In [1]:

```
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import KFold, cross_val_score, GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import MinMaxScaler
from matplotlib import pyplot

%matplotlib inline
```

In [2]:

```
data = pd.read_csv("COVID-19BehaviorData_CAN_USA.csv")
```

In [3]:

```
# Data Cleaning
data["i4_health"].replace(["No, they have not","Yes, and they tested negative","Not sur
data['i3_health'].value_counts()
data["i3_health"].replace(["No, I have not","Yes, and I tested negative","Not sure"," "
data["i6_health"].value_counts()
data["i5_health_1"].replace([" ","No","Yes"],["N/A",'0','1'],inplace=True)
data["i5_health_2"].replace([" ","No","Yes"],["N/A",'0','1'],inplace=True)
data["i5_health_3"].replace([" ","No","Yes"],["N/A",'0','1'],inplace=True)
data["i5_health_4"].replace([" ","No","Yes"],["N/A",'0','1'],inplace=True)
data["i5_health_5"].replace([" ","No","Yes"],["N/A",'0','1'],inplace=True)
data["i5_health_99"].replace([" ","No","Yes"],["N/A",'0','1'],inplace=True)
data["i5a_health"].replace([" ","No","Yes","Not sure"],["N/A","2","1","99"],inplace=Tru
data["i8_health"].replace([" ","No","Yes","Not sure"],["N/A",'2','1','99'],inplace=True
data["i6_health"].replace([" ","Always","Frequently","Sometimes","Rarely","Not at all"]
data["i7b_health"].replace([" ","No","Yes"],["N/A",'2','1'],inplace=True)
data["i9_health"].replace([" ","No","Yes","Not sure"],['0','2','1','99'],inplace=True)
data["i10_health"].replace([" ","Very easy","Somewhat easy","Somewhat difficult","Neith
data["i11_health"].replace([" ","Very willing","Somewhat willing","Somewhat unwilling",
data["i12_health_1"].replace([" ","Always","Frequently","Sometimes","Rarely","Not at al
data["i12_health_2"].replace([" ","Always","Frequently","Sometimes","Rarely","Not at al
data["i12_health_3"].replace([" ","Always","Frequently","Sometimes","Rarely","Not at al
data["i12_health_4"].replace([" ","Always","Frequently","Sometimes","Rarely","Not at al
data["i12_health_5"].replace([" ","Always","Frequently","Sometimes","Rarely","Not at al
data["i12_health_6"].replace([" ","Always","Frequently","Sometimes","Rarely","Not at al
data["i12_health_7"].replace([" ","Always","Frequently","Sometimes","Rarely","Not at al
data["i12_health_8"].replace([" ","Always","Frequently","Sometimes","Rarely","Not at al
data["i12_health_9"].replace([" ","Always","Frequently","Sometimes","Rarely","Not at al
data["i12_health_10"].replace([" ","Always","Frequently","Sometimes","Rarely","Not at a
data["i12_health_11"].replace([" ","Always","Frequently","Sometimes","Rarely","Not at a
data["i12_health_12"].replace([" ","Always","Frequently","Sometimes","Rarely","Not at a
data["i12_health_13"].replace([" ","Always","Frequently","Sometimes","Rarely","Not at a
data["i12_health_14"].replace([" ","Always","Frequently","Sometimes","Rarely","Not at a
data["i12_health_15"].replace([" ","Always","Frequently","Sometimes","Rarely","Not at a
data["i12_health_16"].replace([" ","Always","Frequently","Sometimes","Rarely","Not at a
data["i12_health_17"].replace([" ","Always","Frequently","Sometimes","Rarely","Not at a
data["i12_health_18"].replace([" ","Always","Frequently","Sometimes","Rarely","Not at a
data["i12_health_19"].replace([" ","Always","Frequently","Sometimes","Rarely","Not at a
data["i12_health_20"].replace([" ","Always","Frequently","Sometimes","Rarely","Not at a
data["i14_health_1"].replace([" ","No","Yes"],["N/A",'0','1'],inplace=True)
```

```
data["i14_health_2"].replace([" ","No","Yes"],["N/A",'0','1'],inplace=True)
data["i14_health_3"].replace([" ","No","Yes"],["N/A",'0','1'],inplace=True)
data["i14_health_4"].replace([" ","No","Yes"],["N/A",'0','1'],inplace=True)
data["i14_health_5"].replace([" ","No","Yes"],["N/A",'0','1'],inplace=True)
data["i14_health_6"].replace([" ","No","Yes"],["N/A",'0','1'],inplace=True)
data["i14_health_7"].replace([" ","No","Yes"],["N/A",'0','1'],inplace=True)
data["i14_health_8"].replace([" ","No","Yes"],["N/A",'0','1'],inplace=True)
data["i14_health_9"].replace([" ","No","Yes"],["N/A",'0','1'],inplace=True)
data["i14_health_10"].replace([" ","No","Yes"],["N/A",'0','1'],inplace=True)
data["i14_health_96"].replace([" ","No","Yes"],["N/A",'0','1'],inplace=True)
data["i14_health_98"].replace([" ","No","Yes"],["N/A",'0','1'],inplace=True)
data["i14_health_99"].replace([" ","No","Yes"],["N/A",'0','1'],inplace=True)
data["d1_health_1"].replace([" ","No","Yes"],["N/A",'0','1'],inplace=True)
data["d1_health_2"].replace([" ","No","Yes"],["N/A",'0','1'],inplace=True)
data["d1_health_3"].replace([" ","No","Yes"],["N/A",'0','1'],inplace=True)
data["d1_health_4"].replace([" ","No","Yes"],["N/A",'0','1'],inplace=True)
data["d1_health_5"].replace([" ","No","Yes"],["N/A",'0','1'],inplace=True)
data["d1_health_6"].replace([" ","No","Yes"],["N/A",'0','1'],inplace=True)
data["d1_health_7"].replace([" ","No","Yes"],["N/A",'0','1'],inplace=True)
data["d1_health_8"].replace([" ","No","Yes"],["N/A",'0','1'],inplace=True)
data["d1_health_9"].replace([" ","No","Yes"],["N/A",'0','1'],inplace=True)
data["d1_health_10"].replace([" ","No","Yes"],["N/A",'0','1'],inplace=True)
data["d1_health_11"].replace([" ","No","Yes"],["N/A",'0','1'],inplace=True)
data["d1_health_12"].replace([" ","No","Yes"],["N/A",'0','1'],inplace=True)
data["d1_health_13"].replace([" ","No","Yes"],["N/A",'0','1'],inplace=True)
data["d1_health_98"].replace([" ","No","Yes"],["N/A",'0','1'],inplace=True)
data["d1_health_99"].replace([" ","No","Yes"],["N/A",'0','1'],inplace=True)
data["gender"].replace([" ","Male","Female"],["N/A",'1','2'],inplace=True)
data["household_size"].replace(["8 or more","Prefer not to say","Don't know"],["8",'10'
data["household_children"].replace(["5 or more","Prefer not to say","0","1","2","3","4"
data["employment_status"].replace(["Full time employment",'Part time employment','Full
data["qweek"].replace(["week 1","week 2","week 3","week 4","week 5","week 6","week 7","
data.to_csv("cleaned.csv")
```

In [4]:

```
# Cleaned dataset df
# Raw dataset df_raw
# full dataset df_full
df = pd.read_csv("cleaned.csv", index_col=0)
df_raw = pd.read_csv("COVID-19BehaviorData_CAN_USA.csv")
df_raw.rename(columns=lambda x: x+'_raw', inplace=True)
df_full = df.join(df_raw)
```

In [6]:

```
#Make Target Variable If they avoided going out frequently or not
df["i12_health_6"].replace([1,2], 1,inplace=True)
df["i12_health_6"].replace([4,5,3], 0, inplace=True)

#Impute These Variables Using Mode
df['i11_health'].fillna(df['i11_health'].mode()[0], inplace=True)
df['i5_health_99'].fillna(df['i5_health_99'].mode()[0], inplace=True)
df['d1_health_98'].fillna(df['d1_health_98'].mode()[0], inplace=True)
df['d1_health_99'].fillna(df['d1_health_99'].mode()[0], inplace=True)
```

In [ ]:

```
pd.set_option('display.max_columns', 500)
df.tail(10)
```

In [7]:

```
# Country Column
df['Country'] = df['RecordNo'].str[0:3]
df_raw['Country'] = df_raw['RecordNo_raw'].str[0:3]
df_full['country'] = df_full['RecordNo'].str[0:3]
df_full_usa = df_full.loc[df_full['country'] == 'USA']
df_full_can = df_full.loc[df_full['country'] == 'CAN']
```

In [8]:

```
# Scaling variables
min_max_scaler = MinMaxScaler()
df["age"]= min_max_scaler.fit_transform(df[["age"]])
df["i1_health"]= min_max_scaler.fit_transform(df[["i1_health"]])
df["i2_health"]= min_max_scaler.fit_transform(df[["i2_health"]])
df["i7a_health"]= min_max_scaler.fit_transform(df[["i7a_health"]])
df["i13_health"]= min_max_scaler.fit_transform(df[["i13_health"]])
```

In [ ]:

```
df["i13_health"].tail()
```

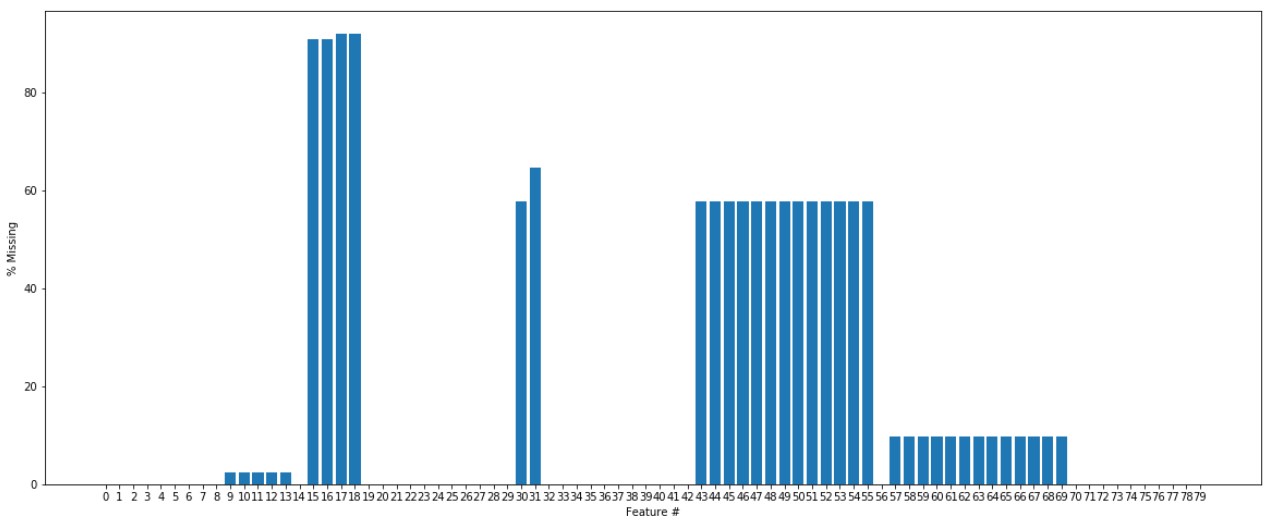
In [ ]:

```
pd.set_option('display.max_rows', 120)
df.isnull().sum()/len(df)*100
```

In [9]:

```
plt.figure(figsize=(20,8))
plt.xticks(np.arange(0, 79+1, 1.0))
pyplot.bar([x for x in range(len(df.columns))], df.isnull().sum()/len(df)*100)
plt.xlabel("Feature #")
plt.ylabel("% Missing")
```

Out[9]: Text(0, 0.5, '% Missing')



In [10]:

```
# Change type
for i in range (7,41):
    df[df.columns[i]] = df[df.columns[i]].astype('category')

for i in range (43,72):
    df[df.columns[i]] = df[df.columns[i]].astype('category')

df[df.columns[73]] = df[df.columns[73]].astype('category')
df[df.columns[75]] = df[df.columns[75]].astype('category')
```

```
df[df.columns[76]] = df[df.columns[76]].astype('category')
df[df.columns[77]] = df[df.columns[77]].astype('category')
df[df.columns[78]] = df[df.columns[78]].astype('category')

df[df.columns[0]] = df[df.columns[0]].astype('O')
df[df.columns[1]] = df[df.columns[1]].astype('O')
df[df.columns[56]] = df[df.columns[56]].astype('O')
```

In [11]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 28825 entries, 0 to 28824
Data columns (total 80 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Index            28825 non-null  object
 1   RecordNo         28825 non-null  object
 2   endtime          28825 non-null  object
 3   qweek            28825 non-null  int64
 4   i1_health        28825 non-null  float64
 5   i2_health        28825 non-null  float64
 6   i7a_health       28825 non-null  float64
 7   i3_health        28825 non-null  category
 8   i4_health        28825 non-null  category
 9   i5_health_1      28106 non-null  category
 10  i5_health_2      28106 non-null  category
 11  i5_health_3      28106 non-null  category
 12  i5_health_4      28106 non-null  category
 13  i5_health_5      28106 non-null  category
 14  i5_health_99     28825 non-null  category
 15  i5a_health       2606 non-null   category
 16  i6_health        2606 non-null   category
 17  i7b_health       2316 non-null   category
 18  i8_health        2266 non-null   category
 19  i9_health        28825 non-null  category
 20  i10_health       28825 non-null  category
 21  i11_health       28825 non-null  category
 22  i12_health_1     28825 non-null  category
 23  i12_health_2     28825 non-null  category
 24  i12_health_3     28825 non-null  category
 25  i12_health_4     28825 non-null  category
 26  i12_health_5     28825 non-null  category
 27  i12_health_6     28825 non-null  category
 28  i12_health_7     28825 non-null  category
 29  i12_health_8     28825 non-null  category
 30  i12_health_9     12160 non-null  category
 31  i12_health_10    10146 non-null  category
 32  i12_health_11    28825 non-null  category
 33  i12_health_12    28825 non-null  category
 34  i12_health_13    28825 non-null  category
 35  i12_health_14    28825 non-null  category
 36  i12_health_15    28825 non-null  category
 37  i12_health_16    28825 non-null  category
 38  i12_health_17    28825 non-null  category
 39  i12_health_18    28825 non-null  category
 40  i12_health_19    28825 non-null  category
 41  i12_health_20    28825 non-null  int64
 42  i13_health       28825 non-null  float64
 43  i14_health_1     12160 non-null  category
 44  i14_health_2     12160 non-null  category
 45  i14_health_3     12160 non-null  category
 46  i14_health_4     12160 non-null  category
 47  i14_health_5     12160 non-null  category
```

```
 48  i14_health_6       12160 non-null  category
 49  i14_health_7       12160 non-null  category
 50  i14_health_8       12160 non-null  category
 51  i14_health_9       12160 non-null  category
 52  i14_health_10      12160 non-null  category
 53  i14_health_96      12160 non-null  category
 54  i14_health_98      12160 non-null  category
 55  i14_health_99      12160 non-null  category
 56  i14_health_other   28825 non-null  object
 57  d1_health_1        26019 non-null  category
 58  d1_health_2        26019 non-null  category
 59  d1_health_3        26019 non-null  category
 60  d1_health_4        26019 non-null  category
 61  d1_health_5        26019 non-null  category
 62  d1_health_6        26019 non-null  category
 63  d1_health_7        26019 non-null  category
 64  d1_health_8        26019 non-null  category
 65  d1_health_9        26019 non-null  category
 66  d1_health_10       26019 non-null  category
 67  d1_health_11       26019 non-null  category
 68  d1_health_12       26019 non-null  category
 69  d1_health_13       26019 non-null  category
 70  d1_health_98       28825 non-null  category
 71  d1_health_99       28825 non-null  category
 72  weight             28825 non-null  float64
 73  gender             28825 non-null  category
 74  age                28825 non-null  float64
 75  region_state       28825 non-null  category
 76  household_size     28825 non-null  category
 77  household_children 28825 non-null  category
 78  employment_status  28825 non-null  category
 79  Country            28825 non-null  object
dtypes: category(67), float64(6), int64(2), object(5)
memory usage: 6.2+ MB
```

In [ ]:

```
df.tail()
```

In [ ]:

```
# EDA
```

In [ ]:

```
plt.figure(figsize=(15,8))
sns.barplot(x="qweek", y="i7b_health", hue="i12_health_6", palette=["m", "g"], data=df)
```

In [ ]:

```
sns.countplot(x="i9_health_raw", data=df_raw)
```

In [ ]:

```
sns.countplot(x="i12_health_6_raw", hue="country", data=df_full).set_title("Avoided goi
```

In [ ]:

```
i12_crosstab = pd.crosstab(df_full["i12_health_6_raw"], df_full["country"] ,margins = F
i12_crosstab
```

In [ ]:

```
i12_crosstab.plot(kind = 'bar', stacked = True)
```

In [ ]:

```
i12_crosstab_norm = i12_crosstab.div(i12_crosstab.sum(axis=0), axis=1)
i12_crosstab_norm
```

In [ ]:
```
i12_crosstab_norm.plot(kind = 'bar', stacked = True)
```

In [ ]:
```
def without_hue(plot, feature):
    total = len(feature)
    for p in ax.patches:
        percentage = '{:.1f}%'.format(100 * p.get_height()/total)
        x = p.get_x() + p.get_width() / 2 - 0.05
        y = p.get_y() + p.get_height()
        ax.annotate(percentage, (x, y), size = 12, ha='center', va='bottom')

    plt.xlabel(" ", size=12)
    plt.ylabel("Count", size=12)
    plt.ylim(0, 7000)
    plt.title("Avoided going out in general in USA", size=12)
    plt.show()

ax = sns.countplot(x="i12_health_6_raw", data=df_full_usa, order=df_full_usa["i12_healt

without_hue(ax, df_full_usa.i12_health_6)
```

In [ ]:
```
ax = sns.countplot(x="i12_health_6_raw", data=df_full_can, order=df_full_can["i12_healt
without_hue(ax, df_full_can.i12_health_6)
```

In [ ]:
```
check_missing = df_full.iloc[:,0:79].isnull()
count_missing = check_missing.apply(lambda x: x.value_counts())
with_missing = count_missing.loc[:, np.logical_not(count_missing.isna().any())]
```

In [ ]:
```
with_missing.iloc[1,:].plot(kind='barh', figsize=(40, 30))
```

In [ ]:
```
# Model
```

In [ ]:
```
# Compute the correlation matrix
corr = df_raw.corr()

# Generate a mask for the upper triangle
mask = np.triu(np.ones_like(corr, dtype=bool))

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 9))

# Generate a custom diverging colormap
cmap = sns.diverging_palette(230, 20, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
```

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

In [12]:

```
df_feats= df[[ 'i1_health', 'i2_health',
       'i7a_health', 'i3_health', 'i4_health', 'i5_health_99',
         'i9_health',
       'i10_health', 'i11_health', 'i12_health_1', 'i12_health_2',
       'i12_health_3', 'i12_health_4', 'i12_health_5',
       'i12_health_7', 'i12_health_8',
       'i12_health_11', 'i12_health_12', 'i12_health_13', 'i12_health_14',
       'i12_health_15', 'i12_health_16', 'i12_health_17', 'i12_health_18',
       'i12_health_19', 'i12_health_20', 'i13_health',
        'd1_health_98', 'd1_health_99',
       'weight', 'gender', 'age',  'household_size',
       'household_children', 'employment_status','i12_health_6']]
```

In [16]:

```
df_feats.isnull().sum()
```

Out[16]:

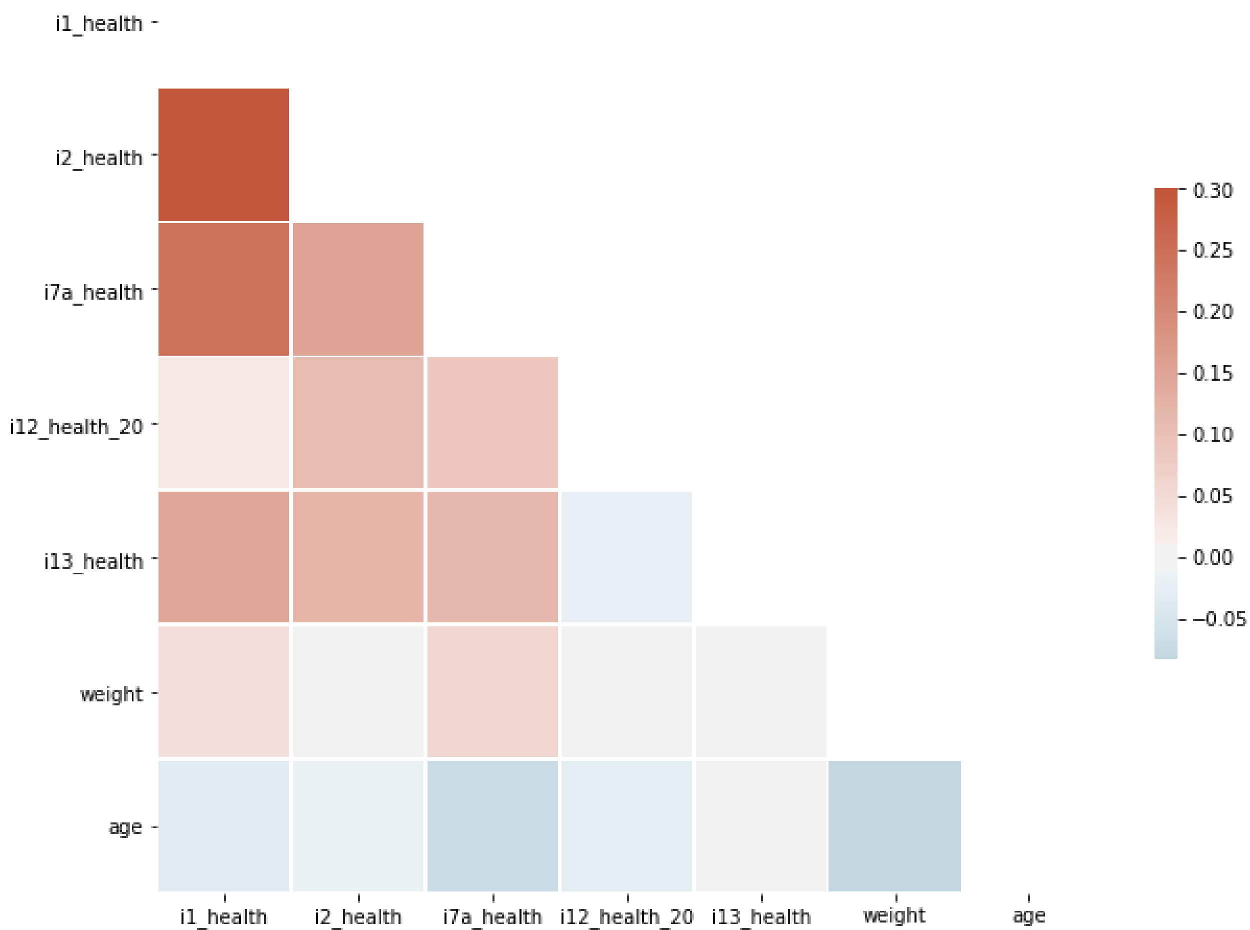
```
i1_health             0
i2_health             0
i7a_health            0
i3_health             0
i4_health             0
i5_health_99          0
i9_health             0
i10_health            0
i11_health            0
i12_health_1          0
i12_health_2          0
i12_health_3          0
i12_health_4          0
i12_health_5          0
i12_health_7          0
i12_health_8          0
i12_health_11         0
i12_health_12         0
i12_health_13         0
i12_health_14         0
i12_health_15         0
i12_health_16         0
i12_health_17         0
i12_health_18         0
i12_health_19         0
i12_health_20         0
i13_health            0
d1_health_98          0
d1_health_99          0
weight                0
gender                0
age                   0
household_size        0
household_children    0
employment_status     0
i12_health_6          0
dtype: int64
```

In [13]:

```
# Compute the correlation matrix
corr = df_feats.corr()

# Generate a mask for the upper triangle
mask = np.triu(np.ones_like(corr, dtype=bool))

# Set up the matplotlib figure
```

```
f, ax = plt.subplots(figsize=(11, 9))

# Generate a custom diverging colormap
cmap = sns.diverging_palette(230, 20, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
```

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x1f38f9f38c8>



In [14]:

```
x = df_feats.drop(["i12_health_6"], axis=1)
y = df_feats["i12_health_6"]

X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=
```

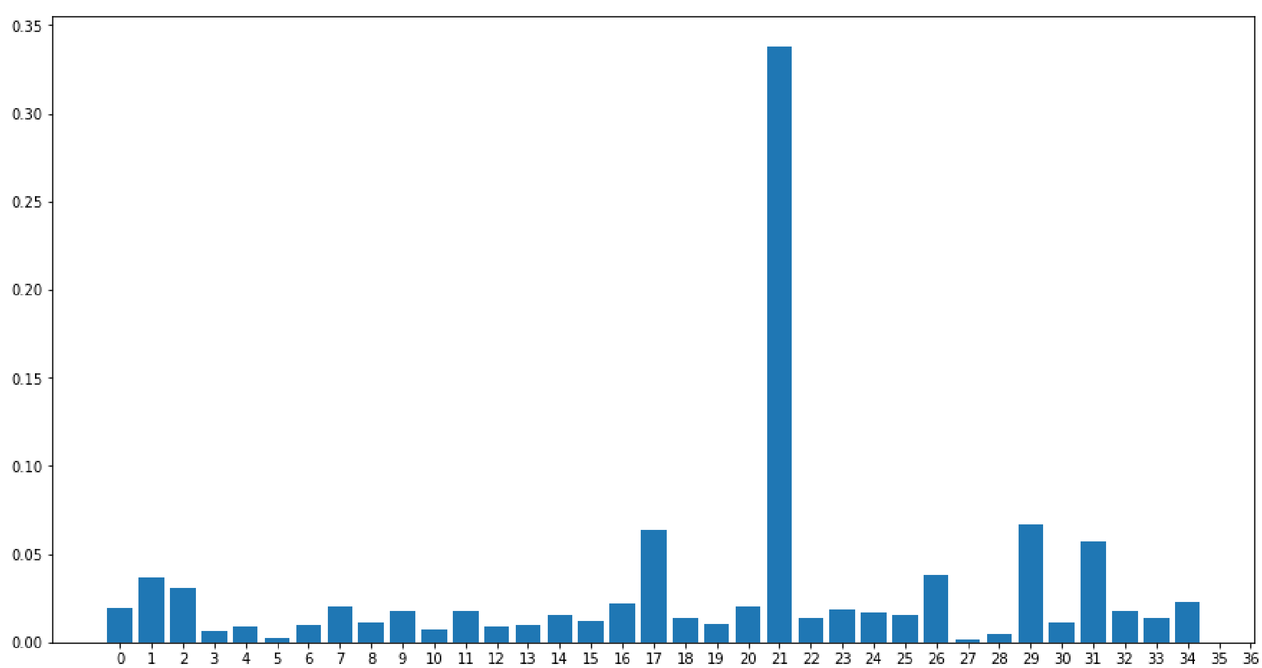
In [15]:

```
# define the model
model1 = DecisionTreeClassifier()
# fit the model
model1.fit(X_train, y_train)
# get importance
importance = model1.feature_importances_
# summarize feature importance
for i,v in enumerate(importance):
        print('Feature: %0d, Score: %.5f' % (i,v))
# plot feature importance
```

```
plt.figure(figsize=(15,8))
plt.xticks(np.arange(0, 55, 1.0))
pyplot.bar([x for x in range(len(importance))], importance)
```

```
Feature: 0, Score: 0.01920
Feature: 1, Score: 0.03669
Feature: 2, Score: 0.03064
Feature: 3, Score: 0.00624
Feature: 4, Score: 0.00921
Feature: 5, Score: 0.00262
Feature: 6, Score: 0.00993
Feature: 7, Score: 0.02013
Feature: 8, Score: 0.01149
Feature: 9, Score: 0.01751
Feature: 10, Score: 0.00722
Feature: 11, Score: 0.01753
Feature: 12, Score: 0.00901
Feature: 13, Score: 0.00933
Feature: 14, Score: 0.01557
Feature: 15, Score: 0.01232
Feature: 16, Score: 0.02209
Feature: 17, Score: 0.06375
Feature: 18, Score: 0.01388
Feature: 19, Score: 0.01022
Feature: 20, Score: 0.02059
Feature: 21, Score: 0.33844
Feature: 22, Score: 0.01334
Feature: 23, Score: 0.01829
Feature: 24, Score: 0.01699
Feature: 25, Score: 0.01528
Feature: 26, Score: 0.03829
Feature: 27, Score: 0.00110
Feature: 28, Score: 0.00455
Feature: 29, Score: 0.06657
Feature: 30, Score: 0.01100
Feature: 31, Score: 0.05684
Feature: 32, Score: 0.01760
Feature: 33, Score: 0.01347
Feature: 34, Score: 0.02307
```

Out[15]: <BarContainer object of 35 artists>



In [17]:

```
graph=[]
graph2=[]
print(sum(importance))
for i in range (35):
    graph.append(X_train.columns[i])
    graph2.append(importance[i])
    print(X_train.columns[i]," ",i, ": ", importance[i])
```

```
0.9999999999999999
i1_health   0 :  0.019200790124659504
i2_health   1 :  0.036686946555292026
i7a_health   2 :  0.03064250093035997
i3_health   3 :  0.006242163895913354
i4_health   4 :  0.009207985049177537
i5_health_99   5 :  0.0026245949671743793
i9_health   6 :  0.009931495113005624
i10_health   7 :  0.02012956485676338
i11_health   8 :  0.011494545118959743
i12_health_1   9 :  0.017508732454955638
i12_health_2   10 :  0.007215898585836496
i12_health_3   11 :  0.017534531388162952
i12_health_4   12 :  0.009009882785721247
i12_health_5   13 :  0.00933428111933562
i12_health_7   14 :  0.01557076759475509
i12_health_8   15 :  0.012315506654286993
i12_health_11   16 :  0.022086155837100815
i12_health_12   17 :  0.06375354114849442
i12_health_13   18 :  0.013880937723536595
i12_health_14   19 :  0.010217441101093017
i12_health_15   20 :  0.02058506954042459
i12_health_16   21 :  0.33844480837679086
i12_health_17   22 :  0.013342629626018255
i12_health_18   23 :  0.018285504816849863
i12_health_19   24 :  0.016985086313315228
i12_health_20   25 :  0.015282051002820863
i13_health   26 :  0.0382888358021357
d1_health_98   27 :  0.001099686598628937
d1_health_99   28 :  0.004548393739772186
weight   29 :  0.0665690231119636
gender   30 :  0.011001560722636299
age   31 :  0.056836543539419956
household_size   32 :  0.017603393160531156
household_children   33 :  0.013466908587988635
employment_status   34 :  0.023072242056119442
```

In [18]:

```
model = KNeighborsClassifier(n_neighbors=19)

model.fit(X_train, y_train)

pred = model.predict(X_test)
pred2 = model.predict(X_train)

print(confusion_matrix(y_test, pred))
print(" ")
print(confusion_matrix(y_train, pred2))
print("Test")
print(classification_report(y_test, pred))
print("Train")
print(classification_report(y_train, pred2))
```

```
[[1785 1098]
 [ 439 3885]]
```

```
[[ 5506  3199]
 [ 1188 11725]]
Test
              precision    recall  f1-score   support

           0       0.80      0.62      0.70      2883
           1       0.78      0.90      0.83      4324

    accuracy                           0.79      7207
   macro avg       0.79      0.76      0.77      7207
weighted avg       0.79      0.79      0.78      7207

Train
              precision    recall  f1-score   support

           0       0.82      0.63      0.72      8705
           1       0.79      0.91      0.84     12913

    accuracy                           0.80     21618
   macro avg       0.80      0.77      0.78     21618
weighted avg       0.80      0.80      0.79     21618
```

In [19]:

```
num_folds = 10
#scoring (note that Python is set to maximize the MSE function so you need to make it n
scoring = "accuracy"
#split for crossvalidation
kfold = KFold(n_splits=num_folds, random_state=101,shuffle=True)
cv_results = cross_val_score(model, X_train, y_train, cv=kfold, scoring=scoring)
msg = "KNN: %f (%f)" % (cv_results.mean(),cv_results.std())
print(msg)
```

```
KNN: 0.783051 (0.008929)
```

In [20]:

```
k_values = np.array([1,3,5,7,9,13,15,17,19,21])
param_grid = dict(n_neighbors=k_values)
#define the grid search cross validation method
grid=GridSearchCV(estimator=model,param_grid=param_grid,scoring=scoring,cv=kfold)

#generate the results of the grid search cross validation method
grid_result=grid.fit(X_train,y_train)

#print the best score achieved by any of the specified k_values
print("Best: %f using %s" % (grid_result.best_score_,grid_result.best_params_))
```

```
Best: 0.783051 using {'n_neighbors': 19}
```

In [21]:

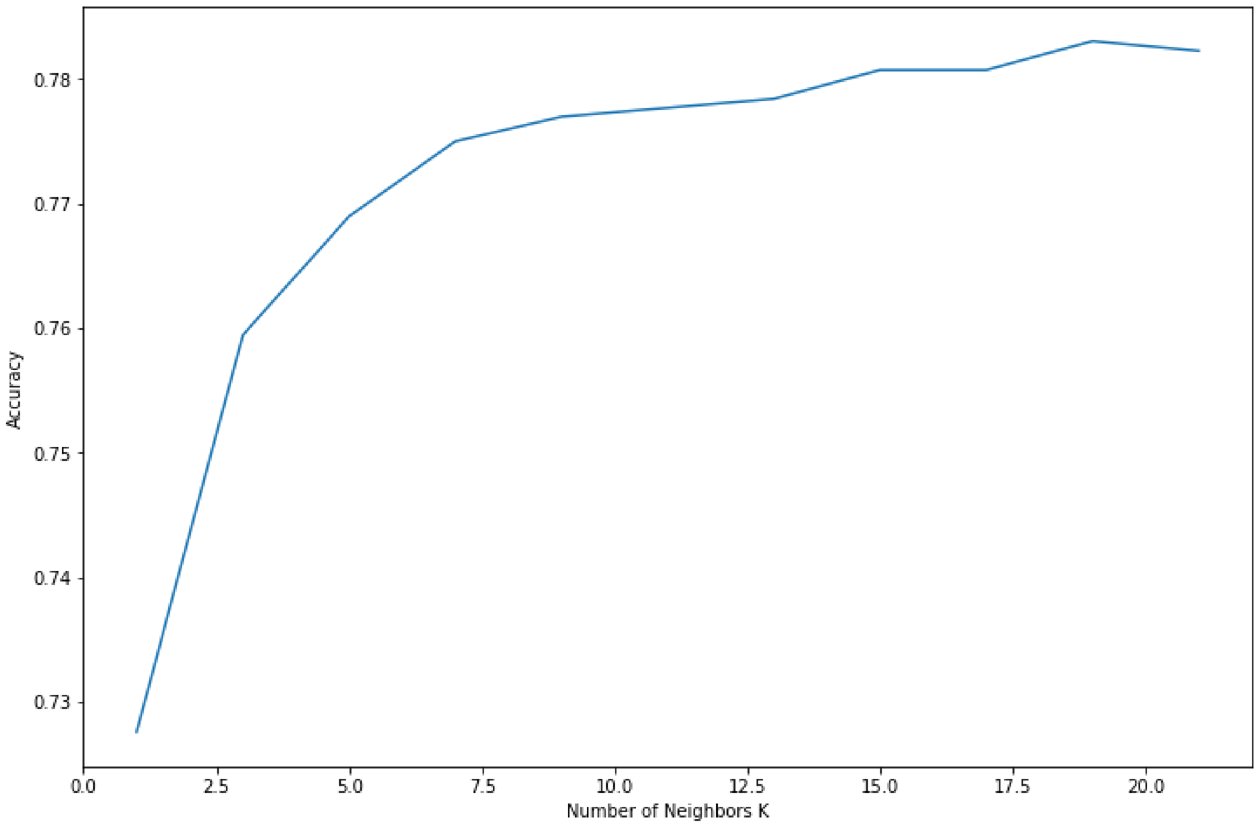
```
means = grid_result.cv_results_["mean_test_score"]
stds = grid_result.cv_results_["std_test_score"]
params = grid_result.cv_results_["params"]
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with %s" % (mean, stdev, param))
```

```
0.727634 (0.007736) with {'n_neighbors': 1}
0.759460 (0.008357) with {'n_neighbors': 3}
0.768988 (0.009531) with {'n_neighbors': 5}
0.775002 (0.008954) with {'n_neighbors': 7}
0.776991 (0.011011) with {'n_neighbors': 9}
0.778425 (0.008877) with {'n_neighbors': 13}
0.780738 (0.009193) with {'n_neighbors': 15}
```

```
0.780738 (0.009653) with {'n_neighbors': 17}
0.783051 (0.008929) with {'n_neighbors': 19}
0.782264 (0.009995) with {'n_neighbors': 21}
```

In [22]:

```python
#plot the accuracy levels at different K
plt.figure(figsize=(12,8))
plt.plot(k_values,means)
plt.xlabel("Number of Neighbors K")
plt.ylabel("Accuracy")
plt.savefig('AccuracyChart.png')
```



In [ ]: