In [21]:
```
import sys
# Install packages
!pip install category_encoders==2.0.0
!pip install pandas-profiling==2.3.0
!pip install plotly==4.1.1
```

```
Requirement already satisfied: category_encoders==2.0.0 in c:\users\asg\.conda\envs\lambda\lib\site-
packages (2.0.0)
Requirement already satisfied: pandas>=0.21.1 in c:\users\asg\.conda\envs\lambda\lib\site-packages
(from category_encoders==2.0.0) (0.23.4)
Requirement already satisfied: numpy>=1.11.3 in c:\users\asg\.conda\envs\lambda\lib\site-packages (f
rom category_encoders==2.0.0) (1.16.4)
Requirement already satisfied: statsmodels>=0.6.1 in c:\users\asg\.conda\envs\lambda\lib\site-packag
es (from category_encoders==2.0.0) (0.10.1)
Requirement already satisfied: patsy>=0.4.1 in c:\users\asg\.conda\envs\lambda\lib\site-packages (fr
om category_encoders==2.0.0) (0.5.1)
Requirement already satisfied: scipy>=0.19.0 in c:\users\asg\.conda\envs\lambda\lib\site-packages (f
rom category_encoders==2.0.0) (1.3.1)
Requirement already satisfied: scikit-learn>=0.20.0 in c:\users\asg\.conda\envs\lambda\lib\site-pack
ages (from category_encoders==2.0.0) (0.21.3)
Requirement already satisfied: python-dateutil>=2.5.0 in c:\users\asg\.conda\envs\lambda\lib\site-pa
ckages (from pandas>=0.21.1->category_encoders==2.0.0) (2.8.0)
Requirement already satisfied: pytz>=2011k in c:\users\asg\.conda\envs\lambda\lib\site-packages (fro
m pandas>=0.21.1->category_encoders==2.0.0) (2019.2)
Requirement already satisfied: six in c:\users\asg\.conda\envs\lambda\lib\site-packages (from patsy>
=0.4.1->category_encoders==2.0.0) (1.12.0)
Requirement already satisfied: joblib>=0.11 in c:\users\asg\.conda\envs\lambda\lib\site-packages (fr
om scikit-learn>=0.20.0->category_encoders==2.0.0) (0.13.2)
Requirement already satisfied: pandas-profiling==2.3.0 in c:\users\asg\.conda\envs\lambda\lib\site-p
ackages (2.3.0)
Requirement already satisfied: htmlmin>=0.1.12 in c:\users\asg\.conda\envs\lambda\lib\site-packages
(from pandas-profiling==2.3.0) (0.1.12)
Requirement already satisfied: phik>=0.9.8 in c:\users\asg\.conda\envs\lambda\lib\site-packages (fro
m pandas-profiling==2.3.0) (0.9.8)
Requirement already satisfied: jinja2>=2.8 in c:\users\asg\.conda\envs\lambda\lib\site-packages (fro
m pandas-profiling==2.3.0) (2.10.1)
Requirement already satisfied: pandas>=0.19 in c:\users\asg\.conda\envs\lambda\lib\site-packages (fr
om pandas-profiling==2.3.0) (0.23.4)
Requirement already satisfied: matplotlib>=1.4 in c:\users\asg\.conda\envs\lambda\lib\site-packages
(from pandas-profiling==2.3.0) (3.1.1)
Requirement already satisfied: missingno>=0.4.2 in c:\users\asg\.conda\envs\lambda\lib\site-packages
(from pandas-profiling==2.3.0) (0.4.2)
Requirement already satisfied: astropy in c:\users\asg\.conda\envs\lambda\lib\site-packages (from pa
ndas-profiling==2.3.0) (3.2.1)
Requirement already satisfied: confuse>=1.0.0 in c:\users\asg\.conda\envs\lambda\lib\site-packages
(from pandas-profiling==2.3.0) (1.0.0)
Requirement already satisfied: scipy>=1.1.0 in c:\users\asg\.conda\envs\lambda\lib\site-packages (fr
om phik>=0.9.8->pandas-profiling==2.3.0) (1.3.1)
Requirement already satisfied: pytest-pylint>=0.13.0 in c:\users\asg\.conda\envs\lambda\lib\site-pac
kages (from phik>=0.9.8->pandas-profiling==2.3.0) (0.14.1)
Requirement already satisfied: jupyter-client>=5.2.3 in c:\users\asg\.conda\envs\lambda\lib\site-pac
kages (from phik>=0.9.8->pandas-profiling==2.3.0) (5.3.1)
Requirement already satisfied: pytest>=4.0.2 in c:\users\asg\.conda\envs\lambda\lib\site-packages (f
rom phik>=0.9.8->pandas-profiling==2.3.0) (5.0.1)
Requirement already satisfied: numpy>=1.15.4 in c:\users\asg\.conda\envs\lambda\lib\site-packages (f
rom phik>=0.9.8->pandas-profiling==2.3.0) (1.16.4)
Requirement already satisfied: nbconvert>=5.3.1 in c:\users\asg\.conda\envs\lambda\lib\site-packages
(from phik>=0.9.8->pandas-profiling==2.3.0) (5.5.0)
Requirement already satisfied: numba>=0.38.1 in c:\users\asg\.conda\envs\lambda\lib\site-packages (f
rom phik>=0.9.8->pandas-profiling==2.3.0) (0.45.1)
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\asg\.conda\envs\lambda\lib\site-packages
(from jinja2>=2.8->pandas-profiling==2.3.0) (1.1.1)
Requirement already satisfied: python-dateutil>=2.5.0 in c:\users\asg\.conda\envs\lambda\lib\site-pa
ckages (from pandas>=0.19->pandas-profiling==2.3.0) (2.8.0)
Requirement already satisfied: pytz>=2011k in c:\users\asg\.conda\envs\lambda\lib\site-packages (fro
m pandas>=0.19->pandas-profiling==2.3.0) (2019.2)
Requirement already satisfied: cycler>=0.10 in c:\users\asg\.conda\envs\lambda\lib\site-packages (fr
om matplotlib>=1.4->pandas-profiling==2.3.0) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\asg\.conda\envs\lambda\lib\site-package
s (from matplotlib>=1.4->pandas-profiling==2.3.0) (1.1.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in c:\users\asg\.conda\envs
\lambda\lib\site-packages (from matplotlib>=1.4->pandas-profiling==2.3.0) (2.4.2)
Requirement already satisfied: seaborn in c:\users\asg\.conda\envs\lambda\lib\site-packages (from mi
ssingno>=0.4.2->pandas-profiling==2.3.0) (0.9.0)
Requirement already satisfied: pyyaml in c:\users\asg\.conda\envs\lambda\lib\site-packages (from con
fuse>=1.0.0->pandas-profiling==2.3.0) (5.1.2)
```

```
Requirement already satisfied: six in c:\users\asg\.conda\envs\lambda\lib\site-packages (from pytest
-pylint>=0.13.0->phik>=0.9.8->pandas-profiling==2.3.0) (1.12.0)
Requirement already satisfied: pylint>=1.4.5 in c:\users\asg\.conda\envs\lambda\lib\site-packages (f
rom pytest-pylint>=0.13.0->phik>=0.9.8->pandas-profiling==2.3.0) (2.3.1)
Requirement already satisfied: pyzmq>=13 in c:\users\asg\.conda\envs\lambda\lib\site-packages (from
jupyter-client>=5.2.3->phik>=0.9.8->pandas-profiling==2.3.0) (18.1.0)
Requirement already satisfied: jupyter-core in c:\users\asg\.conda\envs\lambda\lib\site-packages (fr
om jupyter-client>=5.2.3->phik>=0.9.8->pandas-profiling==2.3.0) (4.5.0)
Requirement already satisfied: traitlets in c:\users\asg\.conda\envs\lambda\lib\site-packages (from
jupyter-client>=5.2.3->phik>=0.9.8->pandas-profiling==2.3.0) (4.3.2)
Requirement already satisfied: tornado>=4.1 in c:\users\asg\.conda\envs\lambda\lib\site-packages (fr
om jupyter-client>=5.2.3->phik>=0.9.8->pandas-profiling==2.3.0) (6.0.3)
Requirement already satisfied: py>=1.5.0 in c:\users\asg\.conda\envs\lambda\lib\site-packages (from
pytest>=4.0.2->phik>=0.9.8->pandas-profiling==2.3.0) (1.8.0)
Requirement already satisfied: packaging in c:\users\asg\.conda\envs\lambda\lib\site-packages (from
pytest>=4.0.2->phik>=0.9.8->pandas-profiling==2.3.0) (19.1)
Requirement already satisfied: attrs>=17.4.0 in c:\users\asg\.conda\envs\lambda\lib\site-packages (f
rom pytest>=4.0.2->phik>=0.9.8->pandas-profiling==2.3.0) (19.1.0)
Requirement already satisfied: more-itertools>=4.0.0 in c:\users\asg\.conda\envs\lambda\lib\site-pac
kages (from pytest>=4.0.2->phik>=0.9.8->pandas-profiling==2.3.0) (7.2.0)
Requirement already satisfied: atomicwrites>=1.0 in c:\users\asg\.conda\envs\lambda\lib\site-package
s (from pytest>=4.0.2->phik>=0.9.8->pandas-profiling==2.3.0) (1.3.0)
Requirement already satisfied: pluggy<1.0,>=0.12 in c:\users\asg\.conda\envs\lambda\lib\site-package
s (from pytest>=4.0.2->phik>=0.9.8->pandas-profiling==2.3.0) (0.12.0)
Requirement already satisfied: importlib-metadata>=0.12 in c:\users\asg\.conda\envs\lambda\lib\site-
packages (from pytest>=4.0.2->phik>=0.9.8->pandas-profiling==2.3.0) (0.19)
Requirement already satisfied: wcwidth in c:\users\asg\.conda\envs\lambda\lib\site-packages (from py
test>=4.0.2->phik>=0.9.8->pandas-profiling==2.3.0) (0.1.7)
Requirement already satisfied: colorama in c:\users\asg\.conda\envs\lambda\lib\site-packages (from p
ytest>=4.0.2->phik>=0.9.8->pandas-profiling==2.3.0) (0.4.1)
Requirement already satisfied: mistune>=0.8.1 in c:\users\asg\.conda\envs\lambda\lib\site-packages
(from nbconvert>=5.3.1->phik>=0.9.8->pandas-profiling==2.3.0) (0.8.4)
Requirement already satisfied: pygments in c:\users\asg\.conda\envs\lambda\lib\site-packages (from n
bconvert>=5.3.1->phik>=0.9.8->pandas-profiling==2.3.0) (2.4.2)
Requirement already satisfied: testpath in c:\users\asg\.conda\envs\lambda\lib\site-packages (from n
bconvert>=5.3.1->phik>=0.9.8->pandas-profiling==2.3.0) (0.4.2)
Requirement already satisfied: bleach in c:\users\asg\.conda\envs\lambda\lib\site-packages (from nbc
onvert>=5.3.1->phik>=0.9.8->pandas-profiling==2.3.0) (3.1.0)
Requirement already satisfied: entrypoints>=0.2.2 in c:\users\asg\.conda\envs\lambda\lib\site-packag
es (from nbconvert>=5.3.1->phik>=0.9.8->pandas-profiling==2.3.0) (0.3)
Requirement already satisfied: defusedxml in c:\users\asg\.conda\envs\lambda\lib\site-packages (from
nbconvert>=5.3.1->phik>=0.9.8->pandas-profiling==2.3.0) (0.6.0)
Requirement already satisfied: nbformat>=4.4 in c:\users\asg\.conda\envs\lambda\lib\site-packages (f
rom nbconvert>=5.3.1->phik>=0.9.8->pandas-profiling==2.3.0) (4.4.0)
Requirement already satisfied: pandocfilters>=1.4.1 in c:\users\asg\.conda\envs\lambda\lib\site-pack
ages (from nbconvert>=5.3.1->phik>=0.9.8->pandas-profiling==2.3.0) (1.4.2)
Requirement already satisfied: llvmlite>=0.29.0dev0 in c:\users\asg\.conda\envs\lambda\lib\site-pack
ages (from numba>=0.38.1->phik>=0.9.8->pandas-profiling==2.3.0) (0.29.0)
Requirement already satisfied: setuptools in c:\users\asg\.conda\envs\lambda\lib\site-packages (from
kiwisolver>=1.0.1->matplotlib>=1.4->pandas-profiling==2.3.0) (41.0.1)
Requirement already satisfied: astroid<3,>=2.2.0 in c:\users\asg\.conda\envs\lambda\lib\site-package
s (from pylint>=1.4.5->pytest-pylint>=0.13.0->phik>=0.9.8->pandas-profiling==2.3.0) (2.2.5)
Requirement already satisfied: isort<5,>=4.2.5 in c:\users\asg\.conda\envs\lambda\lib\site-packages
(from pylint>=1.4.5->pytest-pylint>=0.13.0->phik>=0.9.8->pandas-profiling==2.3.0) (4.3.21)
Requirement already satisfied: mccabe<0.7,>=0.6 in c:\users\asg\.conda\envs\lambda\lib\site-packages
(from pylint>=1.4.5->pytest-pylint>=0.13.0->phik>=0.9.8->pandas-profiling==2.3.0) (0.6.1)
Requirement already satisfied: ipython-genutils in c:\users\asg\.conda\envs\lambda\lib\site-packages
(from traitlets->jupyter-client>=5.2.3->phik>=0.9.8->pandas-profiling==2.3.0) (0.2.0)
Requirement already satisfied: decorator in c:\users\asg\.conda\envs\lambda\lib\site-packages (from
traitlets->jupyter-client>=5.2.3->phik>=0.9.8->pandas-profiling==2.3.0) (4.4.0)
Requirement already satisfied: zipp>=0.5 in c:\users\asg\.conda\envs\lambda\lib\site-packages (from
importlib-metadata>=0.12->pytest>=4.0.2->phik>=0.9.8->pandas-profiling==2.3.0) (0.5.2)
Requirement already satisfied: webencodings in c:\users\asg\.conda\envs\lambda\lib\site-packages (fr
om bleach->nbconvert>=5.3.1->phik>=0.9.8->pandas-profiling==2.3.0) (0.5.1)
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in c:\users\asg\.conda\envs\lambda\lib\site-p
ackages (from nbformat>=4.4->nbconvert>=5.3.1->phik>=0.9.8->pandas-profiling==2.3.0) (3.0.2)
Requirement already satisfied: typed-ast>=1.3.0; implementation_name == "cpython" in c:\users\asg\.c
onda\envs\lambda\lib\site-packages (from astroid<3,>=2.2.0->pylint>=1.4.5->pytest-pylint>=0.13.0->ph
ik>=0.9.8->pandas-profiling==2.3.0) (1.4.0)
Requirement already satisfied: lazy-object-proxy in c:\users\asg\.conda\envs\lambda\lib\site-package
s (from astroid<3,>=2.2.0->pylint>=1.4.5->pytest-pylint>=0.13.0->phik>=0.9.8->pandas-profiling==2.3.
```

```
0) (1.4.2)
Requirement already satisfied: wrapt in c:\users\asg\.conda\envs\lambda\lib\site-packages (from astr
oid<3,>=2.2.0->pylint>=1.4.5->pytest-pylint>=0.13.0->phik>=0.9.8->pandas-profiling==2.3.0) (1.11.2)
Requirement already satisfied: pyrsistent>=0.14.0 in c:\users\asg\.conda\envs\lambda\lib\site-packag
es (from jsonschema!=2.5.0,>=2.4->nbformat>=4.4->nbconvert>=5.3.1->phik>=0.9.8->pandas-profiling==2.
3.0) (0.14.11)
Requirement already satisfied: plotly==4.1.1 in c:\users\asg\.conda\envs\lambda\lib\site-packages
(4.1.1)
Requirement already satisfied: retrying>=1.3.3 in c:\users\asg\.conda\envs\lambda\lib\site-packages
(from plotly==4.1.1) (1.3.3)
Requirement already satisfied: six in c:\users\asg\.conda\envs\lambda\lib\site-packages (from plotly
==4.1.1) (1.12.0)
```

In [45]:
```
#Fetch smoking data file
#from google.colab import files
#uploaded = files.upload()
```

In [1]:
```
# Load smoking data
import pandas as pd
import io
# df_smoking = pd.read_csv(io.StringIO(uploaded['C:\\Users\\ASG\\Desktop\\cancerxx - for_import.cs
v'].decode('utf-8')))
df_smoking = pd.read_csv('C:\\Users\\ASG\\Desktop\\cancerxx - for_import.csv')
df_smoking.head()
```

Out[1]:

	language	cereal_serve_per_month	cereal_times_per_month	more_than_one_cereal_type	milk_serve_per_month	milk_times
0	5	3	2	2.0	3	
1	4	0	0	NaN	0	
2	5	5	2	2.0	5	
3	3	1	1	2.0	4	
4	5	2	2	1.0	0	

5 rows × 92 columns

In [2]:
```
# We assess the contents of df_smoking
df_smoking_shape = df_smoking.shape
print ('df_smoking Shape')
print (df_smoking_shape, '\n')
print ('df_smoking Count')
print (df_smoking.count(), '\n')
print ('df_smoking NaN Count')
print (df_smoking.isna().sum(), '\n')
print ('df_smoking Describe')
print (df_smoking.describe())
```

```
df_smoking Shape
(33672, 92)

df_smoking Count
language                          33672
cereal_serve_per_month            33672
cereal_times_per_month            33672
more_than_one_cereal_type         22858
milk_serve_per_month              33672
milk_times_per_month              33672
milk_type                         24044
soda_serve_per_month              33672
soda_times_per_month              33672
juice_serve_per_month             33672
juice_times_per_month             33672
coffee_serve_per_month            33672
coffee_times_per_month            33672
sports_drink_serve_per_month      33672
sports_drink_times_per_month      33672
fruit_drink_serve_per_month       33672
fruit_drink_times_per_month       33672
fruit_eat_serve_per_month         33672
fruit_eat_times_per_month         33672
salad_eat_serve_per_month         33672
salad_eat_times_per_month         33672
fries_eat_serve_per_month         33672
fries_eat_times_per_month         33672
potatoe_eat_serve_per_month       33672
potatoe_eat_times_per_month       33672
beans_eat_serve_per_month         33672
beans_eat_times_per_month         33672
grains_eat_serve_per_month        33672
grains_eat_times_per_month        33672
vegies_eat_serve_per_month        33672
                                  ...
vitD_reason                        6906
1st_kind_cereal_eaten             22858
2nd_kind_cereal_eaten              9958
walk_past_wk                      33672
walk_number_wk                    10246
single_walk_distance              10229
single_walk_time                  10229
walk_leisure_past_wk              32778
walk_leisure_number_wk            16074
walk_leisure_ distance            16055
walk_leisure_ time                16055
see_walking_from_home             33672
weather_discourages_walk          33672
walkway_existence                 33672
walkable_retail                   33672
walkable_bus_stop                 33672
walkable_entertainment            33672
walkable_relaxation               33672
streets_have_walkways             33672
traffic_discourages_walking       33672
crime_discourages_walking         33672
animals_discourage_walking        33672
cigarette_even_once               33672
cigar_even_once                   33672
pipe_even_once                    33672
smokeless_even_once               33672
had_genetic_counseling            33672
genetic_counseling_with_MD        33672
genetic_counseling_for_cancer     33672
cigarettes_per_day                 7602
Length: 92, dtype: int64

df_smoking NaN Count
language                              0
cereal_serve_per_month                0
```

```
cereal_times_per_month                0
more_than_one_cereal_type         10814
milk_serve_per_month                  0
milk_times_per_month                  0
milk_type                          9628
soda_serve_per_month                  0
soda_times_per_month                  0
juice_serve_per_month                 0
juice_times_per_month                 0
coffee_serve_per_month                0
coffee_times_per_month                0
sports_drink_serve_per_month          0
sports_drink_times_per_month          0
fruit_drink_serve_per_month           0
fruit_drink_times_per_month           0
fruit_eat_serve_per_month             0
fruit_eat_times_per_month             0
salad_eat_serve_per_month             0
salad_eat_times_per_month             0
fries_eat_serve_per_month             0
fries_eat_times_per_month             0
potatoe_eat_serve_per_month           0
potatoe_eat_times_per_month           0
beans_eat_serve_per_month             0
beans_eat_times_per_month             0
grains_eat_serve_per_month            0
grains_eat_times_per_month            0
vegies_eat_serve_per_month            0
                                  ...
vitD_reason                       26766
1st_kind_cereal_eaten             10814
2nd_kind_cereal_eaten             23714
walk_past_wk                          0
walk_number_wk                    23426
single_walk_distance              23443
single_walk_time                  23443
walk_leisure_past_wk                894
walk_leisure_number_wk            17598
walk_leisure_ distance            17617
walk_leisure_ time                17617
see_walking_from_home                 0
weather_discourages_walk              0
walkway_existence                     0
walkable_retail                       0
walkable_bus_stop                     0
walkable_entertainment                0
walkable_relaxation                   0
streets_have_walkways                 0
traffic_discourages_walking           0
crime_discourages_walking             0
animals_discourage_walking            0
cigarette_even_once                   0
cigar_even_once                       0
pipe_even_once                        0
smokeless_even_once                   0
had_genetic_counseling                0
genetic_counseling_with_MD            0
genetic_counseling_for_cancer         0
cigarettes_per_day                26070
Length: 92, dtype: int64

df_smoking Describe
           language  cereal_serve_per_month  cereal_times_per_month  \
count  33672.000000            33672.000000            33672.000000
mean       4.670587               57.649976                1.933803
std        1.191156              226.021427                1.840685
min        1.000000                0.000000                0.000000
25%        4.000000                0.000000                0.000000
50%        5.000000                2.000000                2.000000
75%        5.000000                4.000000                3.000000
max        9.000000              999.000000                9.000000
```

```
       more_than_one_cereal_type  milk_serve_per_month  milk_times_per_month  \
count               22858.000000          33672.000000          33672.000000   
mean                    1.596246             58.546804              1.790419   
std                     0.694819            227.558558              1.814330   
min                     1.000000              0.000000              0.000000   
25%                     1.000000              1.000000              1.000000   
50%                     2.000000              1.000000              2.000000   
75%                     2.000000              4.000000              2.000000   
max                     9.000000            999.000000              9.000000   

          milk_type  soda_serve_per_month  soda_times_per_month  \
count  24044.000000          33672.000000          33672.000000   
mean       2.256696             57.167082              1.539172   
std        1.143486            227.664658              1.951008   
min        1.000000              0.000000              0.000000   
25%        1.000000              0.000000              0.000000   
50%        2.000000              1.000000              1.000000   
75%        3.000000              3.000000              3.000000   
max        9.000000            999.000000              9.000000   

       juice_serve_per_month         ...           crime_discourages_walking  \
count           33672.000000         ...                        33672.000000   
mean               59.006266         ...                            2.240259   
std               230.053824         ...                            1.520633   
min                 0.000000         ...                            1.000000   
25%                 0.000000         ...                            2.000000   
50%                 1.000000         ...                            2.000000   
75%                 3.000000         ...                            2.000000   
max               999.000000         ...                            9.000000   

       animals_discourage_walking  cigarette_even_once  cigar_even_once  \
count                33672.000000         33672.000000     33672.000000   
mean                     2.247505             2.217035         2.111368   
std                      1.475663             1.465328         1.525372   
min                      1.000000             1.000000         1.000000   
25%                      2.000000             2.000000         2.000000   
50%                      2.000000             2.000000         2.000000   
75%                      2.000000             2.000000         2.000000   
max                      9.000000             9.000000         9.000000   

       pipe_even_once  smokeless_even_once  had_genetic_counseling  \
count    33672.000000         33672.000000            33672.000000   
mean         2.231171             2.256771                2.432080   
std          1.468190             1.453709                1.596199   
min          1.000000             1.000000                1.000000   
25%          2.000000             2.000000                2.000000   
50%          2.000000             2.000000                2.000000   
75%          2.000000             2.000000                2.000000   
max          9.000000             9.000000                9.000000   

       genetic_counseling_with_MD  genetic_counseling_for_cancer  \
count                33672.000000                   33672.000000   
mean                     2.422785                       2.446038   
std                      1.605351                       1.597866   
min                      1.000000                       1.000000   
25%                      2.000000                       2.000000   
50%                      2.000000                       2.000000   
75%                      2.000000                       2.000000   
max                      9.000000                       9.000000   

       cigarettes_per_day  
count         7602.000000  
mean            22.540647  
std             26.525465  
min              1.000000  
25%              6.000000  
50%             15.000000  
75%             20.000000  
max             99.000000  
```

```
[8 rows x 92 columns]
```

In [3]:
```
# Replace NaN to improve data format
import numpy as np
df_smoking1 = df_smoking.replace ({np.NaN: 0})
df_smoking1.head()
```

Out[3]:

	language	cereal_serve_per_month	cereal_times_per_month	more_than_one_cereal_type	milk_serve_per_month	milk_times
0	5	3	2	2.0	3	
1	4	0	0	0.0	0	
2	5	5	2	2.0	5	
3	3	1	1	2.0	4	
4	5	2	2	1.0	0	

5 rows × 92 columns

In [4]:
```
# Set up boolean columns such that yes = 1 and no = 0
features1 = {'more_than_one_cereal_type', 'vitamin_past_month', 'multivitamin_past_month', 'calcium_
past_month', 'vitD_past_month', 'walk_past_wk', 'walk_leisure_past_wk',
            'walkway_existence', 'walkable_retail', 'walkable_bus_stop', 'walkable_entertainment',
'walkable_relaxation', 'streets_have_walkways', 'traffic_discourages_walking',
            'crime_discourages_walking', 'animals_discourage_walking', 'cigarette_even_once', 'cigar
_even_once', 'pipe_even_once', 'smokeless_even_once',
            'had_genetic_counseling', 'genetic_counseling_with_MD', 'genetic_counseling_for_cancer'}

replacements1 = {
  2: 0,
  3: 0,
  4: 0,
  5: 0,
  6: 0,
  7: 0,
  8: 0,
  9: 0
}

df_smoking2 = df_smoking1.loc[:, features1].replace(replacements1)

df_smoking2.head()
```

Out[4]:

	genetic_counseling_with_MD	cigar_even_once	pipe_even_once	had_genetic_counseling	calcium_past_month	walkway_ex
0	0	1	0	0	0.0	
1	0	0	0	0	1.0	
2	0	1	0	0	0.0	
3	0	1	0	0	0.0	
4	0	0	0	0	0.0	

5 rows × 23 columns

In [5]:
```
df_smoking1.loc[:,'number'] = df_smoking1.index
df_smoking2.loc[:,'number'] = df_smoking2.index

df_smoking1.loc[df_smoking1.number.isin(df_smoking2.number), features1] = df_smoking2.loc[:,features
1]
df_smoking1.head()
```

Out[5]:

	language	cereal_serve_per_month	cereal_times_per_month	more_than_one_cereal_type	milk_serve_per_month	milk_times
0	5	3	2	0.0	3	
1	4	0	0	0.0	0	
2	5	5	2	0.0	5	
3	3	1	1	0.0	4	
4	5	2	2	1.0	0	

5 rows × 93 columns

In [6]:
```
df_smoking1 = df_smoking1.drop('number', axis = 1)
df_smoking1.head()
```

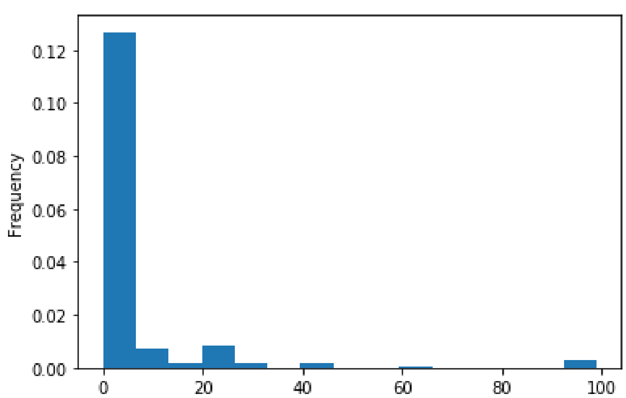
Out[6]:

	language	cereal_serve_per_month	cereal_times_per_month	more_than_one_cereal_type	milk_serve_per_month	milk_times
0	5	3	2	0.0	3	
1	4	0	0	0.0	0	
2	5	5	2	0.0	5	
3	3	1	1	0.0	4	
4	5	2	2	1.0	0	

5 rows × 92 columns

In [7]:
```
# Freqeuncy plot for cigarettes_per_day
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline

d = df_smoking1['cigarettes_per_day']
plt.hist(df_smoking1['cigarettes_per_day'], normed=True, bins=15)
plt.ylabel('Frequency');
```

```
C:\Users\ASG\.conda\envs\Lambda\lib\site-packages\ipykernel_launcher.py:7: MatplotlibDeprecationWarn
ing:
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed in 3.1. Use 'density' instea
d.
  import sys
```

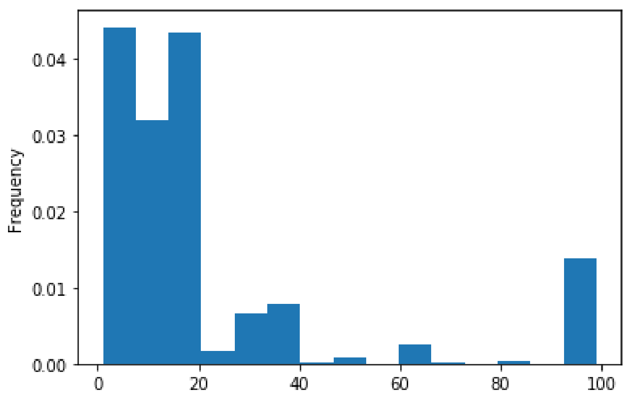


In [8]:
```
# Drop rows where cigarettes_per_day = 0
df_smoking1['cigarettes_per_day'] =  df_smoking1['cigarettes_per_day'].replace ({np.NaN: 0})
df_smoking1 = df_smoking1[df_smoking1['cigarettes_per_day'] > 0]
df_smoking1.shape
```

Out[8]: (7602, 92)

In [9]:
```
# Create frequency plot of cigarettes per day
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline

d = df_smoking1['cigarettes_per_day']
plt.hist(df_smoking1['cigarettes_per_day'], normed=True, bins=15)
plt.ylabel('Frequency');
```

```
C:\Users\ASG\.conda\envs\Lambda\lib\site-packages\ipykernel_launcher.py:7: MatplotlibDeprecationWarn
ing:
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed in 3.1. Use 'density' instea
d.
  import sys
```



In [10]:
```
# Create a column in which cigarettes per day are sorted into 8 bins
df_smoking1['cigarettes_per_day_bins'] = pd.cut(x=df_smoking1['cigarettes_per_day'], bins=[0, 7, 14,
21, 28, 35, 42, 49, 100], labels=[1, 2, 3, 4, 5, 6, 7, 8])
df_smoking1 = df_smoking1.drop('cigarettes_per_day', axis = 1)
df_smoking1['cigarettes_per_day_bins'] =  df_smoking1['cigarettes_per_day_bins'].replace ({np.NaN: 0
})
df_smoking1.head()
```

Out[10]:

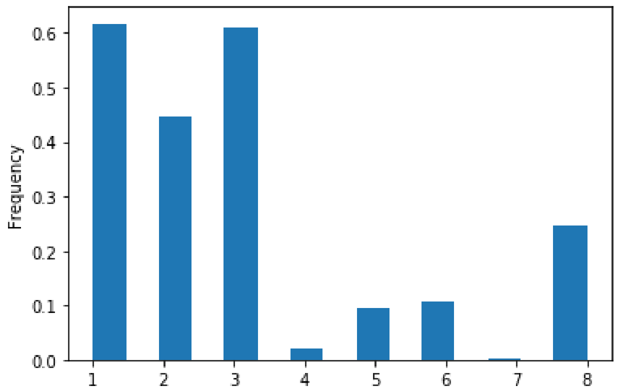
	language	cereal_serve_per_month	cereal_times_per_month	more_than_one_cereal_type	milk_serve_per_month	milk_time
4	5	2	2	1.0	0	
9	1	3	2	0.0	1	
11	5	0	0	0.0	0	
13	5	0	0	0.0	0	
14	2	0	0	0.0	0	

5 rows × 92 columns

In [11]:
```
# Looking at the frequency distribution of cigarettes per day bins
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline

d_bin = df_smoking1['cigarettes_per_day_bins']
plt.hist(d_bin, normed=True, bins=15)
plt.ylabel('Frequency')
```

```
C:\Users\ASG\.conda\envs\Lambda\lib\site-packages\ipykernel_launcher.py:7: MatplotlibDeprecationWarn
ing:
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed in 3.1. Use 'density' instea
d.
  import sys
```

Out[11]: `Text(0, 0.5, 'Frequency')`



In [12]:
```
# Train/validate split: random 80/20% train/validate split.
from sklearn.model_selection import train_test_split
XTrain, XVal, yTrain, yVal = train_test_split(df_smoking1.drop('cigarettes_per_day_bins', axis = 1),
df_smoking1['cigarettes_per_day_bins'], test_size = 0.2, random_state = 42)

XTrain.shape, yTrain.shape, XVal.shape, yVal.shape
```

Out[12]: `((6081, 91), (6081,), (1521, 91), (1521,))`

In [13]:
```
# Look at correlation coefficients
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', 1000)
XTrain.corr()
```

Out[13]:

	language	cereal_serve_per_month	cereal_times_per_month	more_than_one_cereal_typ
language	1.000000	0.436982	0.351576	-0.03536
cereal_serve_per_month	0.436982	1.000000	0.760684	-0.13857
cereal_times_per_month	0.351576	0.760684	1.000000	0.10388
more_than_one_cereal_type	-0.035361	-0.138573	0.103886	1.00000
milk_serve_per_month	0.433675	0.972695	0.735602	-0.13927
milk_times_per_month	0.349838	0.769347	0.739144	-0.00663
milk_type	-0.096036	-0.232121	-0.007578	0.21999
soda_serve_per_month	0.431958	0.959336	0.721514	-0.14112
soda_times_per_month	0.342304	0.734191	0.595590	-0.05803
juice_serve_per_month	0.428804	0.956040	0.720313	-0.14009
juice_times_per_month	0.332304	0.727421	0.597924	-0.06669
coffee_serve_per_month	0.426747	0.951691	0.714146	-0.14404
coffee_times_per_month	0.333119	0.801064	0.622032	-0.10484
sports_drink_serve_per_month	0.432197	0.957457	0.718121	-0.14344
sports_drink_times_per_month	0.359200	0.808602	0.625149	-0.12328
fruit_drink_serve_per_month	0.431355	0.952001	0.713791	-0.14221
fruit_drink_times_per_month	0.358626	0.798001	0.620712	-0.09332
fruit_eat_serve_per_month	0.425964	0.957833	0.721305	-0.14257
fruit_eat_times_per_month	0.384347	0.806646	0.658352	-0.10115
salad_eat_serve_per_month	0.427673	0.950363	0.713253	-0.14131
salad_eat_times_per_month	0.382662	0.789765	0.644858	-0.08688
fries_eat_serve_per_month	0.425416	0.950622	0.710713	-0.14318
fries_eat_times_per_month	0.361141	0.706499	0.579918	-0.06054
potatoe_eat_serve_per_month	0.422435	0.936681	0.699211	-0.14165
potatoe_eat_times_per_month	0.375218	0.743602	0.606211	-0.06284
beans_eat_serve_per_month	0.421520	0.935026	0.698968	-0.14242
beans_eat_times_per_month	0.334060	0.704172	0.577761	-0.04843
grains_eat_serve_per_month	0.422670	0.940141	0.701947	-0.14224
grains_eat_times_per_month	0.352108	0.698946	0.547232	-0.08389
vegies_eat_serve_per_month	0.415677	0.928090	0.693861	-0.14227
vegies_eat_times_per_month	0.359752	0.801514	0.632530	-0.11232
salsa_eat_serve_per_month	0.421930	0.932706	0.695506	-0.13871
salsa_eat_times_per_month	0.332938	0.678452	0.541066	-0.07792
pizza_eat_serve_per_month	0.422585	0.938145	0.699300	-0.14168
pizza_eat_times_per_month	0.358019	0.679303	0.546140	-0.06543
tomatoe_eat_serve_per_month	0.418889	0.930008	0.692785	-0.13838
tomatoe_eat_times_per_month	0.360487	0.700663	0.569326	-0.04879
cheese_eat_serve_per_month	0.417031	0.926477	0.691735	-0.13893
cheese_eat_times_per_month	0.363737	0.769202	0.610668	-0.09293
red_meat_eat_serve_per_month	0.419657	0.929806	0.694151	-0.14063
red_meat_eat_times_per_month	0.376608	0.780559	0.615793	-0.09470
processed_meat_eat_serve_per_month	0.418972	0.928255	0.692179	-0.13983
processed_meat_eat_times_per_month	0.373554	0.707912	0.571415	-0.05094
bread_eat_serve_per_month	0.417267	0.923150	0.689785	-0.13842

	language	cereal_serve_per_month	cereal_times_per_month	more_than_one_cereal_type
bread_eat_times_per_month	0.339279	0.735331	0.595573	-0.089927
candy_eat_serve_per_month	0.411998	0.922073	0.689743	-0.138842
candy_eat_times_per_month	0.372756	0.707072	0.583550	-0.067669
donut_eat_serve_per_month	0.416284	0.926723	0.690687	-0.138930
donut_eat_times_per_month	0.334741	0.680731	0.556009	-0.040507
cookie_eat_serve_per_month	0.409480	0.912101	0.677290	-0.139671
cookie_eat_times_per_month	0.355908	0.682247	0.559441	-0.049372
ice_cream_eat_serve_per_month	0.414443	0.918537	0.683445	-0.141529
ice_cream_eat_times_per_month	0.350857	0.677407	0.552084	-0.047244
pop_corn_eat_serve_per_month	0.415217	0.921843	0.687277	-0.141857
pop_corn_eat_times_per_month	0.354492	0.669004	0.529327	-0.059074
vitamin_past_month	-0.050629	-0.243404	-0.157238	0.075255
multivitamin_past_month	-0.037872	-0.162842	-0.096123	0.044575
multivitamin_days_in_month	-0.029406	-0.150437	-0.089361	0.038086
calcium_past_month	-0.040267	-0.096730	-0.061498	0.042639
calcium_days_in_month	-0.034379	-0.086469	-0.060933	0.033369
vitD_past_month	-0.016192	-0.122617	-0.076643	0.052465
vitD_days_in_month	-0.013972	-0.111407	-0.068578	0.045439
vitD_reason	-0.011984	-0.099275	-0.061147	0.056433
1st_kind_cereal_eaten	-0.066491	-0.213615	0.202229	0.241560
2nd_kind_cereal_eaten	-0.021112	-0.118378	0.093967	0.855270
walk_past_wk	-0.100718	-0.114823	-0.085251	0.009630
walk_number_wk	-0.049873	-0.039521	-0.041604	-0.011125
single_walk_distance	-0.015167	-0.034909	-0.037080	-0.020225
single_walk_time	-0.075258	-0.097345	-0.084728	-0.007337
walk_leisure_past_wk	-0.077325	-0.188538	-0.135776	0.048583
walk_leisure_number_wk	-0.026543	-0.105001	-0.087298	0.017029
walk_leisure_ distance	-0.026035	-0.067584	-0.044969	0.000541
walk_leisure_ time	-0.061651	-0.163052	-0.120797	0.029078
see_walking_from_home	0.322965	0.612504	0.441254	-0.099534
weather_discourages_walk	0.214795	0.481079	0.334835	-0.107099
walkway_existence	-0.203418	-0.385381	-0.283120	0.059284
walkable_retail	-0.159764	-0.199678	-0.134860	0.023874
walkable_bus_stop	-0.188837	-0.181334	-0.142217	0.016855
walkable_entertainment	-0.150265	-0.176244	-0.124428	0.016642
walkable_relaxation	-0.141028	-0.274859	-0.193180	0.041037
streets_have_walkways	-0.188904	-0.217642	-0.159778	-0.006564
traffic_discourages_walking	-0.093775	-0.097254	-0.076176	0.063229
crime_discourages_walking	-0.096958	-0.069252	-0.066612	0.031619
animals_discourage_walking	-0.069518	-0.061819	-0.047632	0.046416
cigarette_even_once	-0.014661	-0.082766	-0.060123	0.005500
cigar_even_once	0.017100	-0.156603	-0.099829	0.082745
pipe_even_once	0.021861	-0.104214	-0.052365	0.084021
smokeless_even_once	0.036964	-0.087348	-0.057695	0.028397

	language	cereal_serve_per_month	cereal_times_per_month	more_than_one_cereal_typ
had_genetic_counseling	-0.011091	-0.026606	-0.011029	0.02821
genetic_counseling_with_MD	-0.021622	-0.039074	-0.013490	0.02999
genetic_counseling_for_cancer	-0.015048	-0.023971	-0.022560	0.02218

In [14]:

```
# Dropping highly corrlated columns
def correlation(dataset, validation_dataset, threshold):
    col_corr = set() # Set of all the names of deleted columns
    corr_matrix = dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if (corr_matrix.iloc[i, j] >= threshold) and (corr_matrix.columns[j] not in col_corr):
                colname = corr_matrix.columns[i] # getting the name of column
                col_corr.add(colname)
                if colname in dataset.columns:
                    del dataset[colname] # deleting the column from the dataset
                    del validation_dataset[colname] # deleting the column from the validation datase
t

correlation(XTrain, XVal, 0.98)

XTrain.shape
XVal.shape
```

Out[14]:

```
(1521, 78)
```

In [15]:

```
# Begin with baselines for classification.
# The baseline accuracy, if the majority class is guessed for every prediction?
# option with pandas function:
yTrain.value_counts(normalize=True)
```

Out[15]:

```
3    0.286466
1    0.285644
2    0.208847
8    0.113633
6    0.049663
5    0.044565
4    0.009702
7    0.001480
Name: cigarettes_per_day_bins, dtype: float64
```

In [16]:

```
# option with scikit-learn function
from sklearn.metrics import accuracy_score
y = yTrain
majority_class = y.mode()[0]
y_pred = [majority_class] * len(y)
accuracy_score(y, y_pred)
```

Out[16]:

```
0.2864660417694458
```

In [17]:

```
# Thus, baseline accuracy, if you guessed the majority class for every prediction is 0.286
```

In [72]:
```
# Optimizing Hyperparameters
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier

# Define classifier
forest = RandomForestClassifier(random_state = 1)

# Input
X_train = XTrain
y_train = yTrain
X_val = XVal
y_val = yVal

# Parameters to fit
n_estimators = [5, 10, 45, 46, 152, 205, 358, 393, 1000]
max_depth = [3, 5, 7, 10, 15]
min_samples_split = [2, 5, 10]
min_samples_leaf = [1, 5, 10, 15]
max_leaf_nodes = [None, 10, 52]
max_features = [0.11373956383989692, 0.14621091571560108, 0.17046743865886782, 0.17281968473284381,
0.5545636480509806, 0.6130788778718701, 0.6216883421110927, 0.6843610478580876, 0.840823]

hyperF = dict(n_estimators = n_estimators, max_depth = max_depth,
              min_samples_split = min_samples_split,
             min_samples_leaf = min_samples_leaf,
             max_leaf_nodes = max_leaf_nodes,
             max_features = max_features)

gridF = GridSearchCV(forest, hyperF, cv = 3, verbose = 10,
                     scoring='accuracy', return_train_score=True,
                     n_jobs = -1)
bestF = gridF.fit(X_train, y_train)
```

```
Fitting 3 folds for each of 14580 candidates, totalling 43740 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done   5 tasks      | elapsed:    4.1s
[Parallel(n_jobs=-1)]: Done  10 tasks      | elapsed:    4.6s
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    7.0s
[Parallel(n_jobs=-1)]: Done  24 tasks      | elapsed:   10.5s
[Parallel(n_jobs=-1)]: Done  33 tasks      | elapsed:   11.6s
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:   14.6s
[Parallel(n_jobs=-1)]: Done  53 tasks      | elapsed:   18.1s
[Parallel(n_jobs=-1)]: Done  64 tasks      | elapsed:   20.9s
[Parallel(n_jobs=-1)]: Done  77 tasks      | elapsed:   26.4s
[Parallel(n_jobs=-1)]: Done  90 tasks      | elapsed:   28.4s
[Parallel(n_jobs=-1)]: Done 105 tasks      | elapsed:   34.2s
[Parallel(n_jobs=-1)]: Done 120 tasks      | elapsed:   37.7s
[Parallel(n_jobs=-1)]: Done 137 tasks      | elapsed:   43.3s
[Parallel(n_jobs=-1)]: Done 154 tasks      | elapsed:   48.8s
[Parallel(n_jobs=-1)]: Done 173 tasks      | elapsed:   54.1s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:  1.0min
[Parallel(n_jobs=-1)]: Done 213 tasks      | elapsed:  1.1min
[Parallel(n_jobs=-1)]: Done 234 tasks      | elapsed:  1.2min
[Parallel(n_jobs=-1)]: Done 257 tasks      | elapsed:  1.3min
[Parallel(n_jobs=-1)]: Done 280 tasks      | elapsed:  1.4min
[Parallel(n_jobs=-1)]: Done 305 tasks      | elapsed:  1.6min
[Parallel(n_jobs=-1)]: Done 330 tasks      | elapsed:  1.7min
[Parallel(n_jobs=-1)]: Done 357 tasks      | elapsed:  1.8min
[Parallel(n_jobs=-1)]: Done 384 tasks      | elapsed:  2.0min
[Parallel(n_jobs=-1)]: Done 413 tasks      | elapsed:  2.1min
[Parallel(n_jobs=-1)]: Done 442 tasks      | elapsed:  2.3min
[Parallel(n_jobs=-1)]: Done 473 tasks      | elapsed:  2.4min
[Parallel(n_jobs=-1)]: Done 504 tasks      | elapsed:  2.6min
[Parallel(n_jobs=-1)]: Done 537 tasks      | elapsed:  2.8min
[Parallel(n_jobs=-1)]: Done 570 tasks      | elapsed:  2.9min
[Parallel(n_jobs=-1)]: Done 605 tasks      | elapsed:  3.1min
[Parallel(n_jobs=-1)]: Done 640 tasks      | elapsed:  3.3min
[Parallel(n_jobs=-1)]: Done 677 tasks      | elapsed:  3.5min
[Parallel(n_jobs=-1)]: Done 714 tasks      | elapsed:  3.7min
[Parallel(n_jobs=-1)]: Done 753 tasks      | elapsed:  3.9min
[Parallel(n_jobs=-1)]: Done 792 tasks      | elapsed:  4.1min
[Parallel(n_jobs=-1)]: Done 833 tasks      | elapsed:  4.3min
[Parallel(n_jobs=-1)]: Done 874 tasks      | elapsed:  4.5min
[Parallel(n_jobs=-1)]: Done 917 tasks      | elapsed:  4.8min
[Parallel(n_jobs=-1)]: Done 960 tasks      | elapsed:  5.0min
[Parallel(n_jobs=-1)]: Done 1005 tasks      | elapsed:  5.3min
[Parallel(n_jobs=-1)]: Done 1050 tasks      | elapsed:  5.6min
[Parallel(n_jobs=-1)]: Done 1097 tasks      | elapsed:  5.8min
[Parallel(n_jobs=-1)]: Done 1144 tasks      | elapsed:  6.1min
[Parallel(n_jobs=-1)]: Done 1193 tasks      | elapsed:  6.4min
[Parallel(n_jobs=-1)]: Done 1242 tasks      | elapsed:  6.7min
[Parallel(n_jobs=-1)]: Done 1293 tasks      | elapsed:  7.0min
[Parallel(n_jobs=-1)]: Done 1344 tasks      | elapsed:  7.3min
[Parallel(n_jobs=-1)]: Done 1397 tasks      | elapsed:  7.7min
[Parallel(n_jobs=-1)]: Done 1450 tasks      | elapsed:  8.0min
[Parallel(n_jobs=-1)]: Done 1505 tasks      | elapsed:  8.3min
[Parallel(n_jobs=-1)]: Done 1560 tasks      | elapsed:  8.7min
[Parallel(n_jobs=-1)]: Done 1617 tasks      | elapsed:  9.0min
[Parallel(n_jobs=-1)]: Done 1674 tasks      | elapsed:  9.3min
[Parallel(n_jobs=-1)]: Done 1733 tasks      | elapsed:  9.7min
[Parallel(n_jobs=-1)]: Done 1792 tasks      | elapsed: 10.1min
[Parallel(n_jobs=-1)]: Done 1853 tasks      | elapsed: 10.4min
[Parallel(n_jobs=-1)]: Done 1914 tasks      | elapsed: 10.8min
[Parallel(n_jobs=-1)]: Done 1977 tasks      | elapsed: 11.3min
[Parallel(n_jobs=-1)]: Done 2040 tasks      | elapsed: 11.7min
[Parallel(n_jobs=-1)]: Done 2105 tasks      | elapsed: 12.1min
[Parallel(n_jobs=-1)]: Done 2170 tasks      | elapsed: 12.6min
[Parallel(n_jobs=-1)]: Done 2237 tasks      | elapsed: 13.0min
[Parallel(n_jobs=-1)]: Done 2304 tasks      | elapsed: 13.5min
[Parallel(n_jobs=-1)]: Done 2373 tasks      | elapsed: 14.0min
[Parallel(n_jobs=-1)]: Done 2442 tasks      | elapsed: 14.4min
[Parallel(n_jobs=-1)]: Done 2513 tasks      | elapsed: 14.9min
[Parallel(n_jobs=-1)]: Done 2584 tasks      | elapsed: 15.4min
[Parallel(n_jobs=-1)]: Done 2657 tasks      | elapsed: 15.9min
```

```
[Parallel(n_jobs=-1)]: Done 2730 tasks      | elapsed: 16.4min
[Parallel(n_jobs=-1)]: Done 2805 tasks      | elapsed: 17.0min
[Parallel(n_jobs=-1)]: Done 2880 tasks      | elapsed: 17.5min
[Parallel(n_jobs=-1)]: Done 2957 tasks      | elapsed: 18.0min
[Parallel(n_jobs=-1)]: Done 3034 tasks      | elapsed: 18.6min
[Parallel(n_jobs=-1)]: Done 3113 tasks      | elapsed: 19.1min
[Parallel(n_jobs=-1)]: Done 3192 tasks      | elapsed: 19.7min
[Parallel(n_jobs=-1)]: Done 3273 tasks      | elapsed: 20.3min
[Parallel(n_jobs=-1)]: Done 3354 tasks      | elapsed: 20.9min
[Parallel(n_jobs=-1)]: Done 3437 tasks      | elapsed: 21.5min
[Parallel(n_jobs=-1)]: Done 3520 tasks      | elapsed: 22.1min
[Parallel(n_jobs=-1)]: Done 3605 tasks      | elapsed: 22.8min
[Parallel(n_jobs=-1)]: Done 3690 tasks      | elapsed: 23.4min
[Parallel(n_jobs=-1)]: Done 3777 tasks      | elapsed: 24.1min
[Parallel(n_jobs=-1)]: Done 3864 tasks      | elapsed: 24.8min
[Parallel(n_jobs=-1)]: Done 3953 tasks      | elapsed: 26.0min
[Parallel(n_jobs=-1)]: Done 4042 tasks      | elapsed: 27.5min
[Parallel(n_jobs=-1)]: Done 4133 tasks      | elapsed: 28.8min
[Parallel(n_jobs=-1)]: Done 4224 tasks      | elapsed: 30.2min
[Parallel(n_jobs=-1)]: Done 4317 tasks      | elapsed: 31.7min
[Parallel(n_jobs=-1)]: Done 4410 tasks      | elapsed: 33.1min
[Parallel(n_jobs=-1)]: Done 4505 tasks      | elapsed: 34.7min
[Parallel(n_jobs=-1)]: Done 4600 tasks      | elapsed: 36.0min
[Parallel(n_jobs=-1)]: Done 4697 tasks      | elapsed: 37.7min
[Parallel(n_jobs=-1)]: Done 4794 tasks      | elapsed: 39.2min
[Parallel(n_jobs=-1)]: Done 4893 tasks      | elapsed: 40.8min
[Parallel(n_jobs=-1)]: Done 4992 tasks      | elapsed: 42.4min
[Parallel(n_jobs=-1)]: Done 5093 tasks      | elapsed: 43.9min
[Parallel(n_jobs=-1)]: Done 5194 tasks      | elapsed: 45.5min
[Parallel(n_jobs=-1)]: Done 5297 tasks      | elapsed: 47.2min
[Parallel(n_jobs=-1)]: Done 5400 tasks      | elapsed: 48.9min
[Parallel(n_jobs=-1)]: Done 5505 tasks      | elapsed: 50.6min
[Parallel(n_jobs=-1)]: Done 5610 tasks      | elapsed: 52.3min
[Parallel(n_jobs=-1)]: Done 5717 tasks      | elapsed: 54.0min
[Parallel(n_jobs=-1)]: Done 5824 tasks      | elapsed: 55.8min
[Parallel(n_jobs=-1)]: Done 5933 tasks      | elapsed: 57.5min
[Parallel(n_jobs=-1)]: Done 6042 tasks      | elapsed: 59.2min
[Parallel(n_jobs=-1)]: Done 6153 tasks      | elapsed: 61.0min
[Parallel(n_jobs=-1)]: Done 6264 tasks      | elapsed: 62.8min
[Parallel(n_jobs=-1)]: Done 6377 tasks      | elapsed: 64.7min
[Parallel(n_jobs=-1)]: Done 6490 tasks      | elapsed: 66.5min
[Parallel(n_jobs=-1)]: Done 6605 tasks      | elapsed: 68.5min
[Parallel(n_jobs=-1)]: Done 6720 tasks      | elapsed: 70.5min
[Parallel(n_jobs=-1)]: Done 6837 tasks      | elapsed: 72.2min
[Parallel(n_jobs=-1)]: Done 6954 tasks      | elapsed: 74.3min
[Parallel(n_jobs=-1)]: Done 7073 tasks      | elapsed: 76.5min
[Parallel(n_jobs=-1)]: Done 7192 tasks      | elapsed: 78.7min
[Parallel(n_jobs=-1)]: Done 7313 tasks      | elapsed: 80.9min
[Parallel(n_jobs=-1)]: Done 7434 tasks      | elapsed: 82.9min
[Parallel(n_jobs=-1)]: Done 7557 tasks      | elapsed: 85.2min
[Parallel(n_jobs=-1)]: Done 7680 tasks      | elapsed: 87.3min
[Parallel(n_jobs=-1)]: Done 7805 tasks      | elapsed: 89.5min
[Parallel(n_jobs=-1)]: Done 7930 tasks      | elapsed: 92.0min
[Parallel(n_jobs=-1)]: Done 8057 tasks      | elapsed: 94.7min
[Parallel(n_jobs=-1)]: Done 8184 tasks      | elapsed: 97.3min
[Parallel(n_jobs=-1)]: Done 8313 tasks      | elapsed: 101.3min
[Parallel(n_jobs=-1)]: Done 8442 tasks      | elapsed: 104.3min
[Parallel(n_jobs=-1)]: Done 8573 tasks      | elapsed: 107.5min
[Parallel(n_jobs=-1)]: Done 8704 tasks      | elapsed: 110.6min
[Parallel(n_jobs=-1)]: Done 8837 tasks      | elapsed: 112.6min
[Parallel(n_jobs=-1)]: Done 8970 tasks      | elapsed: 113.8min
[Parallel(n_jobs=-1)]: Done 9105 tasks      | elapsed: 115.0min
[Parallel(n_jobs=-1)]: Done 9240 tasks      | elapsed: 116.2min
[Parallel(n_jobs=-1)]: Done 9377 tasks      | elapsed: 117.4min
[Parallel(n_jobs=-1)]: Done 9514 tasks      | elapsed: 118.5min
[Parallel(n_jobs=-1)]: Done 9653 tasks      | elapsed: 119.8min
[Parallel(n_jobs=-1)]: Done 9792 tasks      | elapsed: 121.1min
[Parallel(n_jobs=-1)]: Done 9933 tasks      | elapsed: 122.6min
[Parallel(n_jobs=-1)]: Done 10074 tasks      | elapsed: 124.0min
[Parallel(n_jobs=-1)]: Done 10217 tasks      | elapsed: 125.5min
[Parallel(n_jobs=-1)]: Done 10360 tasks      | elapsed: 126.9min
```

```
[Parallel(n_jobs=-1)]: Done 10505 tasks      | elapsed: 128.5min
[Parallel(n_jobs=-1)]: Done 10650 tasks      | elapsed: 130.0min
[Parallel(n_jobs=-1)]: Done 10797 tasks      | elapsed: 131.6min
[Parallel(n_jobs=-1)]: Done 10944 tasks      | elapsed: 133.2min
[Parallel(n_jobs=-1)]: Done 11093 tasks      | elapsed: 134.9min
[Parallel(n_jobs=-1)]: Done 11242 tasks      | elapsed: 136.5min
[Parallel(n_jobs=-1)]: Done 11393 tasks      | elapsed: 138.1min
[Parallel(n_jobs=-1)]: Done 11544 tasks      | elapsed: 139.7min
[Parallel(n_jobs=-1)]: Done 11697 tasks      | elapsed: 141.4min
[Parallel(n_jobs=-1)]: Done 11850 tasks      | elapsed: 143.2min
[Parallel(n_jobs=-1)]: Done 12005 tasks      | elapsed: 144.9min
[Parallel(n_jobs=-1)]: Done 12160 tasks      | elapsed: 146.5min
[Parallel(n_jobs=-1)]: Done 12317 tasks      | elapsed: 148.2min
[Parallel(n_jobs=-1)]: Done 12474 tasks      | elapsed: 150.0min
[Parallel(n_jobs=-1)]: Done 12633 tasks      | elapsed: 151.7min
[Parallel(n_jobs=-1)]: Done 12792 tasks      | elapsed: 155.5min
[Parallel(n_jobs=-1)]: Done 12953 tasks      | elapsed: 159.7min
[Parallel(n_jobs=-1)]: Done 13114 tasks      | elapsed: 163.9min
[Parallel(n_jobs=-1)]: Done 13277 tasks      | elapsed: 168.0min
[Parallel(n_jobs=-1)]: Done 13440 tasks      | elapsed: 172.0min
[Parallel(n_jobs=-1)]: Done 13605 tasks      | elapsed: 176.2min
[Parallel(n_jobs=-1)]: Done 13770 tasks      | elapsed: 180.6min
[Parallel(n_jobs=-1)]: Done 13937 tasks      | elapsed: 184.8min
[Parallel(n_jobs=-1)]: Done 14104 tasks      | elapsed: 189.1min
[Parallel(n_jobs=-1)]: Done 14273 tasks      | elapsed: 193.5min
[Parallel(n_jobs=-1)]: Done 14442 tasks      | elapsed: 198.1min
[Parallel(n_jobs=-1)]: Done 14613 tasks      | elapsed: 202.5min
[Parallel(n_jobs=-1)]: Done 14784 tasks      | elapsed: 207.2min
[Parallel(n_jobs=-1)]: Done 14957 tasks      | elapsed: 212.0min
[Parallel(n_jobs=-1)]: Done 15130 tasks      | elapsed: 216.6min
[Parallel(n_jobs=-1)]: Done 15305 tasks      | elapsed: 221.9min
[Parallel(n_jobs=-1)]: Done 15480 tasks      | elapsed: 226.6min
[Parallel(n_jobs=-1)]: Done 15657 tasks      | elapsed: 232.0min
[Parallel(n_jobs=-1)]: Done 15834 tasks      | elapsed: 237.0min
[Parallel(n_jobs=-1)]: Done 16013 tasks      | elapsed: 241.7min
[Parallel(n_jobs=-1)]: Done 16192 tasks      | elapsed: 246.7min
[Parallel(n_jobs=-1)]: Done 16373 tasks      | elapsed: 251.7min
[Parallel(n_jobs=-1)]: Done 16554 tasks      | elapsed: 256.6min
[Parallel(n_jobs=-1)]: Done 16737 tasks      | elapsed: 262.8min
[Parallel(n_jobs=-1)]: Done 16920 tasks      | elapsed: 268.2min
[Parallel(n_jobs=-1)]: Done 17105 tasks      | elapsed: 273.7min
[Parallel(n_jobs=-1)]: Done 17290 tasks      | elapsed: 279.6min
[Parallel(n_jobs=-1)]: Done 17477 tasks      | elapsed: 285.4min
[Parallel(n_jobs=-1)]: Done 17664 tasks      | elapsed: 288.0min
[Parallel(n_jobs=-1)]: Done 17853 tasks      | elapsed: 289.9min
[Parallel(n_jobs=-1)]: Done 18042 tasks      | elapsed: 291.4min
[Parallel(n_jobs=-1)]: Done 18233 tasks      | elapsed: 293.2min
[Parallel(n_jobs=-1)]: Done 18424 tasks      | elapsed: 295.1min
[Parallel(n_jobs=-1)]: Done 18617 tasks      | elapsed: 297.4min
[Parallel(n_jobs=-1)]: Done 18810 tasks      | elapsed: 299.7min
[Parallel(n_jobs=-1)]: Done 19005 tasks      | elapsed: 301.7min
[Parallel(n_jobs=-1)]: Done 19200 tasks      | elapsed: 303.8min
[Parallel(n_jobs=-1)]: Done 19397 tasks      | elapsed: 306.4min
[Parallel(n_jobs=-1)]: Done 19594 tasks      | elapsed: 309.1min
[Parallel(n_jobs=-1)]: Done 19793 tasks      | elapsed: 311.8min
[Parallel(n_jobs=-1)]: Done 19992 tasks      | elapsed: 314.2min
[Parallel(n_jobs=-1)]: Done 20193 tasks      | elapsed: 316.9min
[Parallel(n_jobs=-1)]: Done 20394 tasks      | elapsed: 319.8min
[Parallel(n_jobs=-1)]: Done 20597 tasks      | elapsed: 322.9min
[Parallel(n_jobs=-1)]: Done 20800 tasks      | elapsed: 325.5min
[Parallel(n_jobs=-1)]: Done 21005 tasks      | elapsed: 328.0min
[Parallel(n_jobs=-1)]: Done 21210 tasks      | elapsed: 330.6min
[Parallel(n_jobs=-1)]: Done 21417 tasks      | elapsed: 333.9min
[Parallel(n_jobs=-1)]: Done 21624 tasks      | elapsed: 341.1min
[Parallel(n_jobs=-1)]: Done 21833 tasks      | elapsed: 347.0min
[Parallel(n_jobs=-1)]: Done 22042 tasks      | elapsed: 352.4min
[Parallel(n_jobs=-1)]: Done 22253 tasks      | elapsed: 359.2min
[Parallel(n_jobs=-1)]: Done 22464 tasks      | elapsed: 366.7min
[Parallel(n_jobs=-1)]: Done 22677 tasks      | elapsed: 374.4min
[Parallel(n_jobs=-1)]: Done 22890 tasks      | elapsed: 380.1min
[Parallel(n_jobs=-1)]: Done 23105 tasks      | elapsed: 386.7min
```

```
[Parallel(n_jobs=-1)]: Done 23320 tasks      | elapsed: 395.3min
[Parallel(n_jobs=-1)]: Done 23537 tasks      | elapsed: 404.1min
[Parallel(n_jobs=-1)]: Done 23754 tasks      | elapsed: 411.9min
[Parallel(n_jobs=-1)]: Done 23973 tasks      | elapsed: 418.8min
[Parallel(n_jobs=-1)]: Done 24192 tasks      | elapsed: 428.0min
[Parallel(n_jobs=-1)]: Done 24413 tasks      | elapsed: 438.4min
[Parallel(n_jobs=-1)]: Done 24634 tasks      | elapsed: 448.8min
[Parallel(n_jobs=-1)]: Done 24857 tasks      | elapsed: 456.1min
[Parallel(n_jobs=-1)]: Done 25080 tasks      | elapsed: 466.1min
[Parallel(n_jobs=-1)]: Done 25305 tasks      | elapsed: 475.6min
[Parallel(n_jobs=-1)]: Done 25530 tasks      | elapsed: 486.7min
[Parallel(n_jobs=-1)]: Done 25757 tasks      | elapsed: 496.1min
[Parallel(n_jobs=-1)]: Done 25984 tasks      | elapsed: 504.6min
[Parallel(n_jobs=-1)]: Done 26213 tasks      | elapsed: 516.1min
[Parallel(n_jobs=-1)]: Done 26442 tasks      | elapsed: 521.8min
[Parallel(n_jobs=-1)]: Done 26673 tasks      | elapsed: 524.8min
[Parallel(n_jobs=-1)]: Done 26904 tasks      | elapsed: 527.3min
[Parallel(n_jobs=-1)]: Done 27137 tasks      | elapsed: 530.7min
[Parallel(n_jobs=-1)]: Done 27370 tasks      | elapsed: 534.8min
[Parallel(n_jobs=-1)]: Done 27605 tasks      | elapsed: 538.7min
[Parallel(n_jobs=-1)]: Done 27840 tasks      | elapsed: 541.1min
[Parallel(n_jobs=-1)]: Done 28077 tasks      | elapsed: 544.5min
[Parallel(n_jobs=-1)]: Done 28314 tasks      | elapsed: 548.7min
[Parallel(n_jobs=-1)]: Done 28553 tasks      | elapsed: 552.8min
[Parallel(n_jobs=-1)]: Done 28792 tasks      | elapsed: 555.5min
[Parallel(n_jobs=-1)]: Done 29033 tasks      | elapsed: 558.9min
[Parallel(n_jobs=-1)]: Done 29274 tasks      | elapsed: 563.2min
[Parallel(n_jobs=-1)]: Done 29517 tasks      | elapsed: 567.4min
[Parallel(n_jobs=-1)]: Done 29760 tasks      | elapsed: 570.0min
[Parallel(n_jobs=-1)]: Done 30005 tasks      | elapsed: 573.6min
[Parallel(n_jobs=-1)]: Done 30250 tasks      | elapsed: 581.0min
[Parallel(n_jobs=-1)]: Done 30497 tasks      | elapsed: 591.4min
[Parallel(n_jobs=-1)]: Done 30744 tasks      | elapsed: 598.8min
[Parallel(n_jobs=-1)]: Done 30993 tasks      | elapsed: 609.9min
[Parallel(n_jobs=-1)]: Done 31242 tasks      | elapsed: 622.3min
[Parallel(n_jobs=-1)]: Done 31493 tasks      | elapsed: 634.0min
[Parallel(n_jobs=-1)]: Done 31744 tasks      | elapsed: 641.2min
[Parallel(n_jobs=-1)]: Done 31997 tasks      | elapsed: 652.4min
[Parallel(n_jobs=-1)]: Done 32250 tasks      | elapsed: 665.0min
[Parallel(n_jobs=-1)]: Done 32505 tasks      | elapsed: 675.6min
[Parallel(n_jobs=-1)]: Done 32760 tasks      | elapsed: 683.2min
[Parallel(n_jobs=-1)]: Done 33017 tasks      | elapsed: 694.8min
[Parallel(n_jobs=-1)]: Done 33274 tasks      | elapsed: 709.0min
[Parallel(n_jobs=-1)]: Done 33533 tasks      | elapsed: 719.6min
[Parallel(n_jobs=-1)]: Done 33792 tasks      | elapsed: 729.3min
[Parallel(n_jobs=-1)]: Done 34053 tasks      | elapsed: 742.5min
[Parallel(n_jobs=-1)]: Done 34314 tasks      | elapsed: 760.3min
[Parallel(n_jobs=-1)]: Done 34577 tasks      | elapsed: 770.9min
[Parallel(n_jobs=-1)]: Done 34840 tasks      | elapsed: 783.8min
[Parallel(n_jobs=-1)]: Done 35105 tasks      | elapsed: 795.6min
[Parallel(n_jobs=-1)]: Done 35370 tasks      | elapsed: 799.7min
[Parallel(n_jobs=-1)]: Done 35637 tasks      | elapsed: 802.2min
[Parallel(n_jobs=-1)]: Done 35904 tasks      | elapsed: 805.4min
[Parallel(n_jobs=-1)]: Done 36173 tasks      | elapsed: 810.9min
[Parallel(n_jobs=-1)]: Done 36442 tasks      | elapsed: 814.5min
[Parallel(n_jobs=-1)]: Done 36713 tasks      | elapsed: 817.6min
[Parallel(n_jobs=-1)]: Done 36984 tasks      | elapsed: 822.5min
[Parallel(n_jobs=-1)]: Done 37257 tasks      | elapsed: 828.7min
[Parallel(n_jobs=-1)]: Done 37530 tasks      | elapsed: 831.9min
[Parallel(n_jobs=-1)]: Done 37805 tasks      | elapsed: 835.9min
[Parallel(n_jobs=-1)]: Done 38080 tasks      | elapsed: 842.2min
[Parallel(n_jobs=-1)]: Done 38357 tasks      | elapsed: 846.9min
[Parallel(n_jobs=-1)]: Done 38634 tasks      | elapsed: 850.4min
[Parallel(n_jobs=-1)]: Done 38913 tasks      | elapsed: 856.0min
[Parallel(n_jobs=-1)]: Done 39192 tasks      | elapsed: 872.5min
[Parallel(n_jobs=-1)]: Done 39473 tasks      | elapsed: 880.4min
[Parallel(n_jobs=-1)]: Done 39754 tasks      | elapsed: 891.5min
[Parallel(n_jobs=-1)]: Done 40037 tasks      | elapsed: 908.5min
[Parallel(n_jobs=-1)]: Done 40320 tasks      | elapsed: 920.0min
[Parallel(n_jobs=-1)]: Done 40605 tasks      | elapsed: 929.5min
[Parallel(n_jobs=-1)]: Done 40890 tasks      | elapsed: 943.9min
```

```
[Parallel(n_jobs=-1)]: Done 41177 tasks      | elapsed: 960.6min
[Parallel(n_jobs=-1)]: Done 41464 tasks      | elapsed: 968.5min
[Parallel(n_jobs=-1)]: Done 41753 tasks      | elapsed: 981.4min
[Parallel(n_jobs=-1)]: Done 42042 tasks      | elapsed: 1001.5min
[Parallel(n_jobs=-1)]: Done 42333 tasks      | elapsed: 1012.8min
[Parallel(n_jobs=-1)]: Done 42624 tasks      | elapsed: 1024.7min
[Parallel(n_jobs=-1)]: Done 42917 tasks      | elapsed: 1046.0min
[Parallel(n_jobs=-1)]: Done 43210 tasks      | elapsed: 1063.2min
[Parallel(n_jobs=-1)]: Done 43505 tasks      | elapsed: 1075.1min
[Parallel(n_jobs=-1)]: Done 43740 out of 43740 | elapsed: 1092.0min finished
```

In [18]:

```
# Output best accuracy and best parameters
print('The accuracy achieved with the best parameters = ', gridF.best_score_, '\n')
print('The parameters are:', gridF.best_params_)
```

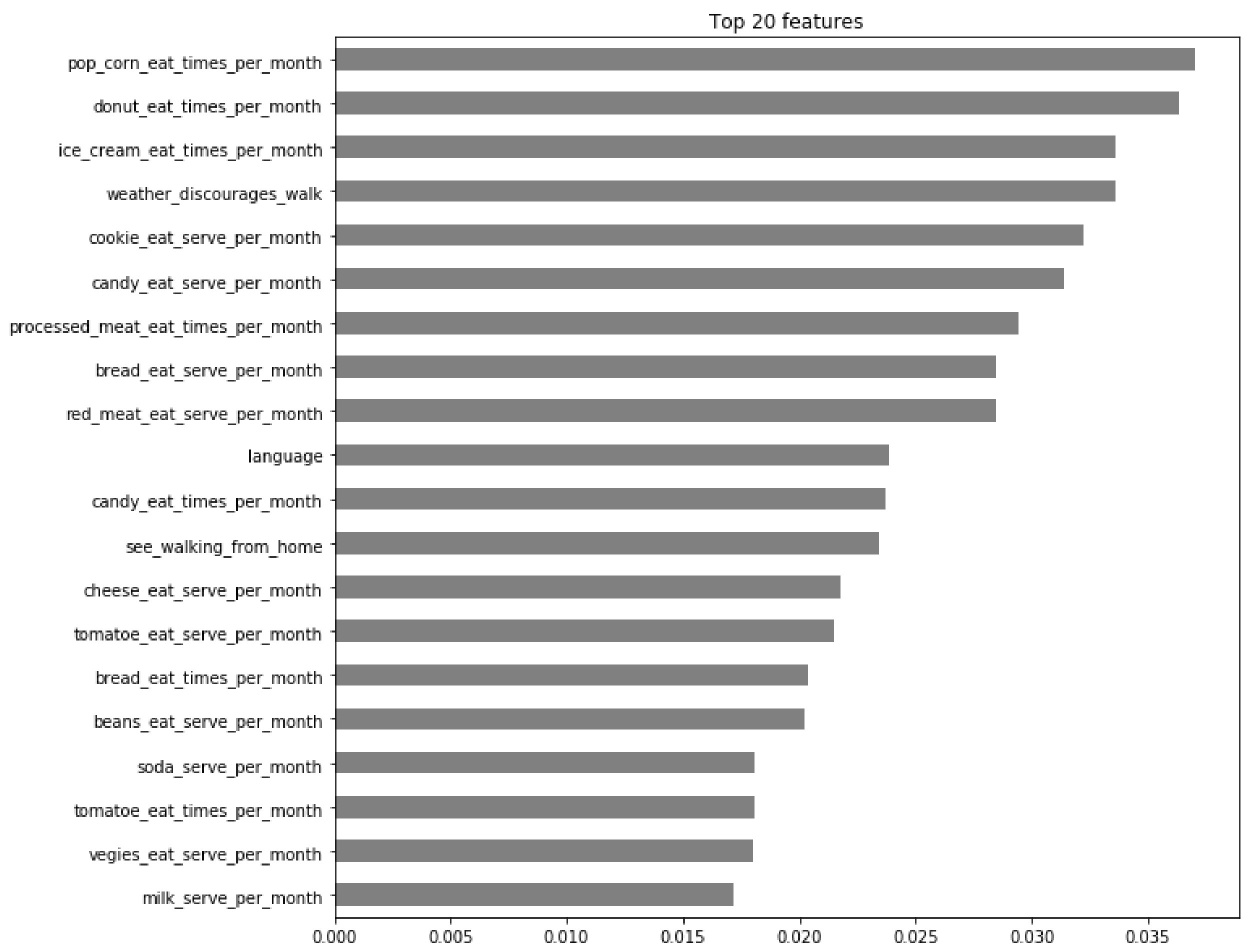
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-18-466135d70878> in <module>
      1 # Output best accuracy and best parameters
----> 2 print('The accuracy achieved with the best parameters = ', gridF.best_score_, '\n')
      3 print('The parameters are:', gridF.best_params_)

NameError: name 'gridF' is not defined
```

In [19]:

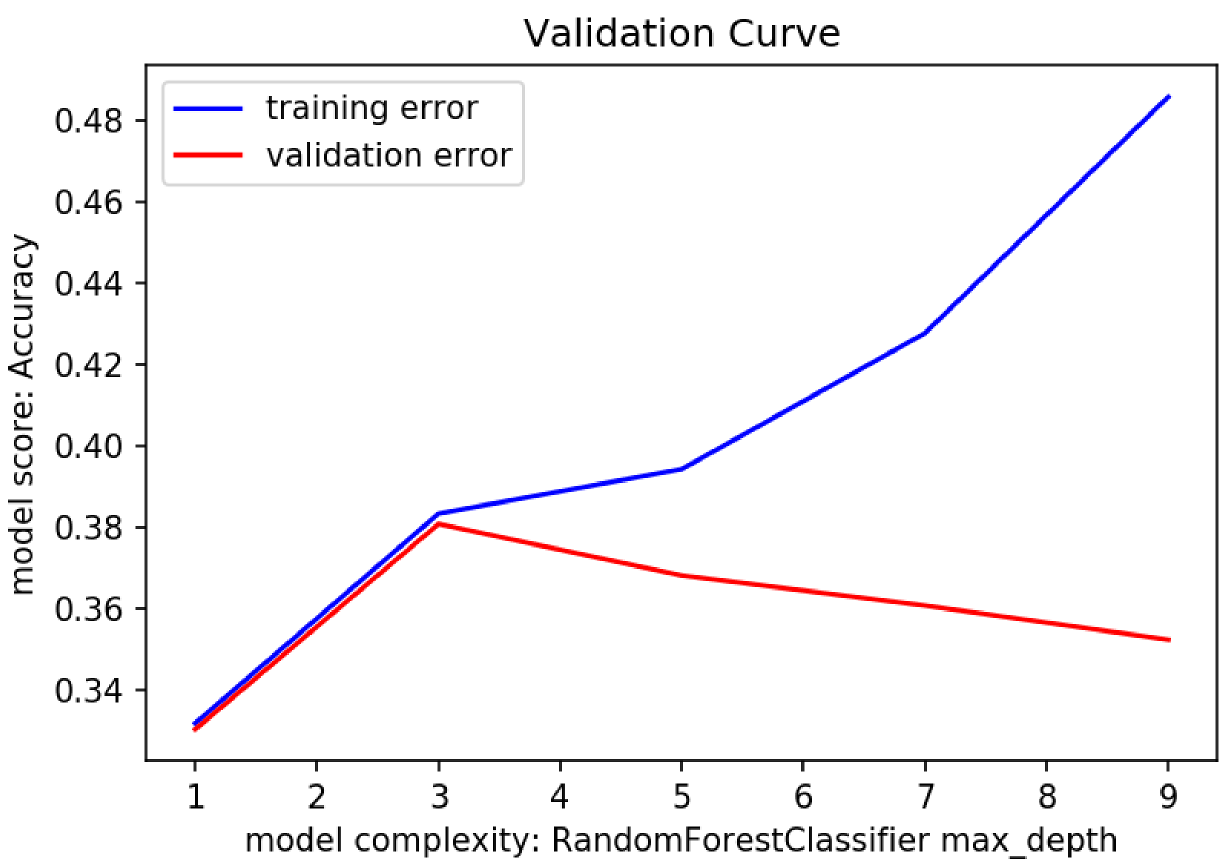
```
# Use a scikit-learn pipeline to encode categoricals and fit a Random Forest Classifier model.

X_train = XTrain
y_train = yTrain
X_val = XVal
y_val = yVal

from sklearn.pipeline import make_pipeline
import category_encoders as ce
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestClassifier

pipeline = make_pipeline(
    ce.OneHotEncoder(use_cat_names=True),
    SimpleImputer(strategy='mean'),
    RandomForestClassifier(random_state = 42, max_depth = 10,
                               max_features = 0.11373956383989692,
                               max_leaf_nodes = None,
                               min_samples_leaf = 1,
                               min_samples_split = 10,
                               n_estimators = 205))
pipeline.fit(X_train, y_train)

# Get the model's validation accuracy
ce.OneHotEncoder(use_cat_names=True),
print('Validation Accuracy', pipeline.score(X_val, y_val))
```

```
Validation Accuracy 0.398422090729783
```

In [20]:
```
# Plot of features
%matplotlib inline
import matplotlib.pyplot as plt
# Get feature importances
encoder = pipeline.named_steps['onehotencoder']
encoded = encoder.transform(X_train)
rf = pipeline.named_steps['randomforestclassifier']
importances1 = pd.Series(rf.feature_importances_, encoded.columns)
# Plot feature importances
n = 20
plt.figure(figsize=(10,n/2))
plt.title(f'Top {n} features')
importances1.sort_values()[-n:].plot.barh(color='grey');
```



In [21]:
```
# Generate validation curves
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import validation_curve
from sklearn.tree import DecisionTreeClassifier
pipeline = make_pipeline(
    ce.OrdinalEncoder(),
    SimpleImputer(),
    DecisionTreeClassifier()
)

depth = range(1, 10, 2)
train_scores, val_scores = validation_curve(
    pipeline, X_train, y_train,
    param_name='decisiontreeclassifier__max_depth',
    param_range=depth, scoring='accuracy',
    cv=3,
    n_jobs=-1
)

plt.figure(dpi=150)
plt.plot(depth, np.mean(train_scores, axis=1), color='blue', label='training error')
plt.plot(depth, np.mean(val_scores, axis=1), color='red', label='validation error')
plt.title('Validation Curve')
plt.xlabel('model complexity: RandomForestClassifier max_depth')
plt.ylabel('model score: Accuracy')
plt.legend();
```



In [22]:
```
# Tuning the hyper-parameters for a Random Forrest Classifier
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from scipy.stats import randint, uniform
from sklearn.pipeline import make_pipeline
import category_encoders as ce
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestClassifier

pipeline = make_pipeline(
    ce.OneHotEncoder(use_cat_names=True),
    SimpleImputer(),
    RandomForestClassifier(random_state = 42, max_depth = 10,
                           max_features = 0.11373956383989692,
                           max_leaf_nodes = None,
                           min_samples_leaf = 1,
                           min_samples_split = 10,
                           n_estimators = 205)
)

param_distributions = {'simpleimputer__strategy': ['mean', 'median', 'most_frequent']}
search = RandomizedSearchCV( pipeline, param_distributions=param_distributions, n_iter=10, cv=3, sco
ring='accuracy', verbose=10, return_train_score=True, n_jobs=-1)

search.fit(X_train, y_train);
```

```
C:\Users\ASG\.conda\envs\Lambda\lib\site-packages\sklearn\model_selection\_search.py:266: UserWarnin
g: The total space of parameters 3 is smaller than n_iter=10. Running 3 iterations. For exhaustive s
earches, use GridSearchCV.
  % (grid_size, self.n_iter, grid_size), UserWarning)
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.

Fitting 3 folds for each of 3 candidates, totalling 9 fits

[Parallel(n_jobs=-1)]: Done   3 out of   9 | elapsed:    2.3s remaining:    4.7s
[Parallel(n_jobs=-1)]: Done   4 out of   9 | elapsed:    2.4s remaining:    3.0s
[Parallel(n_jobs=-1)]: Done   5 out of   9 | elapsed:    5.0s remaining:    4.0s
[Parallel(n_jobs=-1)]: Done   6 out of   9 | elapsed:    5.1s remaining:    2.5s
[Parallel(n_jobs=-1)]: Done   7 out of   9 | elapsed:    5.1s remaining:    1.4s
[Parallel(n_jobs=-1)]: Done   9 out of   9 | elapsed:    7.0s remaining:    0.0s
[Parallel(n_jobs=-1)]: Done   9 out of   9 | elapsed:    7.0s finished
```

In [23]:
```
from sklearn.model_selection import cross_val_score
k = 3
scores = cross_val_score(pipeline, X_val, y_val, cv=k,
scoring='accuracy')
print(f'Validation Accuracy for {k} folds:', scores);
```

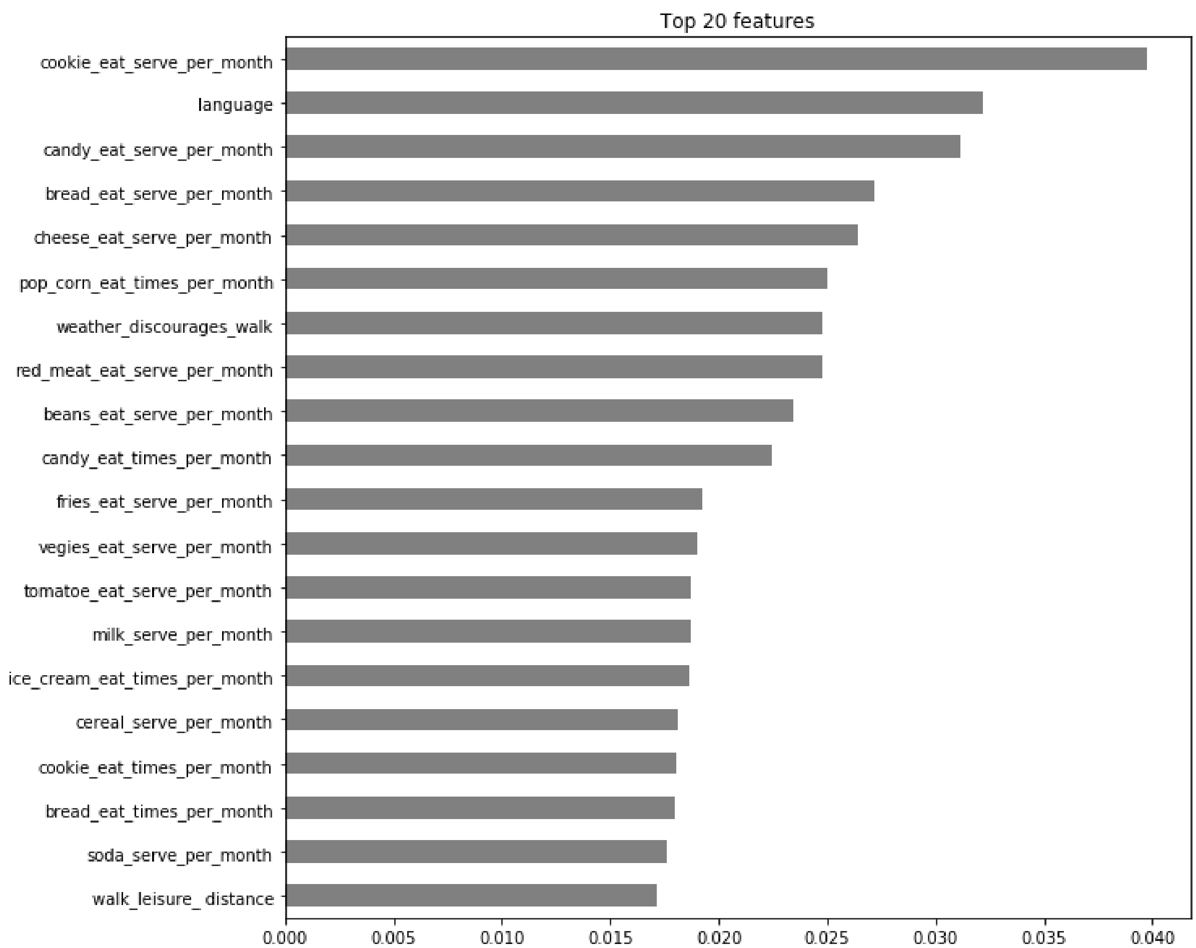
```
Validation Accuracy for 3 folds: [0.38235294 0.40631164 0.38690476]
```

In [24]:
```
print('Best hyperparameters', search.best_params_)
print('Cross-validation Accuracy', search.best_score_)
```

```
Best hyperparameters {'simpleimputer__strategy': 'mean'}
Cross-validation Accuracy 0.3945074823219865
```

In [25]:

```
pipeline.fit(X_val, y_val)
# Plot of features
%matplotlib inline
import matplotlib.pyplot as plt

# Get feature importances
encoder = pipeline.named_steps['onehotencoder']
encoded = encoder.transform(X_val)
rf = pipeline.named_steps['randomforestclassifier']
importances2 = pd.Series(rf.feature_importances_, encoded.columns)

# Plot feature importances
n = 20
plt.figure(figsize=(10,n/2))
plt.title(f'Top {n} features')
importances2.sort_values()[-n:].plot.barh(color='grey');
```



In [26]:
```
# Demonstrate the relatively high cardinatlity of candy_eat_times_per_month

XTrain['cookie_eat_serve_per_month'].value_counts()
```

Out[26]:
```
1      1730
0      1502
2      1138
3       507
4       265
998     254
5       185
10      120
15       62
7        58
6        57
20       45
8        33
997      32
30       23
999      20
12       17
25       14
18        5
14        4
9         3
203       1
13        1
28        1
24        1
22        1
16        1
31        1
Name: cookie_eat_serve_per_month, dtype: int64
```

In [27]:
```
# Get drop-column importances
column = 'cookie_eat_serve_per_month'

# # Fit without column
pipeline = make_pipeline(
    ce.OneHotEncoder(use_cat_names=True),
    SimpleImputer(strategy = 'mean'),
    RandomForestClassifier(random_state = 42, max_depth = 10,
                           max_features = 0.11373956383989692,
                           max_leaf_nodes = None,
                           min_samples_leaf = 1,
                           min_samples_split = 10,
                           n_estimators = 205)
)

pipeline.fit(X_train.drop(columns=column), y_train)
score_without = pipeline.score(X_val.drop(columns=column), y_val)
print(f'Validation Accuracy without {column}: {score_without}')

# Fit with column
pipeline = make_pipeline(
    ce.OneHotEncoder(use_cat_names=True),
    SimpleImputer(strategy = 'mean'),
    RandomForestClassifier(random_state = 42, max_depth = 10,
                           max_features = 0.11373956383989692,
                           max_leaf_nodes = None,
                           min_samples_leaf = 1,
                           min_samples_split = 10,
                           n_estimators = 205)
)

pipeline.fit(X_train, y_train)
score_with = pipeline.score(X_val, y_val)
print(f'Validation Accuracy with {column}: {score_with}')

# Compare the error with & without column
print(f'Drop-Column Importance for {column}: {score_with - score_without}')
```

```
Validation Accuracy without cookie_eat_serve_per_month: 0.40039447731755423
Validation Accuracy with cookie_eat_serve_per_month: 0.398422090729783
Drop-Column Importance for cookie_eat_serve_per_month: -0.0019723865877712132
```

In [28]:
```
# Rerun the permutation importance process, but for a different feature
feature = 'language'
X_val_permuted = X_val.copy()
X_val_permuted[feature] = np.random.permutation(X_val[feature])
score_permuted = pipeline.score(X_val_permuted, y_val)

print(f'Validation Accuracy without {feature} permuted: {score_permuted}')
print(f'Validation Accuracy with {feature}: {score_with}')
print(f'Permutation Importance: {score_with - score_permuted}')
```

```
Validation Accuracy without language permuted: 0.3892176199868508
Validation Accuracy with language: 0.398422090729783
Permutation Importance: 0.009204470742932236
```

In [29]:
```
# Using Eli5 library which does not work with pipelines
transformers = make_pipeline(
    ce.OneHotEncoder(use_cat_names=True),
    SimpleImputer(strategy='mean')
)

X_train_transformed = transformers.fit_transform(X_train)
X_val_transformed = transformers.transform(X_val)

model = RandomForestClassifier(random_state = 42, max_depth = 10,
                               max_features = 0.11373956383989692,
                               max_leaf_nodes = None,
                               min_samples_leaf = 1,
                               min_samples_split = 10,
                               n_estimators = 205)

model.fit(X_train_transformed, y_train)
```

Out[29]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=10, max_features=0.11373956383989692,
                       max_leaf_nodes=None, min_impurity_decrease=0.0,
                       min_impurity_split=None, min_samples_leaf=1,
                       min_samples_split=10, min_weight_fraction_leaf=0.0,
                       n_estimators=205, n_jobs=None, oob_score=False,
                       random_state=42, verbose=0, warm_start=False)
```

In [30]:
```
# Get permutation importances
! pip install eli5
from eli5.sklearn import PermutationImportance
import eli5

permuter = PermutationImportance(
    model,
    scoring='accuracy',
    n_iter=2,
    random_state=42
)

permuter.fit(X_val_transformed, y_val)
feature_names = X_val.columns.tolist()

eli5.show_weights(
    permuter,
    top=None, # show permutation importances for all features
    feature_names=feature_names
)
```

```
Requirement already satisfied: eli5 in c:\users\asg\.conda\envs\lambda\lib\site-packages (0.10.1)
Requirement already satisfied: attrs>16.0.0 in c:\users\asg\.conda\envs\lambda\lib\site-packages (fr
om eli5) (19.1.0)
Requirement already satisfied: scipy in c:\users\asg\.conda\envs\lambda\lib\site-packages (from eli
5) (1.3.1)
Requirement already satisfied: numpy>=1.9.0 in c:\users\asg\.conda\envs\lambda\lib\site-packages (fr
om eli5) (1.16.4)
Requirement already satisfied: graphviz in c:\users\asg\.conda\envs\lambda\lib\site-packages (from e
li5) (0.12)
Requirement already satisfied: tabulate>=0.7.7 in c:\users\asg\.conda\envs\lambda\lib\site-packages
(from eli5) (0.8.3)
Requirement already satisfied: jinja2 in c:\users\asg\.conda\envs\lambda\lib\site-packages (from eli
5) (2.10.1)
Requirement already satisfied: six in c:\users\asg\.conda\envs\lambda\lib\site-packages (from eli5)
(1.12.0)
Requirement already satisfied: scikit-learn>=0.18 in c:\users\asg\.conda\envs\lambda\lib\site-packag
es (from eli5) (0.21.3)
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\asg\.conda\envs\lambda\lib\site-packages
(from jinja2->eli5) (1.1.1)
Requirement already satisfied: joblib>=0.11 in c:\users\asg\.conda\envs\lambda\lib\site-packages (fr
om scikit-learn>=0.18->eli5) (0.13.2)
```

Out[30]:

Weight	Feature
0.0079 ± 0.0026	language
0.0059 ± 0.0000	walk_leisure_ distance
0.0039 ± 0.0013	smokeless_even_once
0.0030 ± 0.0007	red_meat_eat_serve_per_month
0.0030 ± 0.0059	coffee_times_per_month
0.0026 ± 0.0039	walk_number_wk
0.0023 ± 0.0033	red_meat_eat_times_per_month
0.0023 ± 0.0007	cigarette_even_once
0.0020 ± 0.0066	walk_leisure_number_wk
0.0016 ± 0.0020	calcium_past_month
0.0016 ± 0.0020	fries_eat_serve_per_month
0.0016 ± 0.0059	cigar_even_once
0.0016 ± 0.0046	single_walk_distance
0.0016 ± 0.0007	bread_eat_times_per_month
0.0013 ± 0.0079	walkable_bus_stop
0.0010 ± 0.0033	cheese_eat_serve_per_month
0.0010 ± 0.0046	fruit_eat_times_per_month
0.0010 ± 0.0033	weather_discourages_walk
0.0010 ± 0.0007	sports_drink_times_per_month
0.0007 ± 0.0013	walkable_relaxation
0.0007 ± 0.0039	milk_serve_per_month
0.0007 ± 0.0013	vitD_days_in_month
0.0007 ± 0.0026	milk_type
0.0003 ± 0.0059	bread_eat_serve_per_month
0.0003 ± 0.0007	vitamin_past_month
0.0003 ± 0.0046	cheese_eat_times_per_month
0.0003 ± 0.0020	walk_leisure_ time
0 ± 0.0000	vegies_eat_serve_per_month
0 ± 0.0000	genetic_counseling_with_MD
0 ± 0.0000	walkway_existence
0 ± 0.0000	calcium_days_in_month
-0.0000 ± 0.0026	vegies_eat_times_per_month
-0.0003 ± 0.0020	cereal_serve_per_month
-0.0003 ± 0.0059	walkable_entertainment
-0.0007 ± 0.0013	pizza_eat_times_per_month
-0.0007 ± 0.0000	multivitamin_past_month
-0.0007 ± 0.0000	single_walk_time
-0.0007 ± 0.0053	milk_times_per_month
-0.0010 ± 0.0033	candy_eat_serve_per_month
-0.0010 ± 0.0033	cookie_eat_serve_per_month
-0.0010 ± 0.0033	cookie_eat_times_per_month
-0.0010 ± 0.0020	had_genetic_counseling
-0.0010 ± 0.0033	vitD_reason
-0.0010 ± 0.0059	donut_eat_times_per_month
-0.0010 ± 0.0007	1st_kind_cereal_eaten
-0.0010 ± 0.0007	animals_discourage_walking
-0.0010 ± 0.0007	crime_discourages_walking
-0.0010 ± 0.0085	soda_times_per_month
-0.0010 ± 0.0046	juice_times_per_month
-0.0010 ± 0.0007	genetic_counseling_for_cancer
-0.0013 ± 0.0026	ice_cream_eat_times_per_month
-0.0013 ± 0.0000	fruit_drink_times_per_month
-0.0013 ± 0.0039	pop_corn_eat_times_per_month
-0.0013 ± 0.0053	processed_meat_eat_times_per_month
-0.0016 ± 0.0033	grains_eat_times_per_month
-0.0016 ± 0.0007	more_than_one_cereal_type
-0.0016 ± 0.0020	vitD_past_month
-0.0016 ± 0.0007	traffic_discourages_walking
-0.0020 ± 0.0000	walk_past_wk
-0.0020 ± 0.0013	fries_eat_times_per_month
-0.0020 ± 0.0026	beans_eat_times_per_month
-0.0023 ± 0.0007	beans_eat_serve_per_month
-0.0023 ± 0.0007	salsa_eat_times_per_month
-0.0023 ± 0.0007	walk_leisure_past_wk
-0.0023 ± 0.0033	soda_serve_per_month
-0.0026 ± 0.0000	tomatoe_eat_times_per_month
-0.0026 ± 0.0013	grains_eat_serve_per_month
-0.0030 ± 0.0020	tomatoe_eat_serve_per_month
-0.0030 ± 0.0033	pipe_even_once
-0.0030 ± 0.0059	walkable_retail
-0.0033 ± 0.0026	2nd_kind_cereal_eaten
-0.0033 ± 0.0000	potatoe_eat_times_per_month
-0.0036 ± 0.0072	see_walking_from_home
-0.0036 ± 0.0007	cereal_times_per_month
-0.0039 ± 0.0066	streets_have_walkways
-0.0043 ± 0.0007	salad_eat_times_per_month
-0.0043 ± 0.0020	candy_eat_times_per_month
-0.0046 ± 0.0026	multivitamin_days_in_month

In [31]:
```
# Thus, language is way more important according to feature permutation than according to feature im
portance in the Random Forrest model
# Use importances for feature selection
print('Shape before removing features:', X_train.shape)
```

```
Shape before removing features: (6081, 78)
```

In [32]:
```
# Remove features of 0 importance
zero_importance = 0.0003
mask = permuter.feature_importances_ > zero_importance
features = X_train.columns[mask]
X_train = X_train[features]
print('Shape after removing features:', X_train.shape)
```

```
Shape after removing features: (6081, 27)
```

In [33]:
```
# Random Forest with reduced features to 27
X_val = X_val[features]

pipeline = make_pipeline(
    ce.OneHotEncoder(use_cat_names=True),
    SimpleImputer(strategy = 'mean'),
    RandomForestClassifier(random_state = 42, max_depth = 10,
                           max_features = 0.11373956383989692,
                           max_leaf_nodes = None,
                           min_samples_leaf = 1,
                           min_samples_split = 10,
                           n_estimators = 205)
)

# Fit on train, score on val
pipeline.fit(X_train, y_train)
print('Validation Accuracy', pipeline.score(X_val, y_val))
```

```
Validation Accuracy 0.4076265614727153
```

In [34]:
```
# Validation Accuracy History
# 0.2864660417694458- baseline guessing the majority class
# 0.4010853478046374- initial fit with optimal hyperparameters
# 0.398422090729783 - use pipeline with random forest
# 0.3945074823219865- from cross validation
# 0.398422090729783 - doing permutation importance
# 0.4076265614727153- after removing features of zero importance
```

In [35]:
```
# Gradient boosting using XGboost
encoder = ce.OrdinalEncoder()
X_train_encoded = encoder.fit_transform(X_train)
X_val_encoded = encoder.transform(X_val)
X_train.shape, X_val.shape, X_train_encoded.shape, X_val_encoded.shape
```

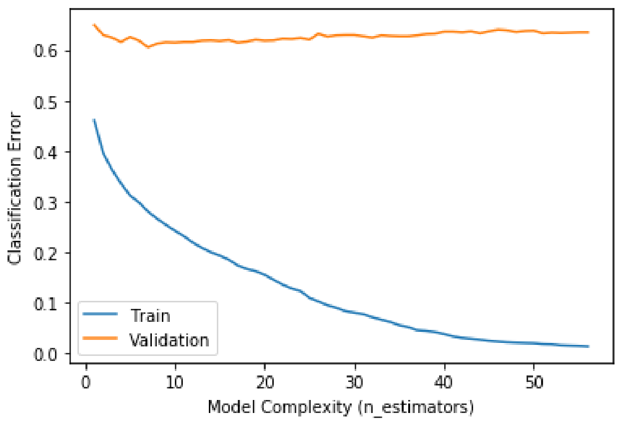
Out[35]:
```
((6081, 27), (1521, 27), (6081, 27), (1521, 27))
```

In [54]:
```
#XGboost with learning_rate=0.25
from xgboost import XGBClassifier

eval_set = [(X_train_encoded, y_train),
            (X_val_encoded, y_val)]

model = XGBClassifier(
    random_state = 42,
    max_depth = 10,
    max_features = 0.11373956383989692,
    max_leaf_nodes = None,
    min_samples_leaf = 1,
    min_samples_split = 10,
    n_estimators = 205,
    learning_rate=0.25,
    n_jobs=-1
)

model.fit(X_train_encoded, y_train, eval_set=eval_set, eval_metric='merror',
    early_stopping_rounds=50)
```

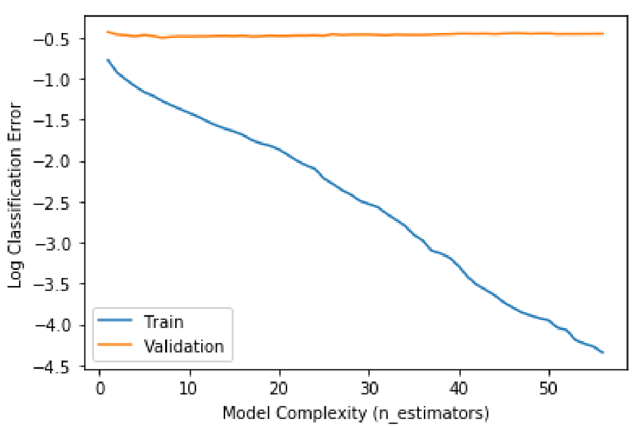
```
[0]	validation_0-merror:0.460615	validation_1-merror:0.649573
Multiple eval metrics have been passed: 'validation_1-merror' will be used for early stopping.

Will train until validation_1-merror hasn't improved in 50 rounds.
[1]	validation_0-merror:0.396152	validation_1-merror:0.629849
[2]	validation_0-merror:0.362769	validation_1-merror:0.624589
[3]	validation_0-merror:0.335636	validation_1-merror:0.616042
[4]	validation_0-merror:0.311955	validation_1-merror:0.625904
[5]	validation_0-merror:0.298142	validation_1-merror:0.618672
[6]	validation_0-merror:0.280053	validation_1-merror:0.605523
[7]	validation_0-merror:0.266075	validation_1-merror:0.612755
[8]	validation_0-merror:0.253741	validation_1-merror:0.615385
[9]	validation_0-merror:0.242065	validation_1-merror:0.614727
[10]	validation_0-merror:0.231376	validation_1-merror:0.616042
[11]	validation_0-merror:0.21855	validation_1-merror:0.616042
[12]	validation_0-merror:0.208189	validation_1-merror:0.618672
[13]	validation_0-merror:0.199638	validation_1-merror:0.619329
[14]	validation_0-merror:0.192896	validation_1-merror:0.618014
[15]	validation_0-merror:0.184838	validation_1-merror:0.619987
[16]	validation_0-merror:0.173491	validation_1-merror:0.614727
[17]	validation_0-merror:0.166584	validation_1-merror:0.6167
[18]	validation_0-merror:0.162144	validation_1-merror:0.620644
[19]	validation_0-merror:0.154909	validation_1-merror:0.618672
[20]	validation_0-merror:0.144877	validation_1-merror:0.619329
[21]	validation_0-merror:0.135175	validation_1-merror:0.622617
[22]	validation_0-merror:0.127611	validation_1-merror:0.621959
[23]	validation_0-merror:0.122348	validation_1-merror:0.623932
[24]	validation_0-merror:0.108864	validation_1-merror:0.620644
[25]	validation_0-merror:0.101792	validation_1-merror:0.632479
[26]	validation_0-merror:0.094392	validation_1-merror:0.626561
[27]	validation_0-merror:0.089295	validation_1-merror:0.629191
[28]	validation_0-merror:0.082717	validation_1-merror:0.629849
[29]	validation_0-merror:0.079592	validation_1-merror:0.629849
[30]	validation_0-merror:0.076632	validation_1-merror:0.627219
[31]	validation_0-merror:0.070548	validation_1-merror:0.624589
[32]	validation_0-merror:0.065779	validation_1-merror:0.629191
[33]	validation_0-merror:0.061174	validation_1-merror:0.627876
[34]	validation_0-merror:0.054761	validation_1-merror:0.627219
[35]	validation_0-merror:0.050978	validation_1-merror:0.627219
[36]	validation_0-merror:0.045058	validation_1-merror:0.629191
[37]	validation_0-merror:0.043578	validation_1-merror:0.631821
[38]	validation_0-merror:0.041276	validation_1-merror:0.632479
[39]	validation_0-merror:0.037329	validation_1-merror:0.636423
[40]	validation_0-merror:0.032725	validation_1-merror:0.636423
[41]	validation_0-merror:0.029765	validation_1-merror:0.635108
[42]	validation_0-merror:0.027956	validation_1-merror:0.637081
[43]	validation_0-merror:0.026147	validation_1-merror:0.633136
[44]	validation_0-merror:0.024009	validation_1-merror:0.637081
[45]	validation_0-merror:0.022529	validation_1-merror:0.640368
[46]	validation_0-merror:0.021214	validation_1-merror:0.639053
[47]	validation_0-merror:0.020391	validation_1-merror:0.635766
[48]	validation_0-merror:0.019734	validation_1-merror:0.637738
[49]	validation_0-merror:0.01924	validation_1-merror:0.638396
[50]	validation_0-merror:0.017596	validation_1-merror:0.633136
[51]	validation_0-merror:0.017102	validation_1-merror:0.634451
[52]	validation_0-merror:0.015129	validation_1-merror:0.633794
[53]	validation_0-merror:0.014471	validation_1-merror:0.634451
[54]	validation_0-merror:0.013978	validation_1-merror:0.635108
[55]	validation_0-merror:0.012991	validation_1-merror:0.635108
[56]	validation_0-merror:0.01184	validation_1-merror:0.635108
Stopping. Best iteration:
[6]	validation_0-merror:0.280053	validation_1-merror:0.605523
```

Out[54]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0,
              learning_rate=0.25, max_delta_step=0, max_depth=10,
              max_features=0.11373956383989692, max_leaf_nodes=None,
              min_child_weight=1, min_samples_leaf=1, min_samples_split=10,
              missing=None, n_estimators=205, n_jobs=-1, nthread=None,
              objective='multi:softprob', random_state=42, reg_alpha=0,
              reg_lambda=1, scale_pos_weight=1, seed=None, silent=None,
              subsample=1, verbosity=1)
```

In [55]:
```
# Plot the results
results = model.evals_result()
train_error = results['validation_0']['merror']
val_error = results['validation_1']['merror']
epoch = range(1, len(train_error)+1)
plt.plot(epoch, train_error, label='Train')
plt.plot(epoch, val_error, label='Validation')
plt.ylabel('Classification Error')
plt.xlabel('Model Complexity (n_estimators)')
# plt.ylim((0.5, 0.7)) # Zoom in
plt.legend();
```



In [56]:
```
# Plot log classification error versus model complexity
import numpy as np
results = model.evals_result()
log_train_error = np.log(results['validation_0']['merror'])
log_val_error = np.log(results['validation_1']['merror'])
epoch = range(1, len(train_error)+1)
plt.plot(epoch, log_train_error, label='Train')
plt.plot(epoch, log_val_error, label='Validation')
plt.ylabel('Log Classification Error')
plt.xlabel('Model Complexity (n_estimators)')
# plt.ylim((-0.75, -0.4)) # Zoom in
plt.legend();
```



In [57]:
```
# Note the Classification Error is minimum at n_estimators = 6 in the above
# This is best scene when using the Zoom In scaling

#Gradient Boosting R^2
from sklearn.metrics import r2_score
from xgboost import XGBRegressor

gb = make_pipeline(
    ce.OrdinalEncoder(),
    XGBRegressor(n_estimators=46, objective='reg:squarederror', n_jobs=-1)
)

gb.fit(X_train, y_train)
y_pred = gb.predict(X_val)
from sklearn.metrics import r2_score
from xgboost import XGBRegressor
print('Gradient Boosting R^2', r2_score(y_val, y_pred))
```

```
Gradient Boosting R^2 0.2737135437482129

C:\Users\ASG\.conda\envs\Lambda\lib\site-packages\xgboost\core.py:587: FutureWarning: Series.base is
deprecated and will be removed in a future version
  if getattr(data, 'base', None) is not None and \
```

In [58]:
```
# Getting the value distribution for the language feature
df_smoking1['language'].value_counts()
```

Out[58]:
```
5    5713
4    1031
8     213
3     203
1     169
2     138
6     134
9       1
Name: language, dtype: int64
```

In [59]:
```
# Define function to vary the language feature while holding all other features constant
import numpy as np

def vary_language(model, example):
    print('Vary language, hold other features constant', '\n')
    example = example.copy()
    preds = []
    for lang in range(1, 9, 1):
        example['language'] = lang
        pred = model.predict(example)[0]
        print(f'Predicted cigarettes_per_day_bin: {pred:.3f}%')
        print(example.to_string(), '\n')
        preds.append(pred)
    print('Difference between predictions')
    print(np.diff(preds))
```

In [64]:
```
# Vary the language feature while holding all other features constant for the first row
example1 = X_val.iloc[[0]]
vary_language(gb, example1)
```

```
Vary language, hold other features constant

Predicted cigarettes_per_day_bin: 2.890%
       language  milk_serve_per_month  milk_type  coffee_times_per_month  sports_drink_times_per_mon
th  fruit_eat_times_per_month  fries_eat_serve_per_month  cheese_eat_serve_per_month  cheese_eat_tim
es_per_month  red_meat_eat_serve_per_month  red_meat_eat_times_per_month  bread_eat_serve_per_month
bread_eat_times_per_month  vitamin_past_month  calcium_past_month  vitD_days_in_month  walk_number_w
k  single_walk_distance  walk_leisure_number_wk  walk_leisure_ distance  walk_leisure_ time  weather
_discourages_walk  walkable_bus_stop  walkable_relaxation  cigarette_even_once  cigar_even_once  smo
keless_even_once
31502        1                     3        2.0                        0
0                          2                           2                           4
2                             3                           2                           4
2                  1                 0.0                0.0              0.0                   0.0
0.0                     0.0                0.0                        3                  0
0                     0                1                    0

Predicted cigarettes_per_day_bin: 2.890%
       language  milk_serve_per_month  milk_type  coffee_times_per_month  sports_drink_times_per_mon
th  fruit_eat_times_per_month  fries_eat_serve_per_month  cheese_eat_serve_per_month  cheese_eat_tim
es_per_month  red_meat_eat_serve_per_month  red_meat_eat_times_per_month  bread_eat_serve_per_month
bread_eat_times_per_month  vitamin_past_month  calcium_past_month  vitD_days_in_month  walk_number_w
k  single_walk_distance  walk_leisure_number_wk  walk_leisure_ distance  walk_leisure_ time  weather
_discourages_walk  walkable_bus_stop  walkable_relaxation  cigarette_even_once  cigar_even_once  smo
keless_even_once
31502        2                     3        2.0                        0
0                          2                           2                           4
2                             3                           2                           4
2                  1                 0.0                0.0              0.0                   0.0
0.0                     0.0                0.0                        3                  0
0                     0                1                    0

Predicted cigarettes_per_day_bin: 3.017%
       language  milk_serve_per_month  milk_type  coffee_times_per_month  sports_drink_times_per_mon
th  fruit_eat_times_per_month  fries_eat_serve_per_month  cheese_eat_serve_per_month  cheese_eat_tim
es_per_month  red_meat_eat_serve_per_month  red_meat_eat_times_per_month  bread_eat_serve_per_month
bread_eat_times_per_month  vitamin_past_month  calcium_past_month  vitD_days_in_month  walk_number_w
k  single_walk_distance  walk_leisure_number_wk  walk_leisure_ distance  walk_leisure_ time  weather
_discourages_walk  walkable_bus_stop  walkable_relaxation  cigarette_even_once  cigar_even_once  smo
keless_even_once
31502        3                     3        2.0                        0
0                          2                           2                           4
2                             3                           2                           4
2                  1                 0.0                0.0              0.0                   0.0
0.0                     0.0                0.0                        3                  0
0                     0                1                    0

Predicted cigarettes_per_day_bin: 3.268%
       language  milk_serve_per_month  milk_type  coffee_times_per_month  sports_drink_times_per_mon
th  fruit_eat_times_per_month  fries_eat_serve_per_month  cheese_eat_serve_per_month  cheese_eat_tim
es_per_month  red_meat_eat_serve_per_month  red_meat_eat_times_per_month  bread_eat_serve_per_month
bread_eat_times_per_month  vitamin_past_month  calcium_past_month  vitD_days_in_month  walk_number_w
k  single_walk_distance  walk_leisure_number_wk  walk_leisure_ distance  walk_leisure_ time  weather
_discourages_walk  walkable_bus_stop  walkable_relaxation  cigarette_even_once  cigar_even_once  smo
keless_even_once
31502        4                     3        2.0                        0
0                          2                           2                           4
2                             3                           2                           4
2                  1                 0.0                0.0              0.0                   0.0
0.0                     0.0                0.0                        3                  0
0                     0                1                    0

Predicted cigarettes_per_day_bin: 3.323%
       language  milk_serve_per_month  milk_type  coffee_times_per_month  sports_drink_times_per_mon
th  fruit_eat_times_per_month  fries_eat_serve_per_month  cheese_eat_serve_per_month  cheese_eat_tim
es_per_month  red_meat_eat_serve_per_month  red_meat_eat_times_per_month  bread_eat_serve_per_month
bread_eat_times_per_month  vitamin_past_month  calcium_past_month  vitD_days_in_month  walk_number_w
k  single_walk_distance  walk_leisure_number_wk  walk_leisure_ distance  walk_leisure_ time  weather
_discourages_walk  walkable_bus_stop  walkable_relaxation  cigarette_even_once  cigar_even_once  smo
keless_even_once
```

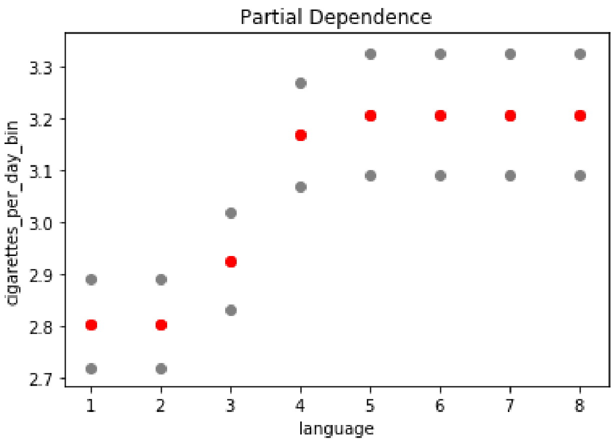
```
31502        5                    3        2.0                       0
0                          2                          2                          4
2                             3                          2                          4
2                  1                0.0                0.0              0.0                  0.0
0.0                     0.0                0.0                     3               0
0                    0                1                    0

Predicted cigarettes_per_day_bin: 3.323%
       language  milk_serve_per_month  milk_type  coffee_times_per_month  sports_drink_times_per_mon
th  fruit_eat_times_per_month  fries_eat_serve_per_month  cheese_eat_serve_per_month  cheese_eat_tim
es_per_month  red_meat_eat_serve_per_month  red_meat_eat_times_per_month  bread_eat_serve_per_month
bread_eat_times_per_month  vitamin_past_month  calcium_past_month  vitD_days_in_month  walk_number_w
k  single_walk_distance  walk_leisure_number_wk  walk_leisure_ distance  walk_leisure_ time  weather
_discourages_walk  walkable_bus_stop  walkable_relaxation  cigarette_even_once  cigar_even_once  smo
keless_even_once
31502        6                    3        2.0                       0
0                          2                          2                          4
2                             3                          2                          4
2                  1                0.0                0.0              0.0                  0.0
0.0                     0.0                0.0                     3               0
0                    0                1                    0

Predicted cigarettes_per_day_bin: 3.323%
       language  milk_serve_per_month  milk_type  coffee_times_per_month  sports_drink_times_per_mon
th  fruit_eat_times_per_month  fries_eat_serve_per_month  cheese_eat_serve_per_month  cheese_eat_tim
es_per_month  red_meat_eat_serve_per_month  red_meat_eat_times_per_month  bread_eat_serve_per_month
bread_eat_times_per_month  vitamin_past_month  calcium_past_month  vitD_days_in_month  walk_number_w
k  single_walk_distance  walk_leisure_number_wk  walk_leisure_ distance  walk_leisure_ time  weather
_discourages_walk  walkable_bus_stop  walkable_relaxation  cigarette_even_once  cigar_even_once  smo
keless_even_once
31502        7                    3        2.0                       0
0                          2                          2                          4
2                             3                          2                          4
2                  1                0.0                0.0              0.0                  0.0
0.0                     0.0                0.0                     3               0
0                    0                1                    0

Predicted cigarettes_per_day_bin: 3.323%
       language  milk_serve_per_month  milk_type  coffee_times_per_month  sports_drink_times_per_mon
th  fruit_eat_times_per_month  fries_eat_serve_per_month  cheese_eat_serve_per_month  cheese_eat_tim
es_per_month  red_meat_eat_serve_per_month  red_meat_eat_times_per_month  bread_eat_serve_per_month
bread_eat_times_per_month  vitamin_past_month  calcium_past_month  vitD_days_in_month  walk_number_w
k  single_walk_distance  walk_leisure_number_wk  walk_leisure_ distance  walk_leisure_ time  weather
_discourages_walk  walkable_bus_stop  walkable_relaxation  cigarette_even_once  cigar_even_once  smo
keless_even_once
31502        8                    3        2.0                       0
0                          2                          2                          4
2                             3                          2                          4
2                  1                0.0                0.0              0.0                  0.0
0.0                     0.0                0.0                     3               0
0                    0                1                    0

Difference between predictions
[0.         0.12740803 0.25086045 0.05542064 0.         0.
 0.        ]
```

In [65]:
```
# Vary the language feature while holding all other features constant for the second row
example2 = X_val.iloc[[2]]
vary_language(gb, example2)
```

```
Vary language, hold other features constant

Predicted cigarettes_per_day_bin: 2.719%
       language  milk_serve_per_month  milk_type  coffee_times_per_month  sports_drink_times_per_mon
th  fruit_eat_times_per_month  fries_eat_serve_per_month  cheese_eat_serve_per_month  cheese_eat_tim
es_per_month  red_meat_eat_serve_per_month  red_meat_eat_times_per_month  bread_eat_serve_per_month 
bread_eat_times_per_month  vitamin_past_month  calcium_past_month  vitD_days_in_month  walk_number_w
k  single_walk_distance  walk_leisure_number_wk  walk_leisure_ distance  walk_leisure_ time  weather
_discourages_walk  walkable_bus_stop  walkable_relaxation  cigarette_even_once  cigar_even_once  smo
keless_even_once
27082        1                     2        2.0                        0
0                          3                          1                           1
2                            2                          3                          0
0                   1                0.0                30.0              0.0                   0.0
0.0                    0.0                0.0                         1                  0
1                    0                0                   0

Predicted cigarettes_per_day_bin: 2.719%
       language  milk_serve_per_month  milk_type  coffee_times_per_month  sports_drink_times_per_mon
th  fruit_eat_times_per_month  fries_eat_serve_per_month  cheese_eat_serve_per_month  cheese_eat_tim
es_per_month  red_meat_eat_serve_per_month  red_meat_eat_times_per_month  bread_eat_serve_per_month 
bread_eat_times_per_month  vitamin_past_month  calcium_past_month  vitD_days_in_month  walk_number_w
k  single_walk_distance  walk_leisure_number_wk  walk_leisure_ distance  walk_leisure_ time  weather
_discourages_walk  walkable_bus_stop  walkable_relaxation  cigarette_even_once  cigar_even_once  smo
keless_even_once
27082        2                     2        2.0                        0
0                          3                          1                           1
2                            2                          3                          0
0                   1                0.0                30.0              0.0                   0.0
0.0                    0.0                0.0                         1                  0
1                    0                0                   0

Predicted cigarettes_per_day_bin: 2.832%
       language  milk_serve_per_month  milk_type  coffee_times_per_month  sports_drink_times_per_mon
th  fruit_eat_times_per_month  fries_eat_serve_per_month  cheese_eat_serve_per_month  cheese_eat_tim
es_per_month  red_meat_eat_serve_per_month  red_meat_eat_times_per_month  bread_eat_serve_per_month 
bread_eat_times_per_month  vitamin_past_month  calcium_past_month  vitD_days_in_month  walk_number_w
k  single_walk_distance  walk_leisure_number_wk  walk_leisure_ distance  walk_leisure_ time  weather
_discourages_walk  walkable_bus_stop  walkable_relaxation  cigarette_even_once  cigar_even_once  smo
keless_even_once
27082        3                     2        2.0                        0
0                          3                          1                           1
2                            2                          3                          0
0                   1                0.0                30.0              0.0                   0.0
0.0                    0.0                0.0                         1                  0
1                    0                0                   0

Predicted cigarettes_per_day_bin: 3.069%
       language  milk_serve_per_month  milk_type  coffee_times_per_month  sports_drink_times_per_mon
th  fruit_eat_times_per_month  fries_eat_serve_per_month  cheese_eat_serve_per_month  cheese_eat_tim
es_per_month  red_meat_eat_serve_per_month  red_meat_eat_times_per_month  bread_eat_serve_per_month 
bread_eat_times_per_month  vitamin_past_month  calcium_past_month  vitD_days_in_month  walk_number_w
k  single_walk_distance  walk_leisure_number_wk  walk_leisure_ distance  walk_leisure_ time  weather
_discourages_walk  walkable_bus_stop  walkable_relaxation  cigarette_even_once  cigar_even_once  smo
keless_even_once
27082        4                     2        2.0                        0
0                          3                          1                           1
2                            2                          3                          0
0                   1                0.0                30.0              0.0                   0.0
0.0                    0.0                0.0                         1                  0
1                    0                0                   0

Predicted cigarettes_per_day_bin: 3.090%
       language  milk_serve_per_month  milk_type  coffee_times_per_month  sports_drink_times_per_mon
th  fruit_eat_times_per_month  fries_eat_serve_per_month  cheese_eat_serve_per_month  cheese_eat_tim
es_per_month  red_meat_eat_serve_per_month  red_meat_eat_times_per_month  bread_eat_serve_per_month 
bread_eat_times_per_month  vitamin_past_month  calcium_past_month  vitD_days_in_month  walk_number_w
k  single_walk_distance  walk_leisure_number_wk  walk_leisure_ distance  walk_leisure_ time  weather
_discourages_walk  walkable_bus_stop  walkable_relaxation  cigarette_even_once  cigar_even_once  smo
keless_even_once
```

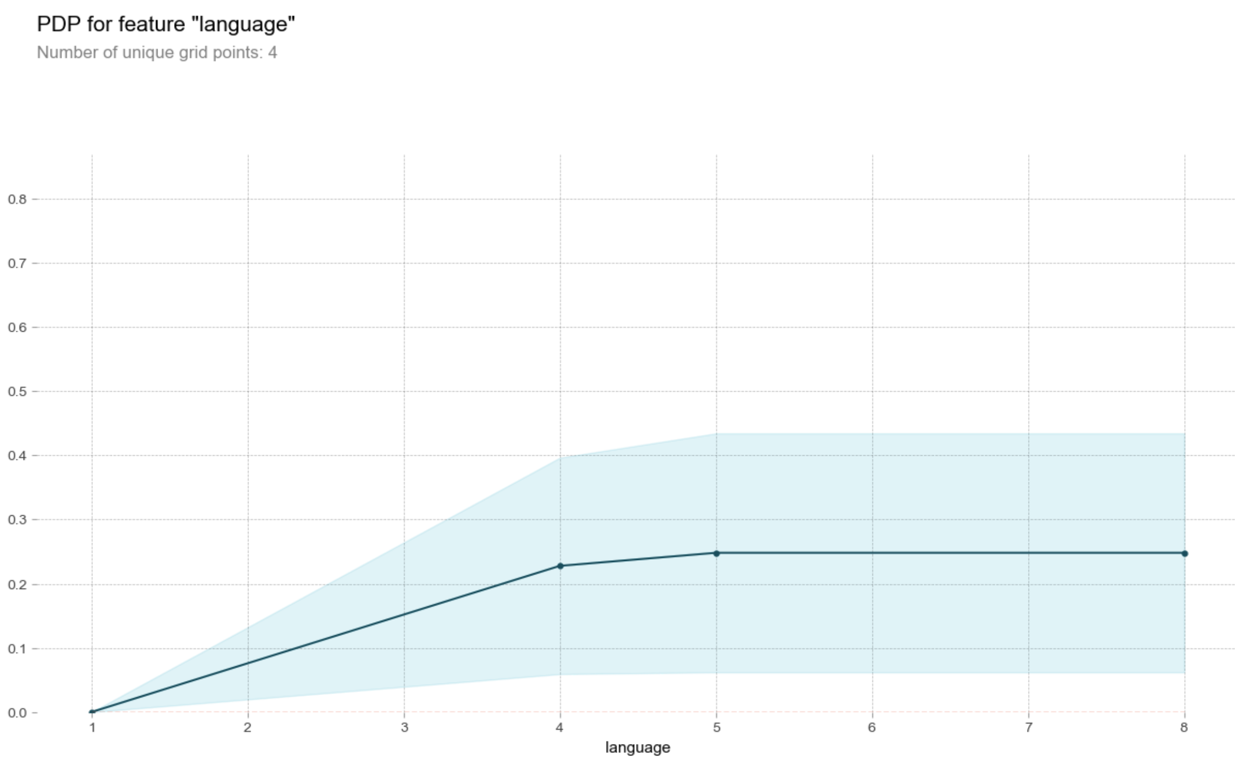
```
27082        5                    2        2.0                     0
0                          3                          1                          1
2                             2                             3                          0
0                   1                 0.0                30.0              0.0                 0.0
0.0                     0.0                 0.0                        1                 0
1                    0                 0                    0

Predicted cigarettes_per_day_bin: 3.090%
       language  milk_serve_per_month  milk_type  coffee_times_per_month  sports_drink_times_per_mon
th  fruit_eat_times_per_month  fries_eat_serve_per_month  cheese_eat_serve_per_month  cheese_eat_tim
es_per_month  red_meat_eat_serve_per_month  red_meat_eat_times_per_month  bread_eat_serve_per_month 
bread_eat_times_per_month  vitamin_past_month  calcium_past_month  vitD_days_in_month  walk_number_w
k  single_walk_distance  walk_leisure_number_wk  walk_leisure_ distance  walk_leisure_ time  weather
_discourages_walk  walkable_bus_stop  walkable_relaxation  cigarette_even_once  cigar_even_once  smo
keless_even_once
27082        6                    2        2.0                     0
0                          3                          1                          1
2                             2                             3                          0
0                   1                 0.0                30.0              0.0                 0.0
0.0                     0.0                 0.0                        1                 0
1                    0                 0                    0

Predicted cigarettes_per_day_bin: 3.090%
       language  milk_serve_per_month  milk_type  coffee_times_per_month  sports_drink_times_per_mon
th  fruit_eat_times_per_month  fries_eat_serve_per_month  cheese_eat_serve_per_month  cheese_eat_tim
es_per_month  red_meat_eat_serve_per_month  red_meat_eat_times_per_month  bread_eat_serve_per_month 
bread_eat_times_per_month  vitamin_past_month  calcium_past_month  vitD_days_in_month  walk_number_w
k  single_walk_distance  walk_leisure_number_wk  walk_leisure_ distance  walk_leisure_ time  weather
_discourages_walk  walkable_bus_stop  walkable_relaxation  cigarette_even_once  cigar_even_once  smo
keless_even_once
27082        7                    2        2.0                     0
0                          3                          1                          1
2                             2                             3                          0
0                   1                 0.0                30.0              0.0                 0.0
0.0                     0.0                 0.0                        1                 0
1                    0                 0                    0

Predicted cigarettes_per_day_bin: 3.090%
       language  milk_serve_per_month  milk_type  coffee_times_per_month  sports_drink_times_per_mon
th  fruit_eat_times_per_month  fries_eat_serve_per_month  cheese_eat_serve_per_month  cheese_eat_tim
es_per_month  red_meat_eat_serve_per_month  red_meat_eat_times_per_month  bread_eat_serve_per_month 
bread_eat_times_per_month  vitamin_past_month  calcium_past_month  vitD_days_in_month  walk_number_w
k  single_walk_distance  walk_leisure_number_wk  walk_leisure_ distance  walk_leisure_ time  weather
_discourages_walk  walkable_bus_stop  walkable_relaxation  cigarette_even_once  cigar_even_once  smo
keless_even_once
27082        8                    2        2.0                     0
0                          3                          1                          1
2                             2                             3                          0
0                   1                 0.0                30.0              0.0                 0.0
0.0                     0.0                 0.0                        1                 0
1                    0                 0                    0

Difference between predictions
[0.         0.11254644 0.23789716 0.02041197 0.         0.
 0.        ]
```

In [66]:
```
# Plot pair dependency of the language feature for rows 1 and 2
%matplotlib inline
import matplotlib.pyplot as plt

examples = pd.concat([example1, example2])
for lang in range(1, 9, 1):
    examples['language'] = lang
    preds = gb.predict(examples)
    for pred in preds:
        plt.scatter(lang, pred, color='grey')
        plt.scatter(lang, np.mean(preds), color='red')
    plt.title('Partial Dependence')
    plt.xlabel('language')
    plt.ylabel('cigarettes_per_day_bin')
```

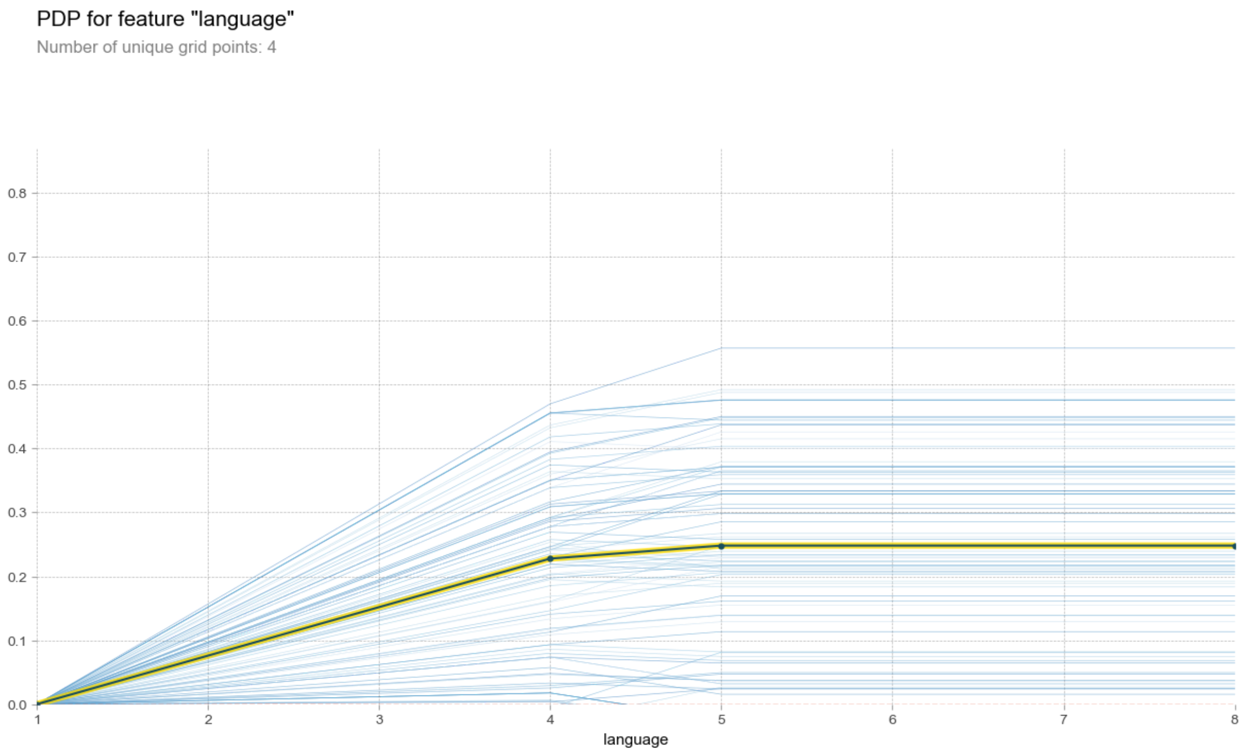


In [67]:
```
# Create patrial dependence plots with one feature
import matplotlib.pyplot as plt
! pip install PDPbox

# First for the language feature
plt.rcParams['figure.dpi'] = 100
from pdpbox.pdp import pdp_isolate, pdp_plot
feature = 'language'
isolated = pdp_isolate(
    model=gb,
    dataset=X_val,
    model_features=X_val.columns,
    feature=feature
)

pdp_plot(isolated, feature_name=feature);
```

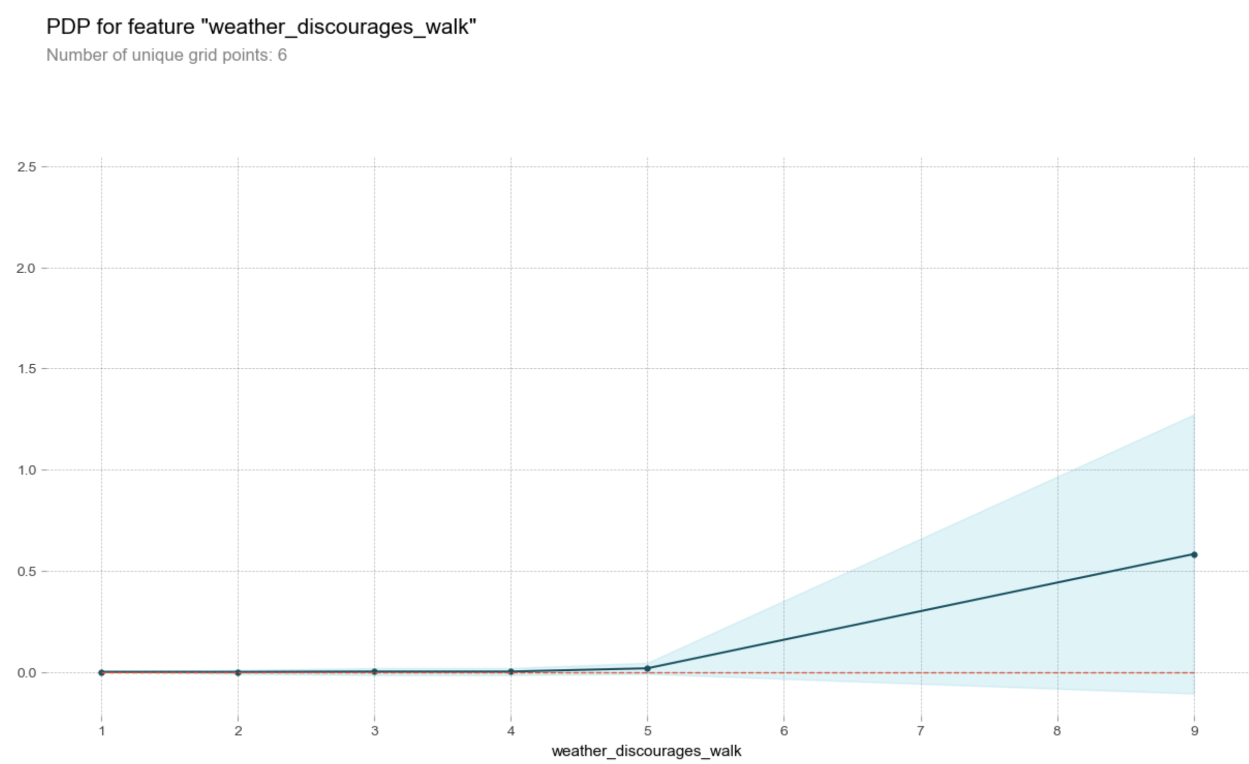
```
Requirement already satisfied: PDPbox in c:\users\asg\.conda\envs\lambda\lib\site-packages (0.2.0)
Requirement already satisfied: joblib in c:\users\asg\.conda\envs\lambda\lib\site-packages (from PDP
box) (0.13.2)
Requirement already satisfied: matplotlib>=2.1.2 in c:\users\asg\.conda\envs\lambda\lib\site-package
s (from PDPbox) (3.1.1)
Requirement already satisfied: psutil in c:\users\asg\.conda\envs\lambda\lib\site-packages (from PDP
box) (5.6.3)
Requirement already satisfied: scipy in c:\users\asg\.conda\envs\lambda\lib\site-packages (from PDPb
ox) (1.3.1)
Requirement already satisfied: numpy in c:\users\asg\.conda\envs\lambda\lib\site-packages (from PDPb
ox) (1.16.4)
Requirement already satisfied: pandas in c:\users\asg\.conda\envs\lambda\lib\site-packages (from PDP
box) (0.23.4)
Requirement already satisfied: scikit-learn in c:\users\asg\.conda\envs\lambda\lib\site-packages (fr
om PDPbox) (0.21.3)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in c:\users\asg\.conda\envs
\lambda\lib\site-packages (from matplotlib>=2.1.2->PDPbox) (2.4.2)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\asg\.conda\envs\lambda\lib\site-pack
ages (from matplotlib>=2.1.2->PDPbox) (2.8.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\asg\.conda\envs\lambda\lib\site-package
s (from matplotlib>=2.1.2->PDPbox) (1.1.0)
Requirement already satisfied: cycler>=0.10 in c:\users\asg\.conda\envs\lambda\lib\site-packages (fr
om matplotlib>=2.1.2->PDPbox) (0.10.0)
Requirement already satisfied: pytz>=2011k in c:\users\asg\.conda\envs\lambda\lib\site-packages (fro
m pandas->PDPbox) (2019.2)
Requirement already satisfied: six>=1.5 in c:\users\asg\.conda\envs\lambda\lib\site-packages (from p
ython-dateutil>=2.1->matplotlib>=2.1.2->PDPbox) (1.12.0)
Requirement already satisfied: setuptools in c:\users\asg\.conda\envs\lambda\lib\site-packages (from
kiwisolver>=1.0.1->matplotlib>=2.1.2->PDPbox) (41.0.1)
```



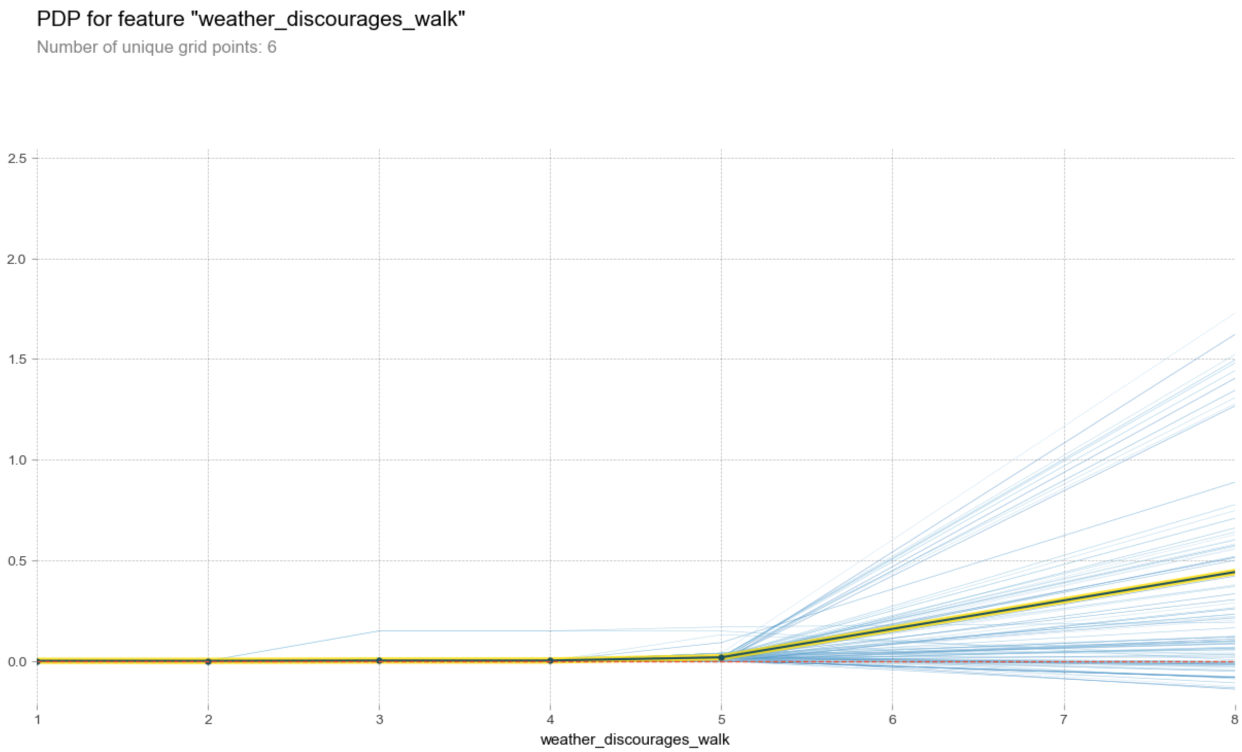
In [68]:
```
# Plot partial dependence plot with ICE lines for the language feature
pdp_plot(isolated, feature_name=feature, plot_lines=True, frac_to_plot=100) # Plot 100 ICE lines
plt.xlim(1,8);
```



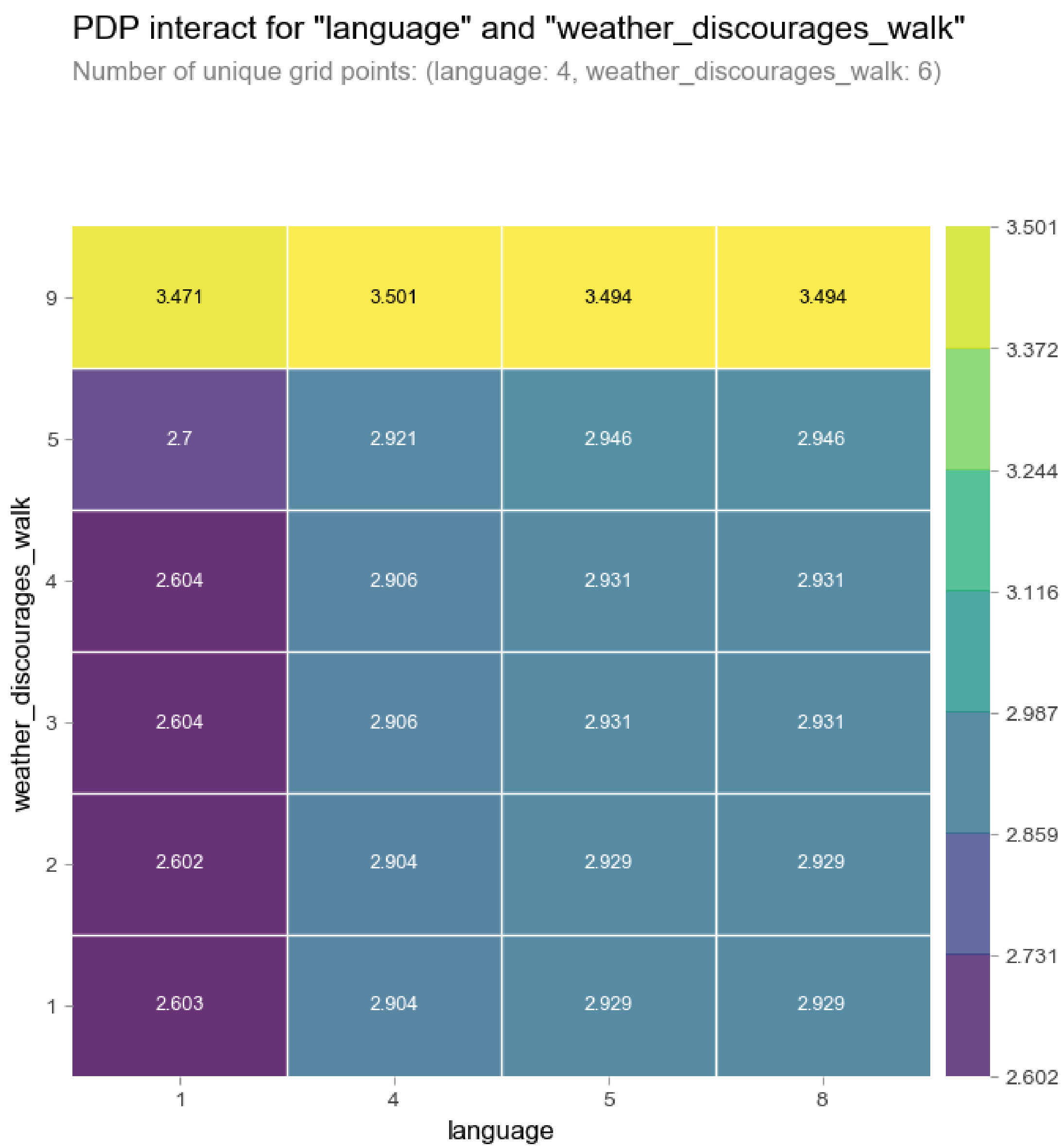
In [69]:
```
# First for the weather_discourages_walk feature
plt.rcParams['figure.dpi'] = 100
from pdpbox.pdp import pdp_isolate, pdp_plot
feature = 'weather_discourages_walk'
isolated = pdp_isolate(
    model=gb,
    dataset=X_val,
    model_features=X_val.columns,
    feature=feature
)

pdp_plot(isolated, feature_name=feature);
```



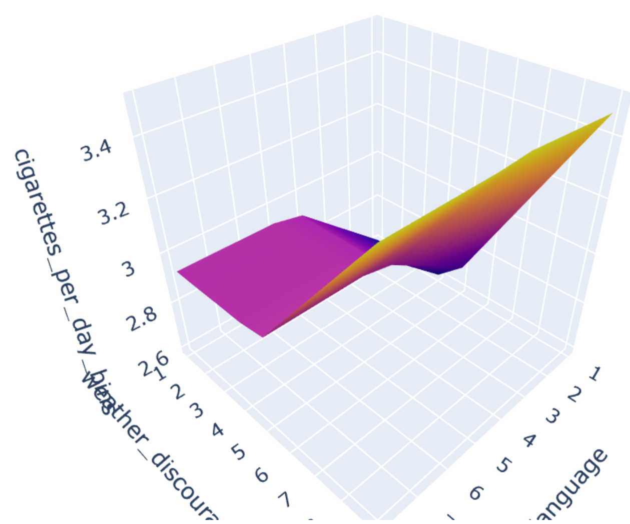
In [70]:
```
# Plot partial dependence plot with ICE lines for the weather_discourages_walk feature
pdp_plot(isolated, feature_name=feature, plot_lines=True, frac_to_plot=100) # Plot 100 ICE lines
plt.xlim(1,8);
```



In [71]:
```
# Partial Dependence Plots with 2 features
from pdpbox.pdp import pdp_interact, pdp_interact_plot

features = ['language', 'weather_discourages_walk']
interaction = pdp_interact(
    model=gb,
    dataset=X_val,
    model_features=X_val.columns,
    features=features
)

pdp_interact_plot(interaction, plot_type='grid', feature_names=features);
```



In [72]:
```
# A two feature partical dependence plot in 3D
pdp = interaction.pdp.pivot_table(
    values='preds',
    columns=features[0],
    index=features[1]
)[::-1] # Slice notation to reverse index order so y axis is ascending

import plotly.graph_objs as go

target = 'cigarettes_per_day_bins'

surface = go.Surface(x=pdp.columns,
                     y=pdp.index,
                     z=pdp.values)

layout = go.Layout(
    scene=dict(
    xaxis=dict(title=features[0]),
    yaxis=dict(title=features[1]),
    zaxis=dict(title=target)
    )
)
fig = go.Figure(surface, layout)
fig.show()
```



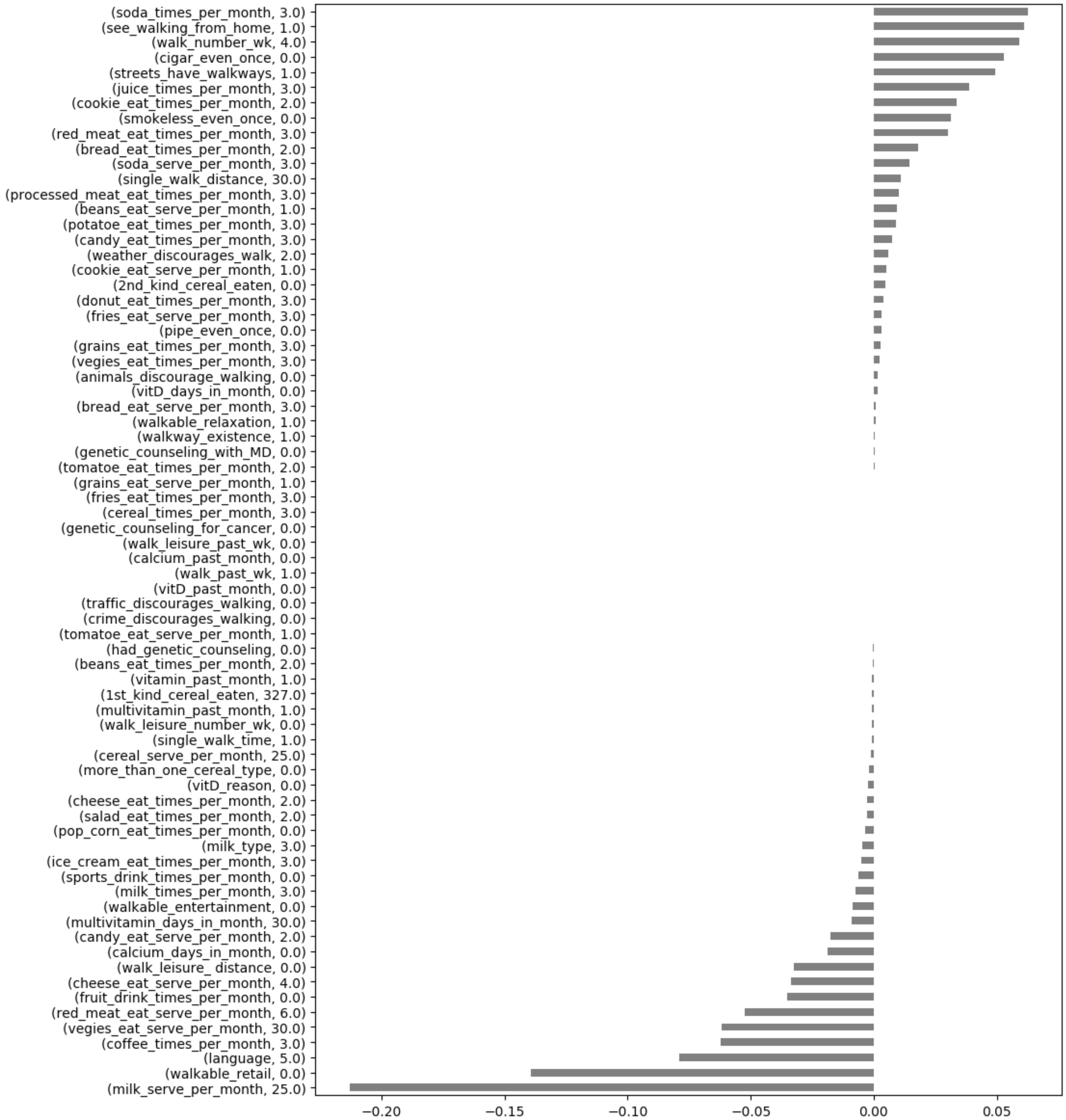
In [73]:
```
# Test ROC AUC
from sklearn.metrics import roc_auc_score
from sklearn.impute import SimpleImputer
from sklearn.pipeline import make_pipeline
from xgboost import XGBClassifier
import category_encoders as ce

processor = make_pipeline(
    ce.OrdinalEncoder(),
    SimpleImputer(strategy='mean')
)

# Note ROC AUC ranges from 0 - 1, the higher the better
X_val_processed = processor.fit_transform(X_val)
```

In [ ]:
```
# Contributrions to making bin 1 (1 - 7 cigarettes per day) for sample 170
! pip install shap==0.23.0
! pip install -I shap

import shap

row = X_val.iloc[[170]]

explainer = shap.TreeExplainer(model)
row_processed = processor.transform(row)
shap_values_input = explainer.shap_values(row_processed)

shap.initjs()
shap.force_plot(
    base_value=explainer.expected_value[0],
    shap_values=shap_values_input[0],
    features=row
)
```

```
Requirement already satisfied: shap==0.23.0 in c:\users\asg\.conda\envs\lambda\lib\site-packages (0.
23.0)
Requirement already satisfied: scikit-learn in c:\users\asg\.conda\envs\lambda\lib\site-packages (fr
om shap==0.23.0) (0.21.3)
Requirement already satisfied: ipython in c:\users\asg\.conda\envs\lambda\lib\site-packages (from sh
ap==0.23.0) (7.8.0)
Requirement already satisfied: pandas in c:\users\asg\.conda\envs\lambda\lib\site-packages (from sha
p==0.23.0) (0.23.4)
Requirement already satisfied: scipy in c:\users\asg\.conda\envs\lambda\lib\site-packages (from shap
==0.23.0) (1.3.1)
Requirement already satisfied: numpy in c:\users\asg\.conda\envs\lambda\lib\site-packages (from shap
==0.23.0) (1.16.4)
Requirement already satisfied: matplotlib in c:\users\asg\.conda\envs\lambda\lib\site-packages (from
shap==0.23.0) (3.1.1)
Requirement already satisfied: tqdm in c:\users\asg\.conda\envs\lambda\lib\site-packages (from shap=
=0.23.0) (4.32.1)
Requirement already satisfied: joblib>=0.11 in c:\users\asg\.conda\envs\lambda\lib\site-packages (fr
om scikit-learn->shap==0.23.0) (0.13.2)
Requirement already satisfied: traitlets>=4.2 in c:\users\asg\.conda\envs\lambda\lib\site-packages
(from ipython->shap==0.23.0) (4.3.2)
Requirement already satisfied: pickleshare in c:\users\asg\.conda\envs\lambda\lib\site-packages (fro
m ipython->shap==0.23.0) (0.7.5)
Requirement already satisfied: backcall in c:\users\asg\.conda\envs\lambda\lib\site-packages (from i
python->shap==0.23.0) (0.1.0)
Requirement already satisfied: setuptools>=18.5 in c:\users\asg\.conda\envs\lambda\lib\site-packages
(from ipython->shap==0.23.0) (41.0.1)
Requirement already satisfied: jedi>=0.10 in c:\users\asg\.conda\envs\lambda\lib\site-packages (from
ipython->shap==0.23.0) (0.15.1)
Requirement already satisfied: prompt-toolkit<2.1.0,>=2.0.0 in c:\users\asg\.conda\envs\lambda\lib\s
ite-packages (from ipython->shap==0.23.0) (2.0.9)
Requirement already satisfied: decorator in c:\users\asg\.conda\envs\lambda\lib\site-packages (from
ipython->shap==0.23.0) (4.4.0)
Requirement already satisfied: pygments in c:\users\asg\.conda\envs\lambda\lib\site-packages (from i
python->shap==0.23.0) (2.4.2)
Requirement already satisfied: colorama; sys_platform == "win32" in c:\users\asg\.conda\envs\lambda
\lib\site-packages (from ipython->shap==0.23.0) (0.4.1)
Requirement already satisfied: python-dateutil>=2.5.0 in c:\users\asg\.conda\envs\lambda\lib\site-pa
ckages (from pandas->shap==0.23.0) (2.8.0)
Requirement already satisfied: pytz>=2011k in c:\users\asg\.conda\envs\lambda\lib\site-packages (fro
m pandas->shap==0.23.0) (2019.2)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\asg\.conda\envs\lambda\lib\site-package
s (from matplotlib->shap==0.23.0) (1.1.0)
Requirement already satisfied: cycler>=0.10 in c:\users\asg\.conda\envs\lambda\lib\site-packages (fr
om matplotlib->shap==0.23.0) (0.10.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in c:\users\asg\.conda\envs
\lambda\lib\site-packages (from matplotlib->shap==0.23.0) (2.4.2)
Requirement already satisfied: ipython-genutils in c:\users\asg\.conda\envs\lambda\lib\site-packages
(from traitlets>=4.2->ipython->shap==0.23.0) (0.2.0)
Requirement already satisfied: six in c:\users\asg\.conda\envs\lambda\lib\site-packages (from traitl
ets>=4.2->ipython->shap==0.23.0) (1.12.0)
Requirement already satisfied: parso>=0.5.0 in c:\users\asg\.conda\envs\lambda\lib\site-packages (fr
om jedi>=0.10->ipython->shap==0.23.0) (0.5.1)
Requirement already satisfied: wcwidth in c:\users\asg\.conda\envs\lambda\lib\site-packages (from pr
ompt-toolkit<2.1.0,>=2.0.0->ipython->shap==0.23.0) (0.1.7)
```

In [ ]:
```
# Contributrions to making bin 8 (49 - 100 cigarettes per day) for sample 170
import shap

row = X_val.iloc[[170]]

explainer = shap.TreeExplainer(model)
row_processed = processor.transform(row)
shap_values_input = explainer.shap_values(row_processed)

shap.initjs()
shap.force_plot(
    base_value=explainer.expected_value[7],
    shap_values=shap_values_input[7],
    features=row
)
```

In [0]:
```
# Featues importances for sample 170

feature_names = row.columns
feature_values = row.values[0]
shap_values_array = np.asarray(shap_values_input)
shaps = pd.Series(shap_values_array[0,0,:], zip(feature_names, feature_values))
shaps.sort_values().plot.barh(color='grey', figsize=(10,15));
```



In [0]:
```
# Create a dataframe for sample 170
# bin versus feature

my_python_list = [shap_values_array[0, 0, :], shap_values_array[1, 0, :], shap_values_array[2, 0,
:], shap_values_array[3, 0, :], shap_values_array[4, 0, :], shap_values_array[5, 0, :], shap_values_
array[6, 0, :], shap_values_array[7, 0, :]]
df_bins = pd.DataFrame(columns=np.array(feature_names), data=my_python_list)

df_bins.head(8)
```

Out[0]:

	language	cereal_serve_per_month	cereal_times_per_month	more_than_one_cereal_type	milk_serve_per_month	milk_times
0	-0.078904	-0.001079	0.000000	-0.001774	-0.213007	
1	0.009190	-0.028916	0.000124	0.000488	0.137537	
2	0.038114	0.189786	0.001831	0.000429	0.065633	
3	-0.002156	0.002513	0.000000	-0.000402	0.008714	
4	-0.000780	-0.026380	0.001205	-0.000794	0.018316	
5	-0.000124	0.035602	0.021127	-0.000245	-0.004743	
6	0.000000	0.000250	0.000000	0.000000	0.000000	
7	-0.000721	-0.002467	0.001319	0.000119	0.030396	

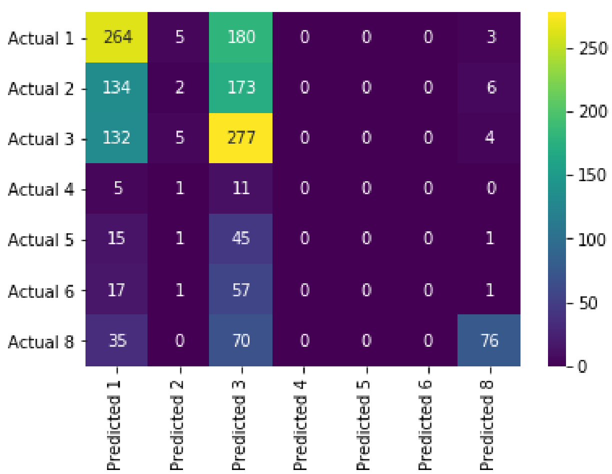
In [0]:
```
# Create a 3D plot of force as a function of cigarettes_per_day_bin and feature for sample 170
# A two feature partical dependence plot in 3D
import plotly.graph_objs as go

surface = go.Surface(x=df_bins.columns,
                     y=df_bins.index + 1,
                     z=df_bins.values)

layout = go.Layout(
    scene=dict(
    xaxis=dict(title= 'Features'),
    yaxis=dict(title= 'cigarettes_per_day_bin'),
    zaxis=dict(title= 'Force')
    )
)
fig = go.Figure(surface, layout)
fig.show()
```

In [0]:
```
pros = shaps.sort_values(ascending=False)[:3].index
cons = shaps.sort_values(ascending=True)[:3].index

print('Pros:')
for i, pro in enumerate(pros, start=1):
    feature_name, feature_value = pro
    print(f'{i}. {feature_name} is {feature_value}')
print('\n')

print('Cons:')
for i, con in enumerate(cons, start=1):
    feature_name, feature_value = con
    print(f'{i}. {feature_name} is {feature_value}')
```

```
Pros:
1. soda_times_per_month is 3.0
2. see_walking_from_home is 1.0
3. walk_number_wk is 4.0


Cons:
1. milk_serve_per_month is 25.0
2. walkable_retail is 0.0
3. language is 5.0
```

In [0]:
```
# Create function for constructing confusion matrix
%matplotlib inline
import seaborn as sns
from sklearn.metrics import confusion_matrix
from sklearn.utils.multiclass import unique_labels
def plot_confusion_matrix(y_true, y_pred):
    labels = unique_labels(y_true)
    columns = [f'Predicted {label}' for label in labels]
    index = [f'Actual {label}' for label in labels]
    table = pd.DataFrame(confusion_matrix(y_true, y_pred),
    columns=columns, index=index)
    return sns.heatmap(table, annot=True, fmt='d', cmap='viridis')
```

In [0]:
```
y_pred = pipeline.predict(X_val)
plot_confusion_matrix(y_val, y_pred);
```



In [0]:
```
# Get precision & recall for majority class baseline
from sklearn.metrics import classification_report
print(classification_report(y_val, y_pred))
```

```
              precision    recall  f1-score   support

           1       0.44      0.58      0.50       452
           2       0.13      0.01      0.01       315
           3       0.34      0.66      0.45       418
           4       0.00      0.00      0.00        17
           5       0.00      0.00      0.00        62
           6       0.00      0.00      0.00        76
           8       0.84      0.42      0.56       181

    accuracy                           0.41      1521
   macro avg       0.25      0.24      0.22      1521
weighted avg       0.35      0.41      0.34      1521
```

In [0]:
```
# Another way to get a classification report using an ROC_AUC approach (https://stackoverflow.com/qu
estions/39685740/calculate-sklearn-roc-auc-score-for-multi-class?rq=1),
import pandas as pd
import numpy as np
from scipy import interp

from  sklearn.metrics import precision_recall_fscore_support
from sklearn.metrics import roc_curve, auc
from sklearn.preprocessing import LabelBinarizer

def class_report(y_true, y_pred, y_score=None, average='micro'):
    if y_true.shape != y_pred.shape:
        print("Error! y_true %s is not the same shape as y_pred %s" % (
              y_true.shape,
              y_pred.shape)
        )
        return

    lb = LabelBinarizer()

    if len(y_true.shape) == 1:
        lb.fit(y_true)

    #Value counts of predictions
    labels, cnt = np.unique(
        y_pred,
        return_counts=True)
    n_classes = len(labels)
    pred_cnt = pd.Series(cnt, index=labels)

    metrics_summary = precision_recall_fscore_support(
            y_true=y_true,
            y_pred=y_pred,
            labels=labels)

    avg = list(precision_recall_fscore_support(
            y_true=y_true,
            y_pred=y_pred,
            average='weighted'))

    metrics_sum_index = ['precision', 'recall', 'f1-score', 'support']
    class_report_df = pd.DataFrame(
        list(metrics_summary),
        index=metrics_sum_index,
        columns=labels)

    support = class_report_df.loc['support']
    total = support.sum()
    class_report_df['avg / total'] = avg[:-1] + [total]

    class_report_df = class_report_df.T
    class_report_df['pred'] = pred_cnt
    class_report_df['pred'].iloc[-1] = total

    if not (y_score is None):
        fpr = dict()
        tpr = dict()
        roc_auc = dict()
        for label_it, label in enumerate(labels):
            fpr[label], tpr[label], _ = roc_curve(
                (y_true == label).astype(int),
                y_score[:, label_it])

            roc_auc[label] = auc(fpr[label], tpr[label])

        if average == 'micro':
            if n_classes <= 2:
                fpr["avg / total"], tpr["avg / total"], _ = roc_curve(
                    lb.transform(y_true).ravel(),
                    y_score[:, 1].ravel())
```

```
            else:
                fpr["avg / total"], tpr["avg / total"], _ = roc_curve(
                        lb.transform(y_true).ravel(),
                        y_score.ravel())

            roc_auc["avg / total"] = auc(
                fpr["avg / total"],
                tpr["avg / total"])

        elif average == 'macro':
            # First aggregate all false positive rates
            all_fpr = np.unique(np.concatenate([
                fpr[i] for i in labels]
            ))

            # Then interpolate all ROC curves at this points
            mean_tpr = np.zeros_like(all_fpr)
            for i in labels:
                mean_tpr += interp(all_fpr, fpr[i], tpr[i])

            # Finally average it and compute AUC
            mean_tpr /= n_classes

            fpr["macro"] = all_fpr
            tpr["macro"] = mean_tpr

            roc_auc["avg / total"] = auc(fpr["macro"], tpr["macro"])

        class_report_df['AUC'] = pd.Series(roc_auc)

    return class_report_df
```

In [0]:

```
# The above function provides the predicted values for each class.
class_report(y_val, y_pred, y_score=None, average='micro')
```

Out[0]:

	precision	recall	f1-score	support	pred
1	0.438538	0.584071	0.500949	452.0	602.0
2	0.133333	0.006349	0.012121	315.0	15.0
3	0.340713	0.662679	0.450041	418.0	813.0
8	0.835165	0.419890	0.558824	181.0	91.0
avg / total	0.350955	0.406969	0.341559	1366.0	1366.0

In [0]:
```
# Deriving an ROC curve for each class in cigarettes_per_day_bins
# Transform y_val and y_pred to arrays that are 1521 by 8 with bins as the columns

y_val_trans = pd.DataFrame(columns=['1','2','3','4','5','6','7', '8'])
y_val_trans['1']=y_val.map(lambda x : 1 if x==1 else 0)
y_val_trans['2']=y_val.map(lambda x : 1 if x==2 else 0)
y_val_trans['3']=y_val.map(lambda x : 1 if x==3 else 0)
y_val_trans['4']=y_val.map(lambda x : 1 if x==4 else 0)
y_val_trans['5']=y_val.map(lambda x : 1 if x==5 else 0)
y_val_trans['6']=y_val.map(lambda x : 1 if x==6 else 0)
y_val_trans['7']=y_val.map(lambda x : 1 if x==7 else 0)
y_val_trans['8']=y_val.map(lambda x : 1 if x==8 else 0)
print ('y_val_trans =')
print (y_val_trans.head(), '\n')

y_pred_proba = model.predict_proba(X_val)

y_pred_trans = pd.DataFrame(y_pred_proba)

print ('y_pred_trans')
print (y_pred_trans.head(), '\n')
```

```
y_val_trans =
       1  2  3  4  5  6  7  8
31502  0  0  1  0  0  0  0  0
4439   1  0  0  0  0  0  0  0
27082  0  1  0  0  0  0  0  0
19317  0  1  0  0  0  0  0  0
2063   0  0  0  0  1  0  0  0

y_pred_trans
          0         1         2         3         4         5         6  \
0  0.134439  0.211467  0.243504  0.060497  0.109879  0.090755  0.056505
1  0.230874  0.172901  0.206736  0.056917  0.087410  0.071219  0.055634
2  0.192338  0.205295  0.260232  0.056505  0.080505  0.059474  0.054726
3  0.042977  0.071380  0.093560  0.043139  0.043097  0.042417  0.042513
4  0.234252  0.118815  0.181867  0.055755  0.112610  0.161509  0.052913

          7
0  0.092955
1  0.118309
2  0.090925
3  0.620918
4  0.082279
```

In [0]:
```
# Learn to predict each class against the other
print(__doc__)

import numpy as np

from sklearn import svm, datasets
from sklearn.metrics import roc_curve, auc

# Compute ROC curve and ROC area for each class
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(8):
    fpr[i], tpr[i], _ = roc_curve(y_val_trans.iloc[:, i], y_pred_trans.iloc[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

# Compute micro-average ROC curve and ROC area
fpr["micro"], tpr["micro"], _ = roc_curve(y_val_trans.values.ravel(), y_pred_trans.values.ravel())
roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])
```

```
Automatically created module for IPython interactive environment
```

In [0]:

```
# Compute macro-average ROC curve and ROC area
import matplotlib.pyplot as plt
from itertools import cycle
from scipy import interp
n_classes = 8
lw = 2

# First aggregate all false positive rates
all_fpr = np.unique(np.concatenate([fpr[i] for i in range(n_classes)]))

# Then interpolate all ROC curves at this points
mean_tpr = np.zeros_like(all_fpr)
for i in range(n_classes):
    mean_tpr += interp(all_fpr, fpr[i], tpr[i])

# Finally average it and compute AUC
mean_tpr /= n_classes

fpr["macro"] = all_fpr
tpr["macro"] = mean_tpr
roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])

# Plot all ROC curves
plt.figure()
plt.plot(fpr["micro"], tpr["micro"],
         label='micro-average ROC curve (area = {0:0.2f})'
               ''.format(roc_auc["micro"]),
         color='deeppink', linestyle=':', linewidth=4)

plt.plot(fpr["macro"], tpr["macro"],
         label='macro-average ROC curve (area = {0:0.2f})'
               ''.format(roc_auc["macro"]),
         color='navy', linestyle=':', linewidth=4)

colors = cycle(['aqua', 'darkorange', 'cornflowerblue', 'blue', 'green'])
for i, color in zip(range(n_classes), colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=lw,
             label='ROC curve of class {0} (area = {1:0.2f})'
             ''.format(i + 1, roc_auc[i]))

plt.plot([0, 1], [0, 1], 'k--', lw=lw)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Some extension of Receiver operating characteristic to multi-class')
plt.legend(loc="lower right")
plt.show()
```