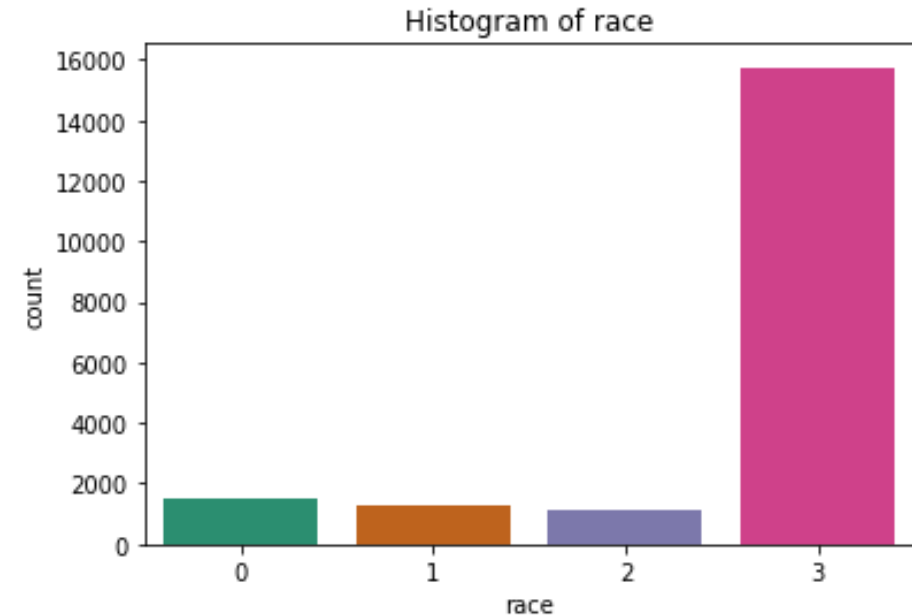
# _____ 24 race - object [NO ORDER, NOMINAL]; 4 Levels
df_vac.race.isnull().sum() #No Missing values
df_vac.race.value_counts()
'''
White          15745
Black          1474
Hispanic       1295
Other or Multiple 1128'''

# Label encoding 'race' ; does alphabetically!
# HERE WE CAN USE LabelEncoder!
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

df_vac['race']= le.fit_transform(df_vac['race'])
df_vac.race.value_counts()
'''
3      15745
0       1474
1       1295
2       1128'''

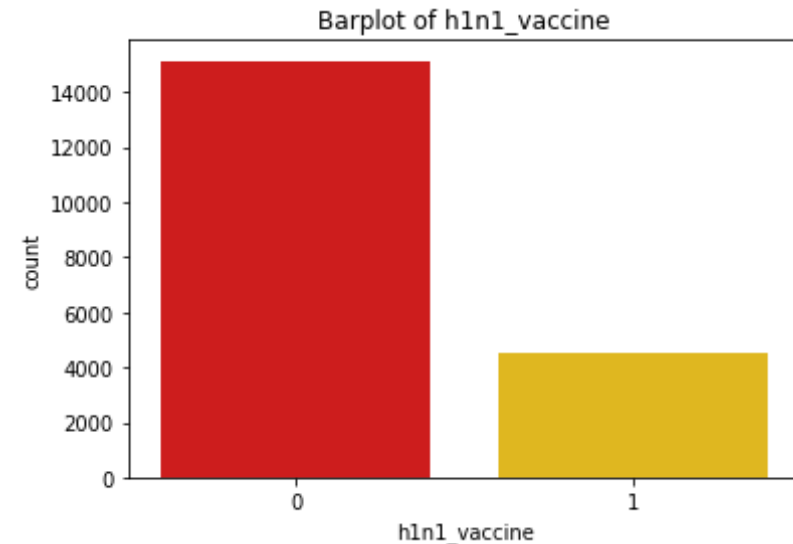
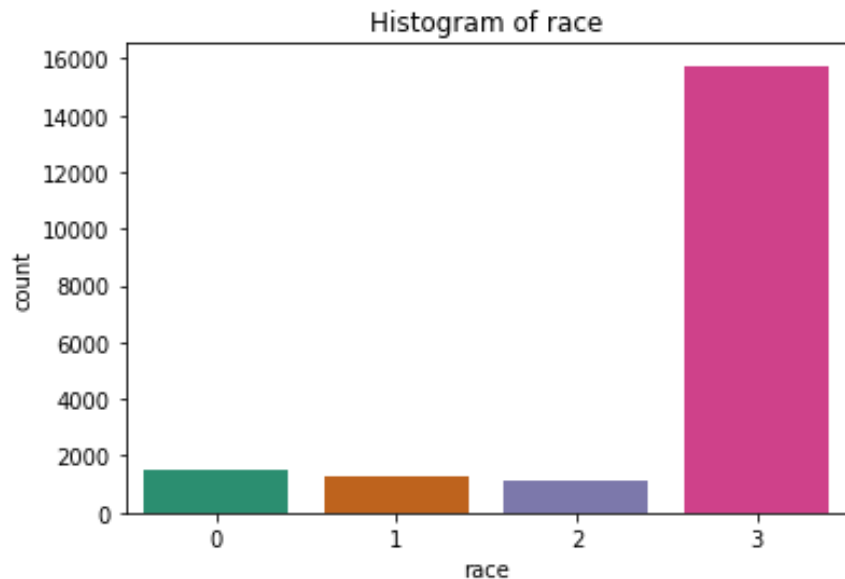
# Bar Plot
sns.countplot(x = 'race', data = df_vac , palette = 'Dark2')
plt.title('Histogram of race')

```



Is this a good predictor?

```
#Hypothesis Testing  
from scipy.stats import chi2_contingency  
ct_race = pd.crosstab(df_vac.h1n1_vaccine, df_vac.race)  
chi2_contingency(ct_race, correction = False)  
# p_val = 2.4e-10, Ho reject, hence association exists, good predictor
```



```

# _____ 25 sex - object [female, male]
df_vac.sex.isnull().sum() #No Missing values
df_vac.sex.value_counts()
'''
Female    11638
Male      8004'''

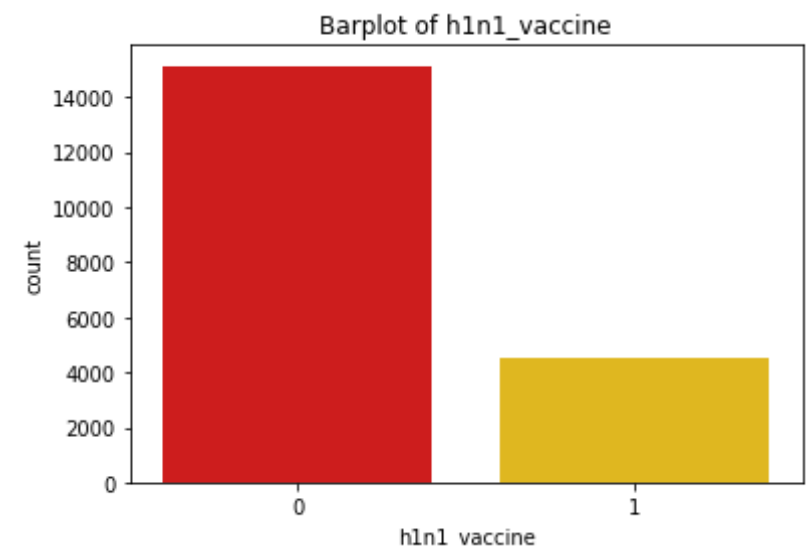
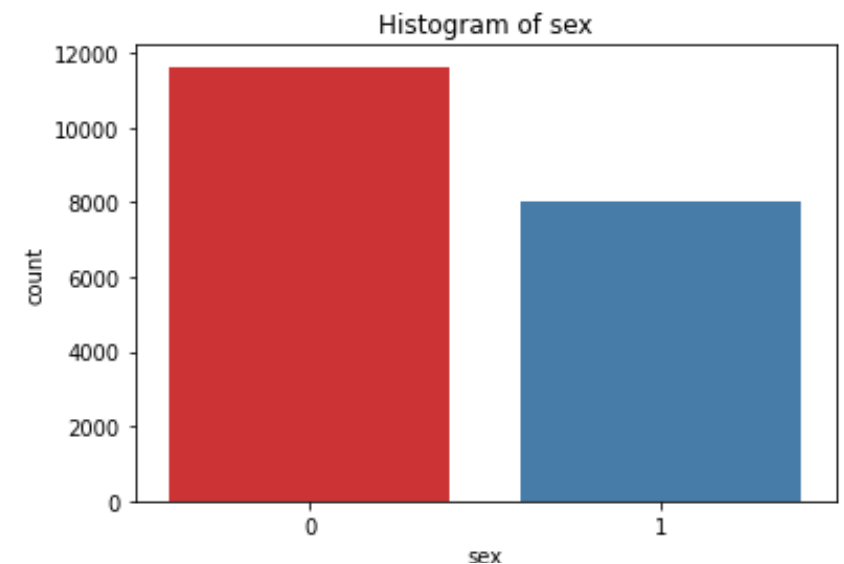
# Label encoding the data
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

df_vac['sex'] = le.fit_transform(df_vac['sex'])
df_vac.sex.value_counts()
'''
0      11638
1      8004'''

# Bar Plot
sns.countplot(x = 'sex', data = df_vac , palette = 'Set1')
plt.title('Histogram of sex')

#Hypothesis Testing
from scipy.stats import chi2_contingency
ct_sex = pd.crosstab(df_vac.h1n1_vaccine, df_vac.sex)
chi2_contingency(ct_sex, correction = False)
# p_val = 0.0, Ho reject, hence association exists, good predictor

```



```

#_____ 26 income_level - object, its ordered
df_vac.income_level.isnull().sum() # no missing values
df_vac.income_level.value_counts()
'''
<= $75,000, Above Poverty    11185
> $75,000                    6159
Below Poverty                2298'''

#Converting to numeric/ integer
df_vac['income_level']=df_vac.get('income_level').replace('Below Poverty', 1)
df_vac['income_level']=df_vac.get('income_level').replace('<= $75,000, Above Poverty', 2)
df_vac['income_level']=df_vac.get('income_level').replace('> $75,000', 3)

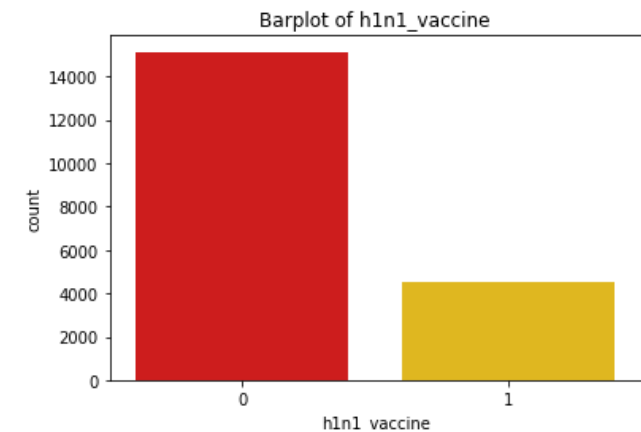
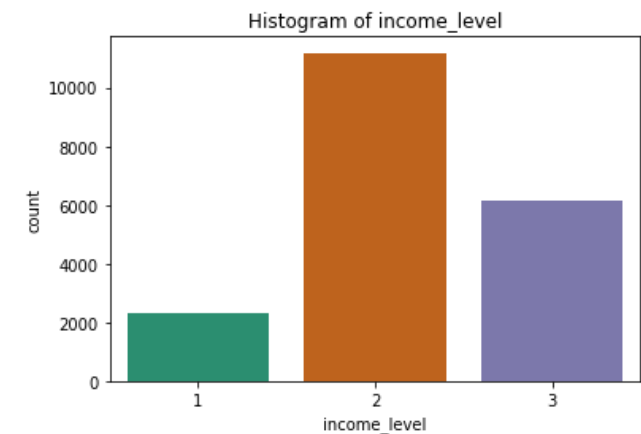
df_vac.income_level.value_counts()
'''
2      11185
3       6159
1       2298
'''

df_vac.info()

# Bar Plot
sns.countplot(x = 'income_level', data = df_vac , palette = 'Dark2')
plt.title('Histogram of income_level')

#Hypothesis Testing
import statsmodels.api as sm
from statsmodels.formula.api import ols
mod = ols('income_level ~ h1n1_vaccine', data = df_vac).fit()
aov_table = sm.stats.anova_lm(mod)
print(aov_table)
# 2.62e-15 ie p_value is <0.05; Ho rejected; Good Predictor

```



```

#_____27 marital_status - object [0,1]
df_vac.marital_status.isnull().sum() #471 Missing values
df_vac.marital_status.value_counts()
'''
Married          10768
Not Married       8874'''

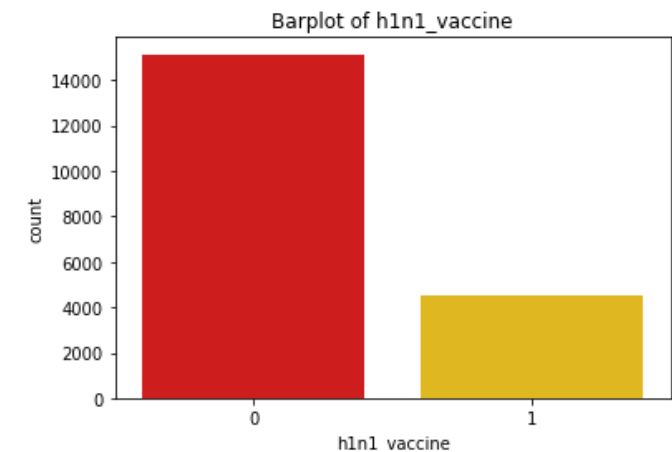
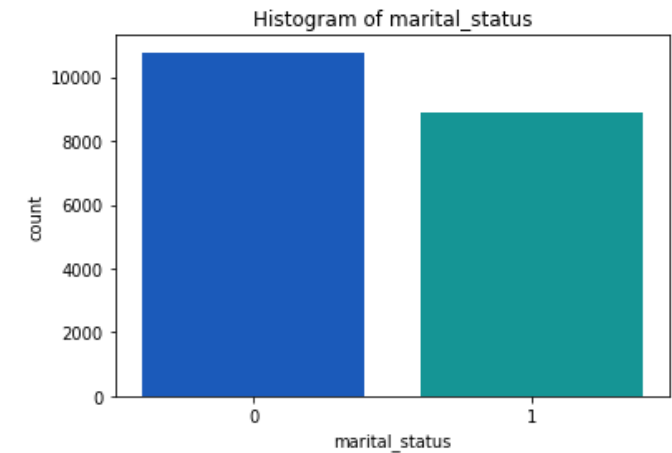
# Label encoding the data
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

df_vac['marital_status'] = le.fit_transform(df_vac['marital_status'])
df_vac.marital_status.value_counts()
'''
0      10768
1       8874'''
df_vac.info()

# Bar Plot
sns.countplot(x = 'marital_status', data = df_vac , palette = 'winter')
plt.title('Histogram of marital_status')

#Hypothesis Testing
from scipy.stats import chi2_contingency
ct_mari = pd.crosstab(df_vac.h1n1_vaccine, df_vac.marital_status)
chi2_contingency(ct_mari, correction = False)
# p_val = 2.14e-13, Ho reject, hence association exists, good predictor

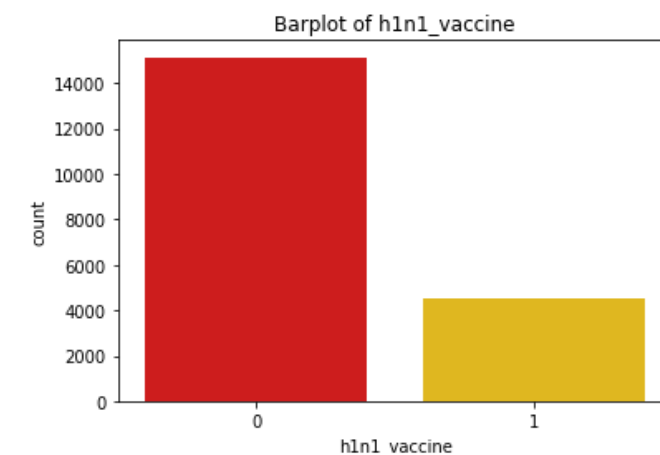
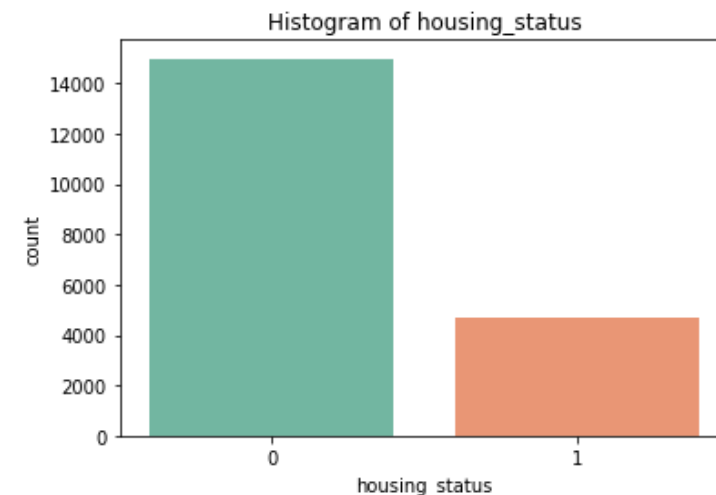
```




```

#_____28 housing_status - object [own, rent]
df_vac.housing_status.isnull().sum()
df_vac.housing_status.value_counts()
'''
Own      14980
Rent     4662'''
# Label encoding the data
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df_vac['housing_status'] = le.fit_transform(df_vac['housing_status'])
df_vac.housing_status.value_counts()
'''
0      14980
1       4662'''
# Bar Plot
sns.countplot(x = 'housing_status', data = df_vac , palette = 'Set2')
plt.title('Histogram of housing_status')
#Hypothesis Testing
from scipy.stats import chi2_contingency
ct_house = pd.crosstab(df_vac.h1n1_vaccine, df_vac.housing_status)
chi2_contingency(ct_house, correction = False)
# p_val = 7.1e-07, Ho reject, hence association exists, good predictor

```



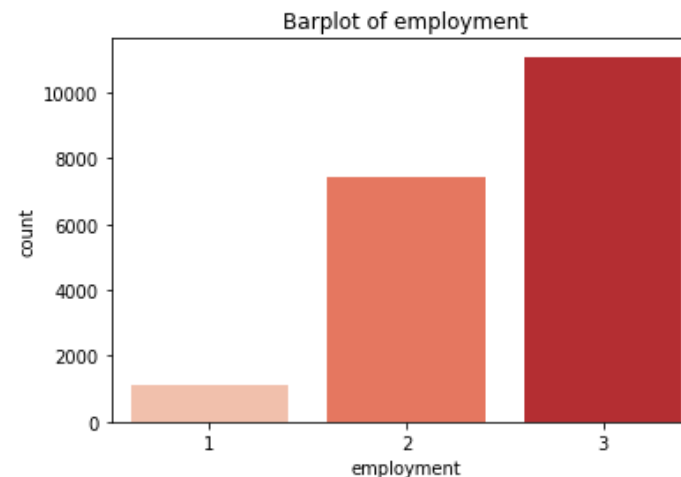
```

# _____ 29 employment - object [3 levels] actually ordered
df_vac.employment.isnull().sum()
df_vac.employment.value_counts()
'''
Employed          11093
Not in Labor Force  7417
Unemployed         1132'''

#Converting to numeric/ integer
df_vac['employment']=df_vac.get('employment').replace('Employed', 3)
df_vac['employment']=df_vac.get('employment').replace('Not in Labor Force', 2)
df_vac['employment']=df_vac.get('employment').replace('Unemployed', 1)

df_vac.employment.value_counts()
'''
3      11093
2       7417
1       1132
'''

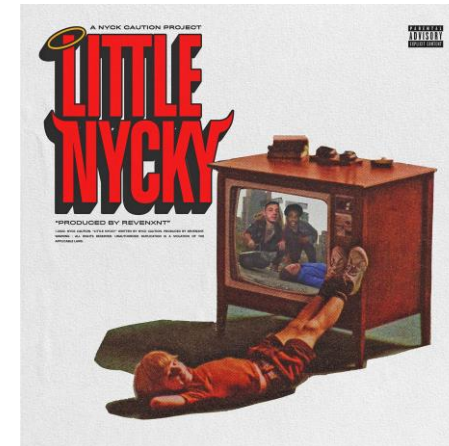
```



```
# LabelEncoder giving opposite notation!
# Label encoding the data ; DO NOT TRY THIS
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

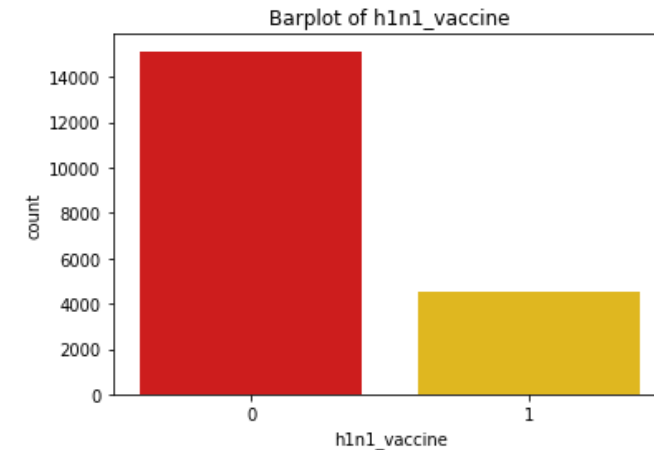
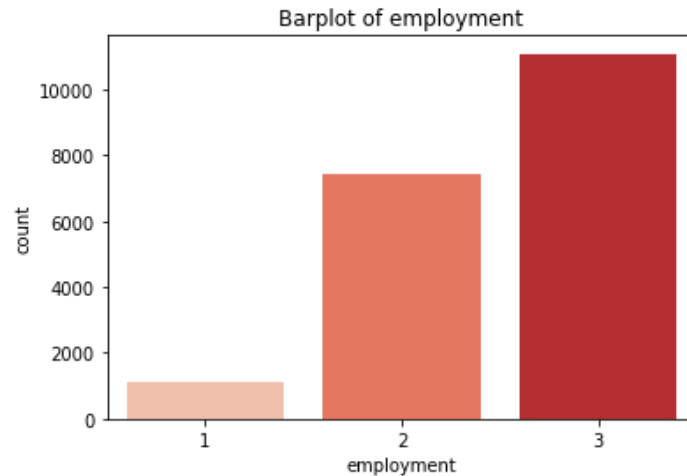
df_vac['employment']= le.fit_transform(df_vac['employment'])
df_vac.employment.value_counts()
'''
0      11093
1       7417
2      1132'''
```

Employed	11093
Not in Labor Force	7417
Unemployed	1132'''



Is this a good predictor?

```
#Hypothesis Testing
import statsmodels.api as sm
from statsmodels.formula.api import ols
mod = ols('employment ~ h1n1_vaccine', data = df_vac).fit()
aov_table = sm.stats.anova_lm(mod)
print(aov_table)
#0.015 ie p_value is <0.05; Ho rejected; Good Predictor
```



```
#_____30 census_msa - object, 3 levels, NOMINAL
df_vac.census_msa.isnull().sum() #No Missing values
df_vac.census_msa.value_counts()
'''
```

```
MSA, Not Principle City    8571
MSA, Principle City        5717
Non-MSA                    5354'''
```



```
# Label encoding the data
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

```
df_vac['census_msa']= le.fit_transform(df_vac['census_msa'])
df_vac.census_msa.value_counts()
'''
```

```
0    8571
1    5717
2    5354'''
```

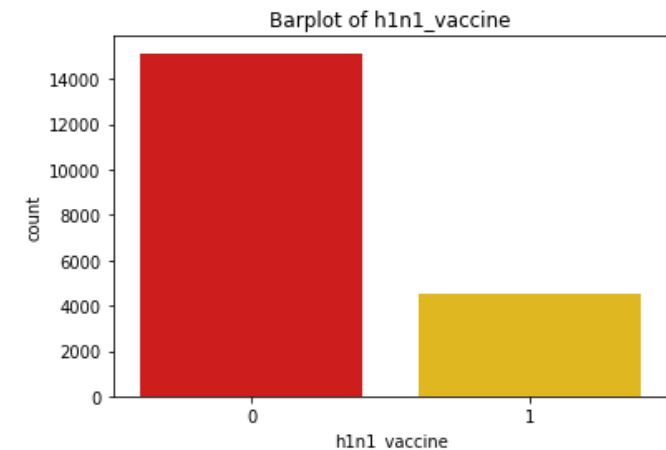
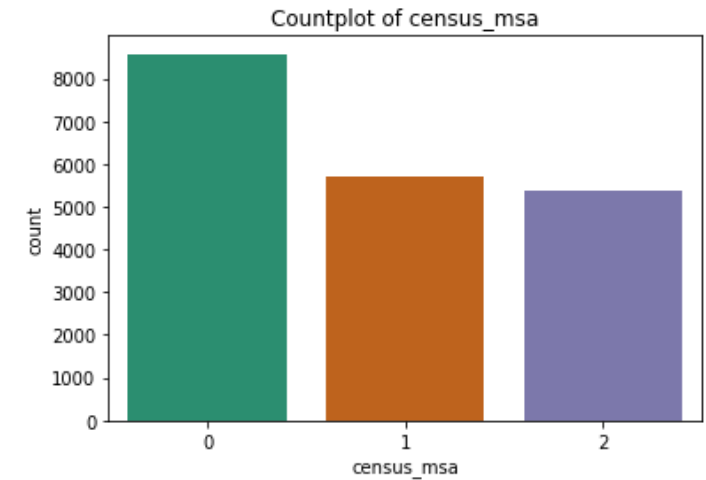
```
# Bar Plot
```

```
sns.countplot(x = 'census_msa', data = df_vac , palette = 'Dark2')
plt.title('Countplot of census_msa')
```

```
#Hypothesis Testing
```

```
from scipy.stats import chi2_contingency
ct_msa = pd.crosstab(df_vac.h1n1_vaccine, df_vac.census_msa)
ct_msa
chi2_contingency(ct_msa, correction = False)
```

```
# p_val = 0.76, > 0.05 Ho accept, hence association does not exists, bad predictor
```



```
#_____31 no_of_adults, ordered
df_vac.no_of_adults.isnull().sum() #No Missing values
df_vac.no_of_adults.value_counts()
'''
```

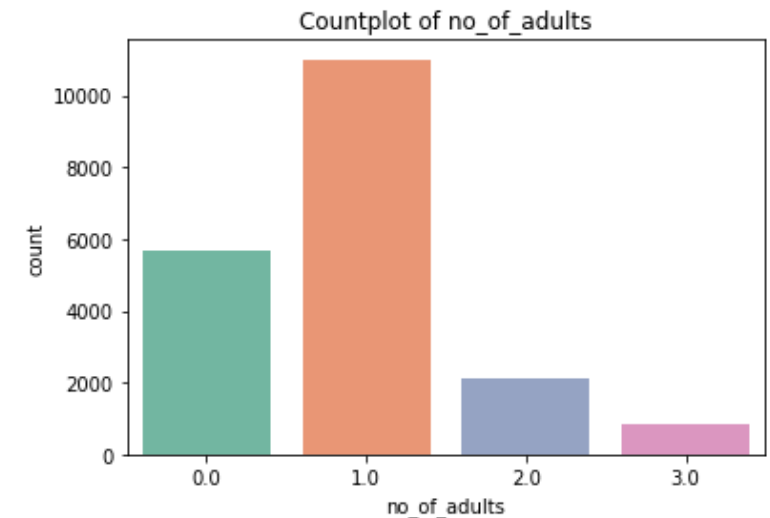
```
1.0    11006
0.0     5683
2.0     2124
3.0      829'''
```

Bar Plot

```
sns.countplot(x = 'no_of_adults', data = df_vac , palette = 'Set2')
plt.title('Countplot of no_of_adults')
```

#Hypothesis Testing

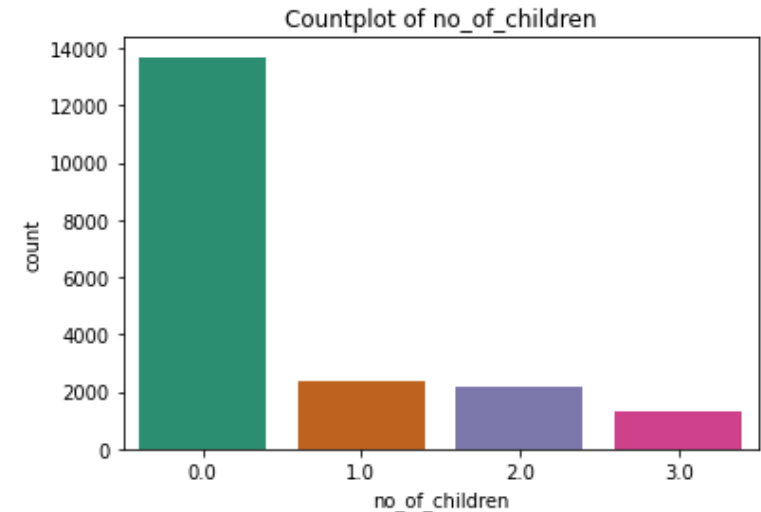
```
import statsmodels.api as sm
from statsmodels.formula.api import ols
mod = ols('no_of_adults ~ h1n1_vaccine', data = df_vac).fit()
aov_table = sm.stats.anova_lm(mod)
print(aov_table)
#0.55 ie p_value is >0.05; Ho accepted; Bad Predictor
```



```

#_____ 32 no_of_children , ordered
df_vac.no_of_children.isnull().sum() #No Missing values
df_vac.no_of_children.value_counts()
'''
0.0    13697
1.0     2402
2.0     2207
3.0     1336'''
# Bar Plot
sns.countplot(x = 'no_of_children', data = df_vac , palette = 'Dark2')
plt.title('Countplot of no_of_children')
#Hypothesis Testing
import statsmodels.api as sm
from statsmodels.formula.api import ols
mod = ols('no_of_children ~ h1n1_vaccine', data = df_vac).fit()
aov_table = sm.stats.anova_lm(mod)
print(aov_table)
#0.63 ie p_value is >0.05; Ho accepted; Bad Predictor

```



We are almost near to our final data!

```
#+++++  
df_vac.info()  
'''  
lets delete  
index 19, sick_from_seas_vacc  
index 28, census_msa  
index 29, no_of_adults  
index 30, no_of_children  
  
AND SAVE NEW DATA AS hn and export to wd and  
THEN START A NEW SCRIPT  
'''  
hn = df_vac.drop(['sick_from_seas_vacc', 'census_msa', 'no_of_adults', 'no_of_children'], axis = 1)  
hn.info() # 19642, 28 columns  
  
hn.to_csv('hn.csv')
```



```
#_____ Lets create dummy variables for 'race'

df2 = pd.get_dummies(hn.race, drop_first = True, prefix = 'race')

hnd = pd.concat([hn, df2], axis = 1)

# we must remove the original col 'race'

hnc = hnd.drop(['race'], axis = 1)
hnc.info() #19642, 30 columns

hnc.to_csv('hnc.csv')
..
```

hnc - Excel

Sign in

File Home Insert Page Layout Formulas Data Review View Developer Help Tell me what you want to do Share

Paste Cut Copy Format Painter Clipboard

Calibri 11 A A Wrap Text Merge & Center Font

General Number Styles Cells Editing

Conditional Formatting Format as Table Cell Styles Insert Delete Format AutoSum Fill Clear Sort & Find & Filter Select

A1

	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE
1	reduced_o	avoid_tou	dr_recc_h	dr_recc_s	chronic_m	cont_child	is_health	is_h1n1_v	is_h1n1_r	sick_from	is_seas_v	is_seas_r	age_brack	qualificatio	sex	income_le	marital_st	housing_st	employe	h1n1_vacc	race_1	race_2	race_3
2	1	1	0	0	0	0	0	3	1	2	2	1	4	1	0	1	1	0	2	0	0	0	1
3	1	1	0	0	0	0	0	5	4	4	4	2	2	2	1	1	1	1	3	0	0	0	1
4	0	0	0	1	1	0	0	3	3	5	5	4	5	2	0	1	1	1	2	0	0	0	1
5	0	1	0	0	0	0	0	3	3	2	3	1	3	3	0	2	0	0	3	0	0	0	1
6	0	1	0	1	0	0	0	5	2	1	5	4	5	2	1	2	0	0	3	0	0	0	1
7	0	0	0	0	0	0	0	4	1	1	4	2	4	1	1	2	1	0	3	0	0	0	1
8	0	1	1	0	1	0	0	5	2	1	4	2	3	3	0	2	0	0	3	1	0	0	1
9	1	1	0	0	0	0	0	4	1	1	4	2	3	4	1	3	0	0	3	0	0	0	1
10	0	1	0	0	1	0	0	4	2	2	4	2	4	2	1	2	1	0	2	0	0	0	1
11	0	0	0	0	1	1	0	4	1	2	5	4	3	2	1	2	0	1	3	1	0	0	1
12	0	0	0	0	0	0	0	3	2	2	4	2	4	4	1	3	0	0	3	1	0	0	1
13	0	1	0	1	0	0	0	3	2	1	5	4	5	4	1	2	0	0	2	0	0	0	1
14	0	1	0	0	1	1	0	3	1	4	4	2	2	3	1	3	0	0	3	0	0	0	1
15	1	1	0	0	0	0	0	5	4	2	4	4	4	2	1	2	0	0	2	0	0	0	1
16	0	0	0	1	0	0	0	4	2	2	2	1	2	4	0	3	0	0	3	0	0	0	1
17	0	1	1	1	0	0	0	2	2	4	4	2	2	3	0	2	1	0	3	0	0	0	1
18	0	1	1	1	0	0	0	5	2	2	5	4	5	4	1	3	0	0	3	0	0	0	1
19	1	1	0	0	0	0	0	4	2	1	5	2	4	3	0	2	0	0	2	0	0	0	0
20	1	1	1	1	0	0	0	4	2	2	4	2	4	4	0	2	1	1	2	0	0	0	0
21	0	1	1	0	0	0	0	3	3	1	3	3	4	3	0	3	1	0	3	0	0	0	1
22	0	1	1	1	0	0	0	4	2	2	4	1	3	4	0	3	0	0	3	0	0	0	1
23	1	1	0	0	0	0	0	5	5	4	4	4	1	1	0	1	1	1	2	0	0	0	0
24	0	1	0	0	0	0	0	3	2	2	2	4	3	3	0	2	0	1	3	0	0	0	1
25	1	1	0	0	0	0	0	4	2	2	2	2	5	3	0	2	0	0	2	0	0	0	1
26	0	0	0	0	0	0	0	1	1	1	2	1	4	2	1	2	0	0	2	0	0	0	1
27	1	1	1	1	0	0	0	2	1	2	4	2	2	4	0	2	1	0	3	0	0	0	1
28	0	1	0	0	0	0	0	4	2	2	4	2	2	3	1	2	0	0	1	0	0	0	0
29	0	0	1	1	0	0	0	5	4	4	5	2	5	2	0	2	1	0	2	1	0	0	1

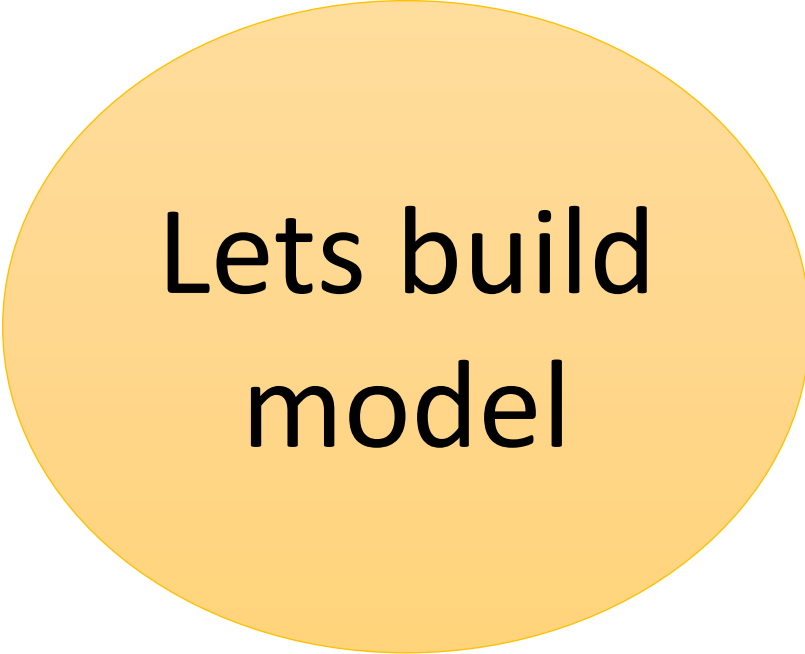
hnc

Ready Accessibility: Unavailable

02:34 12-12-2022

Part 1 is Done!
Congratulations!!





Lets build
model

Libraries

```
# Jesus is my Saviour!
import os
os.chdir('C:\\Users\\Dr Vinod\\Desktop\\WD_python')
import pandas as pd
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from statsmodels.formula.api import ols
from scipy.stats import chi2_contingency
from sklearn.preprocessing import LabelEncoder
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import roc_curve, auc, roc_auc_score
from sklearn.utils import resample
```

```
In [2]: df = pd.read_csv('hnc.csv') #19642; 31 , 1st is unique id
```

```
In [3]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 19642 entries, 0 to 19641
```

```
Data columns (total 31 columns):
```

#	Column	Non-Null Count	Dtype
0	Unnamed: 0	19642 non-null	int64
1	h1n1_worry	19642 non-null	float64
2	h1n1_awareness	19642 non-null	float64
3	antiviral_medication	19642 non-null	float64
4	contact_avoidance	19642 non-null	float64
5	bought_face_mask	19642 non-null	float64
6	wash_hands_frequently	19642 non-null	float64
7	avoid_large_gatherings	19642 non-null	float64
8	reduced_outside_home_cont	19642 non-null	float64
9	avoid_touch_face	19642 non-null	float64
10	dr_recc_h1n1_vacc	19642 non-null	float64
11	dr_recc_seasonal_vacc	19642 non-null	float64
12	chronic_medical_condition	19642 non-null	float64
13	cont_child_undr_6_mnths	19642 non-null	float64
14	is_health_worker	19642 non-null	float64
15	is_h1n1_vacc_effective	19642 non-null	float64
16	is_h1n1_risky	19642 non-null	float64
17	sick_from_h1n1_vacc	19642 non-null	float64
18	is_seas_vacc_effective	19642 non-null	float64
19	is_seas_risky	19642 non-null	float64
20	age_bracket	19642 non-null	int64
21	qualification	19642 non-null	int64
22	sex	19642 non-null	int64
23	income_level	19642 non-null	int64
24	marital_status	19642 non-null	int64
25	housing_status	19642 non-null	int64
26	employment	19642 non-null	int64
27	h1n1_vaccine	19642 non-null	int64
28	race_1	19642 non-null	int64
29	race_2	19642 non-null	int64
30	race_3	19642 non-null	int64

```
dtypes: float64(19), int64(12)
```

```
memory usage: 4.6 MB
```

```

#                      VIF
from statsmodels.stats.outliers_influence import variance_inflation_factor

# first put your predictors in x
x = df.iloc[:, [1,2,15,16,17,18,19,20,21,23,26]] # x is a data frame
# VIF dataframe
vif_data = pd.DataFrame()
vif_data["feature"] = x.columns

# calculating VIF for each feature
vif_data["VIF"] = [variance_inflation_factor(x.values, i)
                   for i in range(len(x.columns))]

print(vif_data) # two columns, feature & VIF will appear
'''
   feature          VIF
0          h1n1_worry    5.759713
1          h1n1_awareness  6.628330
2  is_h1n1_vacc_effective 21.390106
3          is_h1n1_risky    7.203521
4    sick_from_h1n1_vacc    4.689096
5  is_seas_vacc_effective 20.908152
6          is_seas_risky    8.017402
7          age_bracket    5.406338
8      qualification    13.411080
9          income_level    16.792285
10         employment    15.695942
'''

```

Drop vif>10 and 'Unnamed: 0'

```
# drop VIF> 10
df = df.drop(['is_h1n1_vacc_effective', 'is_seas_vacc_effective',
             'qualification', 'employment'], axis = 1)
df.info() # 19642, 27 [with 1st as Unnamed: 0, Lets remove this]
df = df.drop(['Unnamed: 0'], axis = 1)
df.info() # 19642, 26

# X and y
X = df.loc[:, df.columns != 'h1n1_vaccine']
y = df.loc[:, df.columns == 'h1n1_vaccine']
```


Model

```
from sklearn.linear_model import LogisticRegression
model1 = LogisticRegression(solver='liblinear', random_state=0)
model1.fit(X, y)
model1.intercept_
model1.coef_
```

```
In [26]: model1.intercept_
Out[26]: array([-4.77922475])
```

```
In [27]: model1.coef_
Out[27]:
array([[ -0.04498622,  0.23585606,  0.08002789, -0.02169607,  0.11127946,
         0.06433461, -0.18866288, -0.09143327,  0.0279202 ,  2.01272438,
        -0.55614951,  0.13015874,  0.23410973,  0.86014732,  0.45516232,
        -0.06153378,  0.21774852,  0.15974545,  0.20150469,  0.09885106,
        -0.12242907, -0.01306603,  0.13690518,  0.37603554,  0.35276398]])
```

#Predictions

```
y_pred = model1.predict(X)
```

#Confusion matrix

```
from sklearn import metrics
```

```
cm = metrics.confusion_matrix(y, y_pred)
```

```
print(cm)
```

```
'''
```

```
[[14200  928]
```

```
 [ 2608 1906]]
```

```
'''
```

y_pred - NumPy object array

	0
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0

Format

Resize

☒ Background color

Save and Close

Close

#Predictions

```
y_pred = model1.predict(X)
```

#Confusion matrix

```
from sklearn import metrics
```

```
cm = metrics.confusion_matrix(y, y_pred)
```

```
print(cm)
```

```
'''
```

```
[[14200   928]
```

```
 [ 2608  1906]]
```

```
'''
```

#Accuracy Score - correct predictions / total number of data points

```
model1.score(X,y) #.0.82
```

```
(14200+1906)/(14200+928+2608+1906) # 0.82
```

```
In [32]: model1.score(X,y) #.0.82
```

```
Out[32]: 0.8199775990225028
```

```
In [33]: (14200+1906)/(14200+928+2608+1906) # 0.82
```

```
Out[33]: 0.8199775990225028
```

```
#Classification report  
from sklearn.metrics import classification_report  
print(classification_report(y, y_pred))
```

```
In [34]: from sklearn.metrics import classification_report
```

```
In [35]: print(classification_report(y, y_pred))
```

	precision	recall	f1-score	support
0	0.84	0.94	0.89	15128
1	0.67	0.42	0.52	4514
accuracy			0.82	19642
macro avg	0.76	0.68	0.70	19642
weighted avg	0.81	0.82	0.80	19642

```

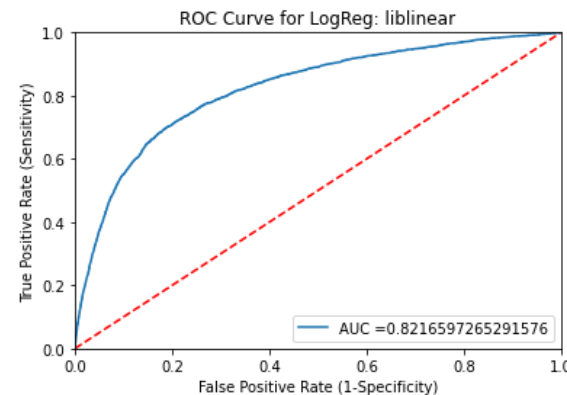
#ROC Curve - Receiver Operating Characteristic curve
#tpr = True Positive Rate
#fpr = False Positive Rate
from sklearn.metrics import roc_curve, auc, roc_auc_score
y_pred_prob = model1.predict_proba(X)
fpr, tpr, thresholds = roc_curve(df["h1n1_vaccine"], y_pred_prob[:,1])
roc_auc = auc(fpr, tpr) #Area under Curve 0.82
print(roc_auc)

```

```

#ROC Curve
plt.title('ROC Curve for LogReg: liblinear')
plt.xlabel('False Positive Rate (1-Specificity)')
plt.ylabel('True Positive Rate (Sensitivity)')
plt.plot([0, 1], [0, 1], 'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.plot(fpr, tpr, label = 'AUC =' + str(roc_auc))
plt.legend(loc=4) #Location of Label
plt.show()

```



y_pred_prob - NumPy object ...

	0	1
0	0.966744	0.0332555
1	0.834273	0.165727
2	0.892517	0.107483
3	0.905644	0.0943557
4	0.869147	0.130853
5	0.935931	0.0640693
6	0.573655	0.426345
7	0.913557	0.0864425
8	0.89974	0.10026
9	0.880152	0.119848
10	0.821641	0.168359

Format Resize ☒ Background color

Save and Close Close

SMOTE

```
# _____ SMOTE

novac = df[df.h1n1_vaccine == 0] #15128,26
vac = df[df.h1n1_vaccine == 1] #4514, 26
# _____ oversample minority_with replacement
from sklearn.utils import resample
vac_oversample = resample(vac,
                           replace=True, # sample with replacement
                           n_samples=len(novac), # match number in majority class
                           random_state=27) # reproducible results

# combine majority and oversampled minority

dfsmote = pd.concat([novac, vac_oversample])
dfsmote.h1n1_vaccine.value_counts()
'''
1      15128
0      15128
'''
```

```

# _____ Lets re do log reg
# X and y
X2 = dfsmote.loc[:, dfsmote.columns != 'h1n1_vaccine']
y2 = dfsmote.loc[:, dfsmote.columns == 'h1n1_vaccine']
y2.value_counts() # both 15,128

# solver = liblinear
'''
liblinear [library for linear classification]: good for small data
newton-cg [newton conjugate]: can be used in this case
lbfgs [limited memory BFGS]: for multiclass problems
BFGS: Broyden-Fletcher-Goldfarb-Shanno algorithm
sag [Stochastic Average Gradient Descent]: good for large data sets
saga: a little variant of sag
'''

from sklearn.linear_model import LogisticRegression
model2 = LogisticRegression(solver='liblinear', random_state=0)
model2.fit(X2, y2)
model2.intercept_
model2.coef_

```

Coefficients

```
In [54]: model2.intercept_  
Out[54]: array([-3.55103733])
```

```
In [55]: model2.coef_  
Out[55]:  
array([[ -0.01118147,  0.2210547 ,  0.1371775 , -0.00679728,  0.14257456,  
         0.09421023, -0.2000553 , -0.11595893, -0.01936164,  2.10424968,  
        -0.63670885,  0.14104786,  0.19295641,  0.93630368,  0.43867647,  
        -0.05666603,  0.22833308,  0.1551444 ,  0.23000128,  0.10607717,  
        -0.11945158,  0.07376133,  0.04801412,  0.26810969,  0.24540742]])
```


With and Without SMOTE

```
#Confusion matrix
from sklearn import metrics
cm2 = metrics.confusion_matrix(y2, y_pred2)
print(cm2)
'''
```

```
WITH SMOTE=
[[11848  3280]
 [ 4154 10974]]
```

```
WITHOUT SMOTE =
[[14200   928]
 [ 2608  1906]]
'''
```

```
#Accuracy Score - correct predictions / total number of data points
model2.score(X2,y2) #WITH SMOTE = 0.75; without = 0.82
(11848+10974)/(11848+3280+4154+10974) # 0.75
```

```
#Classification report
from sklearn.metrics import classification_report
print(classification_report(y2, y_pred2))
```

```
In [61]: from sklearn.metrics import classification_report
```

```
In [62]: print(classification_report(y2, y_pred2))
```

	precision	recall	f1-score	support
0	0.74	0.78	0.76	15128
1	0.77	0.73	0.75	15128
accuracy			0.75	30256
macro avg	0.76	0.75	0.75	30256
weighted avg	0.76	0.75	0.75	30256

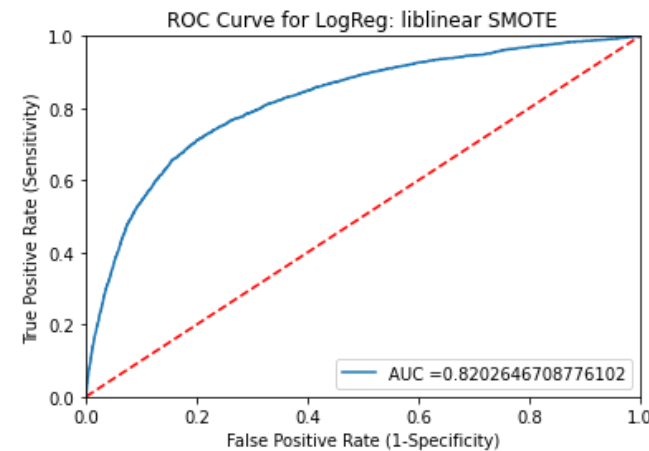
AUC

```
#ROC Curve - Receiver Operating Characteristic curve
#tpr = True Positive Rate
#fpr = False Positive Rate
from sklearn.metrics import roc_curve, auc, roc_auc_score
y_pred_prob2 = model2.predict_proba(X2)
fpr2, tpr2, thresholds2 =roc_curve(dfsmote["h1n1_vaccine"], y_pred_prob2[:,1])
roc_auc2 = auc(fpr2, tpr2) #Area under Curve 0.82
print(roc_auc2) # 0.82, same as without smote
```

```
In [63]: from sklearn.metrics import roc_curve, auc, roc_auc_score
...: y_pred_prob2 = model2.predict_proba(X2)
...: fpr2, tpr2, thresholds2 =roc_curve(dfsmote["h1n1_vaccine"], y_pred_prob2[:,1])
...: roc_auc2 = auc(fpr2, tpr2) #Area under Curve 0.82
...: print(roc_auc2) # 0.82, same as without smote
0.8202646708776102
```

ROC

```
#ROC Curve  
plt.title('ROC Curve for LogReg: liblinear SMOTE')  
plt.xlabel('False Positive Rate (1-Specificity)')  
plt.ylabel('True Positive Rate (Sensitivity)')  
plt.plot([0, 1], [0, 1], 'r--')  
plt.xlim([0, 1])  
plt.ylim([0, 1])  
plt.plot(fpr2, tpr2, label = 'AUC =' +str(roc_auc2))  
plt.legend(loc=4) #Location of label  
plt.show()
```



Awesome so far!

