This file is used to do the data analysis

In [1]:

```
# import the package
from sklearn.model_selection import train_test_split, cross_val_score, KFold, GridSearchCV
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.tree import export_graphviz
from sklearn import utils
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.preprocessing import LabelEncoder
```

In [2]:

```
# load the data
path = "./data/"
train_data = pd.read_csv(path + "Flu_Shot_Learning_Predict_H1N1_and_Seasonal_Flu_Vaccines_-_Training_Features.csv")
train_label = pd.read_csv(path + "Flu_Shot_Learning_Predict_H1N1_and_Seasonal_Flu_Vaccines_-_Training_Labels.csv")
test_data = pd.read_csv(path + "Flu_Shot_Learning_Predict_H1N1_and_Seasonal_Flu_Vaccines_-_Test_Features.csv")

# check if the data has the null value
print(train_data.isnull().any())
print(train_label.isnull().any())
print(test_data.isnull().any())

# check data type
print(train_data.dtypes)
```

```
respondent_id                False
h1n1_concern                  True
h1n1_knowledge                True
behavioral_antiviral_meds     True
behavioral_avoidance          True
behavioral_face_mask          True
behavioral_wash_hands         True
behavioral_large_gatherings   True
behavioral_outside_home       True
behavioral_touch_face         True
doctor_recc_h1n1              True
doctor_recc_seasonal          True
chronic_med_condition         True
child_under_6_months          True
health_worker                 True
health_insurance              True
opinion_h1n1_vacc_effective   True
opinion_h1n1_risk             True
opinion_h1n1_sick_from_vacc   True
opinion_seas_vacc_effective   True
opinion_seas_risk             True
opinion_seas_sick_from_vacc   True
age_group                    False
education                     True
race                         False
sex                          False
income_poverty                True
marital_status                True
rent_or_own                   True
employment_status             True
hhs_geo_region               False
census_msa                   False
household_adults              True
household_children            True
employment_industry           True
employment_occupation         True
dtype: bool
respondent_id       False
h1n1_vaccine        False
seasonal_vaccine    False
dtype: bool
respondent_id                False
h1n1_concern                  True
h1n1_knowledge                True
behavioral_antiviral_meds     True
behavioral_avoidance          True
behavioral_face_mask          True
behavioral_wash_hands         True
behavioral_large_gatherings   True
behavioral_outside_home       True
behavioral_touch_face         True
doctor_recc_h1n1              True
doctor_recc_seasonal          True
chronic_med_condition         True
child_under_6_months          True
health_worker                 True
health_insurance              True
opinion_h1n1_vacc_effective   True
opinion_h1n1_risk             True
opinion_h1n1_sick_from_vacc   True
opinion_seas_vacc_effective   True
```

```
opinion_seas_risk              True
opinion_seas_sick_from_vacc    True
age_group                     False
education                      True
race                          False
sex                           False
income_poverty                 True
marital_status                 True
rent_or_own                    True
employment_status              True
hhs_geo_region                False
census_msa                    False
household_adults               True
household_children             True
employment_industry            True
employment_occupation          True
dtype: bool
respondent_id                  int64
h1n1_concern                 float64
h1n1_knowledge               float64
behavioral_antiviral_meds    float64
behavioral_avoidance         float64
behavioral_face_mask         float64
behavioral_wash_hands        float64
behavioral_large_gatherings  float64
behavioral_outside_home      float64
behavioral_touch_face        float64
doctor_recc_h1n1             float64
doctor_recc_seasonal         float64
chronic_med_condition        float64
child_under_6_months         float64
health_worker                float64
health_insurance             float64
opinion_h1n1_vacc_effective  float64
opinion_h1n1_risk            float64
opinion_h1n1_sick_from_vacc  float64
opinion_seas_vacc_effective  float64
opinion_seas_risk            float64
opinion_seas_sick_from_vacc  float64
age_group                     object
education                     object
race                          object
sex                           object
income_poverty                object
marital_status                object
rent_or_own                   object
employment_status             object
hhs_geo_region                object
census_msa                    object
household_adults             float64
household_children           float64
employment_industry           object
employment_occupation         object
dtype: object
```

In [3]:

```
# check the data
pd.set_option('display.max_columns', None)
train_data[0:5]
```

Out[3]:

	respondent_id	h1n1_concern	h1n1_knowledge	behavioral_antiviral_meds	behavioral_avoid
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	

In [4]:

```
# data preprocess
def data_preprocess(data:pd.DataFrame):
    # remove the null row
    data.dropna(axis=0, how='any', inplace=True)

    # remove the useless column
    data.drop(['employment_status'], axis = 1, inplace=True)

    # object encoder
    categorical_columns = [col for col in data.columns.values if data[col].dtype == 'object']
    for feat in categorical_columns:
        print(feat)
        lbe = LabelEncoder()
        data[feat] = lbe.fit_transform(data[feat])

    # data normalization
    # numeric_features = [col for col in data.columns.values if (data[col].dtype != 'object' and
col != 'respondent_id')][:-1]
    # data[numeric_features] = data[numeric_features].apply(lambda x: (x - x.mean()) / (x.std
()))

    return data

# data preprocess
train_data = data_preprocess(train_data)
```

```
age_group
education
race
sex
income_poverty
marital_status
rent_or_own
hhs_geo_region
census_msa
employment_industry
employment_occupation
```

In [5]:

```
# check the data
print(train_data.isnull().any())
print(train_data.dtypes)
train_data[0:5]
```

```
respondent_id                False
h1n1_concern                 False
h1n1_knowledge               False
behavioral_antiviral_meds    False
behavioral_avoidance         False
behavioral_face_mask         False
behavioral_wash_hands        False
behavioral_large_gatherings  False
behavioral_outside_home      False
behavioral_touch_face        False
doctor_recc_h1n1             False
doctor_recc_seasonal         False
chronic_med_condition        False
child_under_6_months         False
health_worker                False
health_insurance             False
opinion_h1n1_vacc_effective  False
opinion_h1n1_risk            False
opinion_h1n1_sick_from_vacc  False
opinion_seas_vacc_effective  False
opinion_seas_risk            False
opinion_seas_sick_from_vacc  False
age_group                    False
education                    False
race                         False
sex                          False
income_poverty               False
marital_status               False
rent_or_own                  False
hhs_geo_region               False
census_msa                   False
household_adults             False
household_children           False
employment_industry          False
employment_occupation        False
dtype: bool
respondent_id                  int64
h1n1_concern                 float64
h1n1_knowledge               float64
behavioral_antiviral_meds    float64
behavioral_avoidance         float64
behavioral_face_mask         float64
behavioral_wash_hands        float64
behavioral_large_gatherings  float64
behavioral_outside_home      float64
behavioral_touch_face        float64
doctor_recc_h1n1             float64
doctor_recc_seasonal         float64
chronic_med_condition        float64
child_under_6_months         float64
health_worker                float64
health_insurance             float64
opinion_h1n1_vacc_effective  float64
opinion_h1n1_risk            float64
opinion_h1n1_sick_from_vacc  float64
opinion_seas_vacc_effective  float64
opinion_seas_risk            float64
opinion_seas_sick_from_vacc  float64
age_group                      int32
education                      int32
race                           int32
```

```
sex                          int32
income_poverty               int32
marital_status               int32
rent_or_own                  int32
hhs_geo_region               int32
census_msa                   int32
household_adults           float64
household_children         float64
employment_industry          int32
employment_occupation        int32
dtype: object
```

Out[5]:

	respondent_id	h1n1_concern	h1n1_knowledge	behavioral_antiviral_meds	behavioral_avoi
1	1	3.0	2.0	0.0	
7	7	1.0	0.0	0.0	
10	10	2.0	1.0	0.0	
11	11	1.0	2.0	0.0	
15	15	1.0	1.0	0.0	

In [6]:

```
# align the label
train_label = train_label[train_label['respondent_id'].isin(train_data['respondent_id'])]
train_label[0:5]
```

Out[6]:

	respondent_id	h1n1_vaccine	seasonal_vaccine
1	1	0	1
7	7	1	1
10	10	1	1
11	11	1	1
15	15	0	0

In [7]:

```
# combine the data
combine_data = pd.merge(train_data, train_label, on='respondent_id')

# shuffle the data
combine_data = utils.shuffle(combine_data)
print(f"data length is {len(combine_data)}")
combine_data[0:5]
```

data length is 6437

Out[7]:

	respondent_id	h1n1_concern	h1n1_knowledge	behavioral_antiviral_meds	behavioral_av
4923	20490	1.0	2.0	0.0	
227	995	1.0	2.0	0.0	
4586	19166	0.0	2.0	0.0	
2667	11229	2.0	2.0	0.0	
3573	15022	1.0	2.0	0.0	

In [8]:

```
# set lable and train data
label_h1n1 = combine_data['h1n1_vaccine'].values
label_seasonal = combine_data['seasonal_vaccine'].values
train_data = combine_data.drop(['h1n1_vaccine', 'seasonal_vaccine', 'respondent_id'], axis = 1).
values

train_data[0:5]
```

Out[8]:

```
array([[ 1.,  2.,  0.,  0.,  0.,  0.,  0.,  0.,  1.,  1.,  1.,  1.,  0.,
         0.,  1.,  5.,  1.,  2.,  5.,  1.,  2.,  2.,  2.,  3.,  0.,  0.,
         1.,  0.,  0.,  0.,  0.,  0.,  2., 16.],
       [ 1.,  2.,  0.,  1.,  0.,  1.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,
         0.,  1.,  4.,  4.,  2.,  5.,  4.,  2.,  2.,  2.,  3.,  1.,  1.,
         1.,  1.,  1.,  2.,  1.,  0.,  0.,  4.],
       [ 0.,  2.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.,
         0.,  1.,  5.,  1.,  1.,  4.,  2.,  2.,  2.,  0.,  3.,  1.,  0.,
         1.,  1.,  8.,  0.,  0.,  0.,  1., 20.],
       [ 2.,  2.,  0.,  1.,  0.,  1.,  0.,  1.,  1.,  1.,  1.,  0.,  0.,
         1.,  1.,  5.,  1.,  1.,  5.,  1.,  1.,  3.,  2.,  3.,  0.,  0.,
         0.,  0.,  8.,  0.,  1.,  0.,  4.,  2.],
       [ 1.,  2.,  0.,  1.,  0.,  1.,  1.,  1.,  1.,  0.,  0.,  0.,  0.,
         0.,  1.,  4.,  1.,  2.,  4.,  2.,  1.,  2.,  3.,  3.,  1.,  1.,
         0.,  0.,  8.,  1.,  1.,  0.,  3., 19.]])
```

In [9]:

```
# data standardization
scaler = StandardScaler()
scaler.fit(train_data)
train_data = scaler.transform(train_data)
```

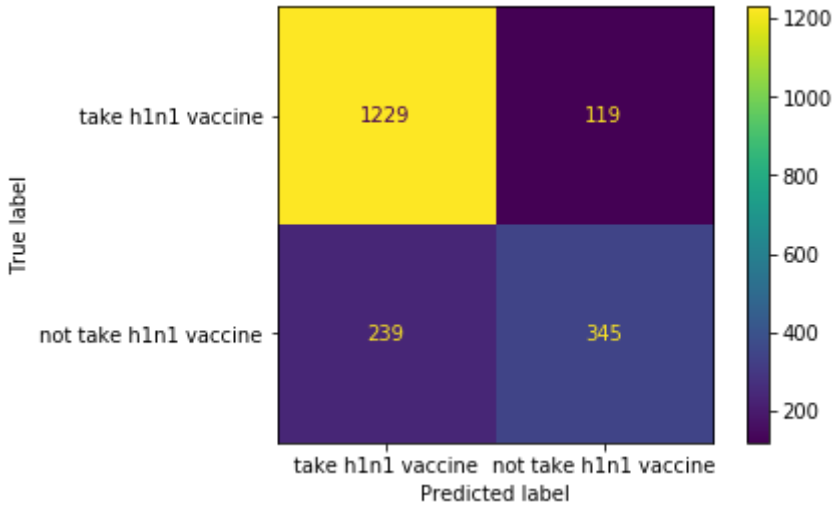
In [10]:

```
# Split training and test sets
train_X_h1n1, test_X_h1n1, train_Y_h1n1, test_Y_h1n1 = train_test_split(train_data, label_h1n1,
test_size=0.3, random_state=0)
train_X_seasonal, test_X_seasonal, train_Y_seasonal, test_Y_seasonal = train_test_split(train_da
ta, label_seasonal, test_size=0.3, random_state=0)

# RandomForest
clf_h1n1 = RandomForestClassifier(n_estimators=10, criterion="entropy", min_samples_leaf=3, max_
depth=10)
clf_seasonal = RandomForestClassifier(n_estimators=10, criterion="entropy", min_samples_leaf=3,
max_depth=10)
```

In [11]:

```
# train h1n1
clf_h1n1.fit(train_X_h1n1, train_Y_h1n1)
predict_Y_h1n1 = clf_h1n1.predict(test_X_h1n1)

print(f"accuracy_score of h1n1 = {accuracy_score(y_pred=predict_Y_h1n1, y_true=test_Y_h1n1)}")

C_h1n1 = confusion_matrix(test_Y_h1n1, predict_Y_h1n1, labels=None, sample_weight=None)
print(f'confusion matrix of h1n1: {C_h1n1}')

class_h1n1 = ['take h1n1 vaccine', 'not take h1n1 vaccine']

disp = ConfusionMatrixDisplay(confusion_matrix=C_h1n1, display_labels=class_h1n1)
disp_fig = disp.plot(include_values=True,cmap="viridis",ax=None,xticks_rotation="horizontal",va
lues_format="d")

plt.savefig('./confusion_matrix_h1n1.png', dpi=300, bbox_inches='tight')
```

```
accuracy_score of h1n1 = 0.8146997929606625
confusion matrix of h1n1: [[1229  119]
 [ 239  345]]
```



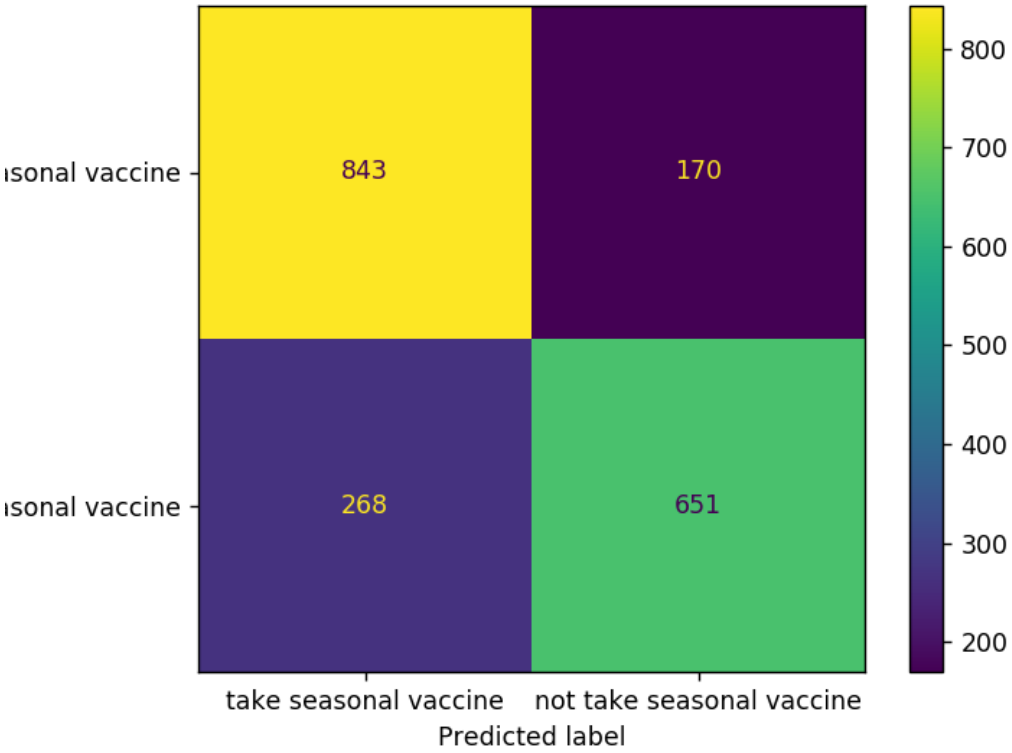
In [12]:

```
%matplotlib notebook
```

In [13]:

```
# train seasonal
clf_seasonal.fit(train_X_seasonal, train_Y_seasonal)
predict_Y_seasonal = clf_seasonal.predict(test_X_seasonal)

print(f"accuracy_score of seasonal = {accuracy_score(y_pred=predict_Y_seasonal, y_true=test_Y_seasonal)}")

C_seasonal = confusion_matrix(test_Y_seasonal, predict_Y_seasonal, labels=None, sample_weight=None)
print(f'confusion matrix of seasonal: {C_seasonal}')

class_seasonal = ['take seasonal vaccine', 'not take seasonal vaccine']

disp = ConfusionMatrixDisplay(confusion_matrix=C_seasonal, display_labels=class_seasonal)
disp_fig = disp.plot(include_values=True,cmap="viridis",ax=None,xticks_rotation="horizontal",values_format="d")

plt.savefig('./confusion_matrix_seasonal.png', dpi=300, bbox_inches='tight')
```

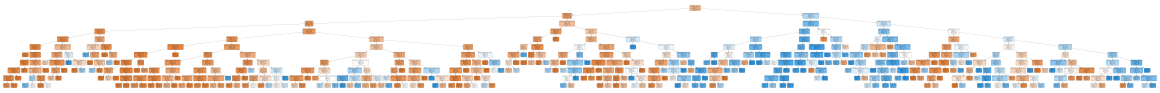
```
accuracy_score of seasonal = 0.7732919254658385
confusion matrix of seasonal: [[843 170]
 [268 651]]
```



In [14]:

```
# plot RF of the h1n1
from sklearn.tree import export_graphviz

estimator = clf_h1n1.estimators_[5]

feature_names_h1n1 = list(combine_data.drop(['h1n1_vaccine', 'seasonal_vaccine', 'respondent_i
d'], axis = 1).columns)

export_graphviz(estimator, out_file='h1n1tree.dot',
                feature_names = feature_names_h1n1, class_names = 'h1n1_vaccine', rounded = Tru
e, proportion = False, precision = 2, filled = True)

# Convert to png using system command (requires Graphviz)

from subprocess import call

call(['dot', '-Tpng', './h1n1tree.dot', '-o', './h1n1tree.png', '-Gdpi=600'])

# Display in jupyter notebook

from IPython.display import Image

Image(filename = 'h1n1tree.png')
```

Out[14]:



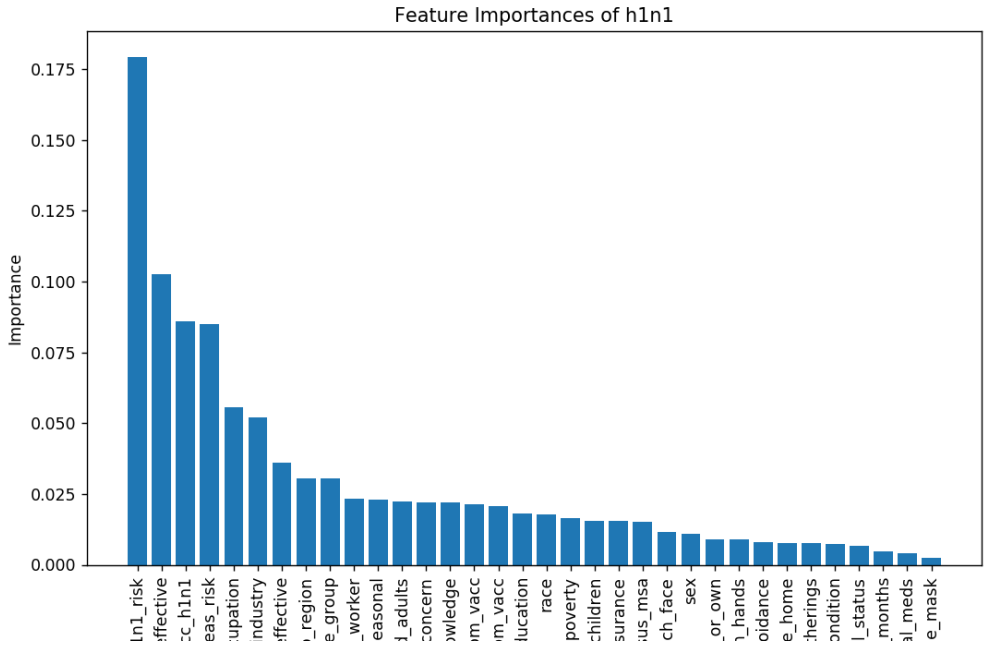
In [15]:

```
# check the feature importance
import numpy as np
importances_h1n1 = clf_h1n1.feature_importances_
indices = np.argsort(importances_h1n1)[::-1]

for f in range(train_X_h1n1.shape[1]):
    print("%2d) %-*s %f" % (f + 1, 30, feature_names_h1n1[indices[f]], importances_h1n1[indices
[f]]))
```

```
 1) opinion_h1n1_risk              0.179358
 2) opinion_h1n1_vacc_effective    0.102577
 3) doctor_recc_h1n1               0.086049
 4) opinion_seas_risk              0.084860
 5) employment_occupation          0.055695
 6) employment_industry            0.051884
 7) opinion_seas_vacc_effective    0.035981
 8) hhs_geo_region                 0.030636
 9) age_group                      0.030547
10) health_worker                  0.023168
11) doctor_recc_seasonal           0.023072
12) household_adults               0.022180
13) h1n1_concern                   0.022091
14) h1n1_knowledge                 0.022010
15) opinion_seas_sick_from_vacc    0.021207
16) opinion_h1n1_sick_from_vacc    0.020847
17) education                      0.018247
18) race                           0.017924
19) income_poverty                 0.016563
20) household_children             0.015603
21) health_insurance               0.015410
22) census_msa                     0.015154
23) behavioral_touch_face          0.011552
24) sex                            0.010885
25) rent_or_own                    0.008962
26) behavioral_wash_hands          0.008930
27) behavioral_avoidance           0.008030
28) behavioral_outside_home        0.007650
29) behavioral_large_gatherings    0.007604
30) chronic_med_condition          0.007294
31) marital_status                 0.006608
32) child_under_6_months           0.004873
33) behavioral_antiviral_meds      0.003994
34) behavioral_face_mask           0.002554
```

In [16]:

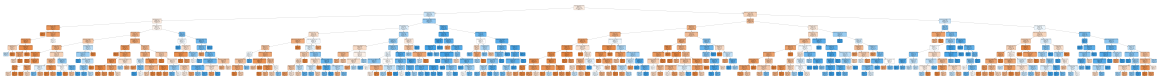
```
import matplotlib.pyplot as plt

# Set the figure size
plt.figure(figsize=(10, 6))

# Plot the feature importances as a bar chart
plt.bar(range(train_X_h1n1.shape[1]), importances_h1n1[indices])

# Add x-axis ticks and labels
plt.xticks(range(train_X_h1n1.shape[1]), [feature_names_h1n1[i] for i in indices], rotation=90)

# Set the x-axis label
plt.xlabel("Feature")

# Set the y-axis label
plt.ylabel("Importance")

# Set the title
plt.title("Feature Importances of h1n1")

plt.savefig('./bar_h1n1.png', dpi=300, bbox_inches='tight')
# Show the plot
plt.show()
```



In [17]:

```
# plot RF of the seasonal
from sklearn.tree import export_graphviz

estimator = clf_seasonal.estimators_[5]

feature_names_seasonal = (list(combine_data.drop(['h1n1_vaccine', 'seasonal_vaccine', 'respondent_id'], axis = 1).columns))
export_graphviz(estimator, out_file='seasonaltree.dot',
                feature_names = feature_names_seasonal, class_names = 'seasonal_vaccine', rounded = True, proportion = False, precision = 2, filled = True)

# Convert to png using system command (requires Graphviz)

from subprocess import call

call(['dot', '-Tpng', './seasonaltree.dot', '-o', './seasonaltree.png', '-Gdpi=600'])

# Display in jupyter notebook

from IPython.display import Image

Image(filename = 'seasonaltree.png')
```

Out[17]:



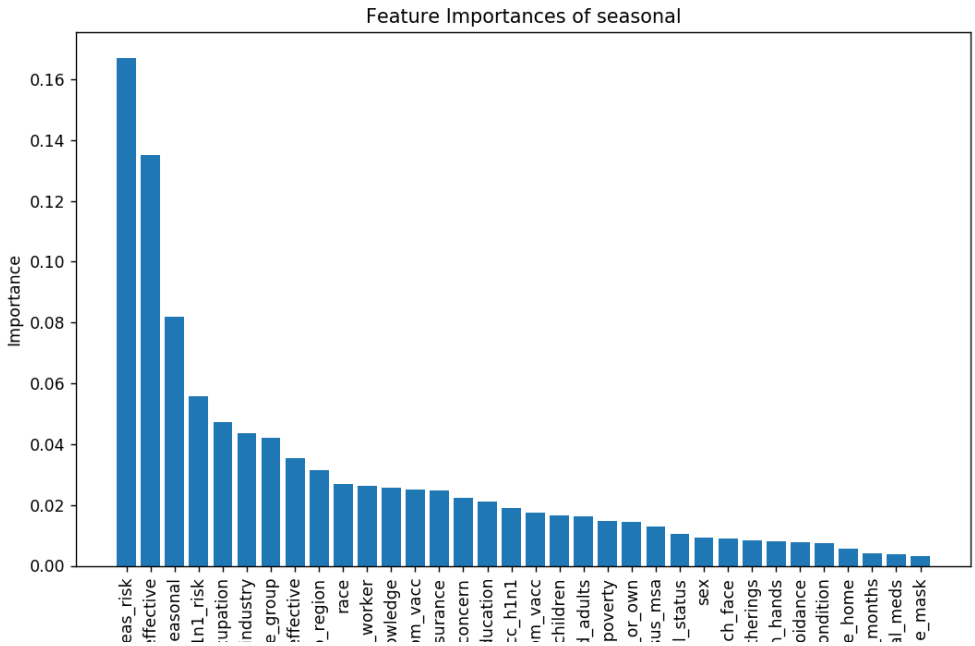
In [18]:

```
# check the feature importance
importances_seasonal = clf_seasonal.feature_importances_
indices = np.argsort(importances_seasonal)[::-1]

for f in range(train_X_seasonal.shape[1]):
    print("%2d) %-*s %f" % (f + 1, 30, feature_names_seasonal[indices[f]], importances_seasonal
[indices[f]]))
```

```
 1) opinion_seas_risk              0.167171
 2) opinion_seas_vacc_effective    0.135201
 3) doctor_recc_seasonal           0.081876
 4) opinion_h1n1_risk              0.055853
 5) employment_occupation          0.047123
 6) employment_industry            0.043657
 7) age_group                      0.042106
 8) opinion_h1n1_vacc_effective    0.035451
 9) hhs_geo_region                 0.031326
10) race                           0.026933
11) health_worker                  0.026262
12) h1n1_knowledge                 0.025760
13) opinion_seas_sick_from_vacc    0.025031
14) health_insurance               0.024651
15) h1n1_concern                   0.022245
16) education                      0.021019
17) doctor_recc_h1n1               0.018887
18) opinion_h1n1_sick_from_vacc    0.017447
19) household_children             0.016542
20) household_adults               0.016354
21) income_poverty                 0.014833
22) rent_or_own                    0.014324
23) census_msa                     0.012832
24) marital_status                 0.010380
25) sex                            0.009215
26) behavioral_touch_face          0.009086
27) behavioral_large_gatherings    0.008275
28) behavioral_wash_hands          0.008107
29) behavioral_avoidance           0.007895
30) chronic_med_condition          0.007497
31) behavioral_outside_home        0.005603
32) child_under_6_months           0.004096
33) behavioral_antiviral_meds      0.003886
34) behavioral_face_mask           0.003075
```

In [19]:

```
import matplotlib.pyplot as plt

# Set the figure size
plt.figure(figsize=(10, 6))

# Plot the feature importances as a bar chart
plt.bar(range(train_X_seasonal.shape[1]), importances_seasonal[indices])

# Add x-axis ticks and labels
plt.xticks(range(train_X_seasonal.shape[1]), [feature_names_seasonal[i] for i in indices], rota
tion=90)

# Set the x-axis label
plt.xlabel("Feature")

# Set the y-axis label
plt.ylabel("Importance")

# Set the title
plt.title("Feature Importances of seasonal")

plt.savefig('./bar_seasonal.png', dpi=300, bbox_inches='tight')
# Show the plot
plt.show()
```



In [ ]:

In [ ]:

In [ ]: