```
import sys
in_colab = 'google.colab' in sys.modules
if in_colab:
    # Install packages in Colab
    !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 /usr/local/lib/python3.6/dist-packages (2.0.0)
Requirement already satisfied: scipy>=0.19.0 in /usr/local/lib/python3.6/dist-packages (from category_encoders=
Requirement already satisfied: patsy>=0.4.1 in /usr/local/lib/python3.6/dist-packages (from category_encoders==
Requirement already satisfied: numpy>=1.11.3 in /usr/local/lib/python3.6/dist-packages (from category_encoders=
Requirement already satisfied: statsmodels>=0.6.1 in /usr/local/lib/python3.6/dist-packages (from category_enco
Requirement already satisfied: pandas>=0.21.1 in /usr/local/lib/python3.6/dist-packages (from category_encoders
Requirement already satisfied: scikit-learn>=0.20.0 in /usr/local/lib/python3.6/dist-packages (from category_en
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from patsy>=0.4.1->category_encod
Requirement already satisfied: pytz>=2011k in /usr/local/lib/python3.6/dist-packages (from pandas>=0.21.1->cate
Requirement already satisfied: python-dateutil>=2.5.0 in /usr/local/lib/python3.6/dist-packages (from pandas>=0
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.6/dist-packages (from scikit-learn>=0.20.
Collecting pandas-profiling==2.3.0
Requirement already satisfied: astropy in /usr/local/lib/python3.6/dist-packages (from pandas-profiling==2.3.0)
Requirement already satisfied: missingno>=0.4.2 in /usr/local/lib/python3.6/dist-packages (from pandas-profilin
Collecting confuse>=1.0.0 (from pandas-profiling==2.3.0)
Requirement already satisfied: jinja2>=2.8 in /usr/local/lib/python3.6/dist-packages (from pandas-profiling==2.
Collecting phik>=0.9.8 (from pandas-profiling==2.3.0)
  Using cached https://files.pythonhosted.org/packages/45/ad/24a16fa4ba612fb96a3c4bb115a5b9741483f53b66d3d3afd9
Requirement already satisfied: pandas>=0.19 in /usr/local/lib/python3.6/dist-packages (from pandas-profiling==2
Requirement already satisfied: matplotlib>=1.4 in /usr/local/lib/python3.6/dist-packages (from pandas-profiling
Requirement already satisfied: htmlmin>=0.1.12 in /usr/local/lib/python3.6/dist-packages (from pandas-profiling
Requirement already satisfied: numpy>=1.10.0 in /usr/local/lib/python3.6/dist-packages (from astropy->pandas-pr
Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packages (from missingno>=0.4.2->pandas-p
Requirement already satisfied: seaborn in /usr/local/lib/python3.6/dist-packages (from missingno>=0.4.2->pandas
Requirement already satisfied: pyyaml in /usr/local/lib/python3.6/dist-packages (from confuse>=1.0.0->pandas-pr
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.6/dist-packages (from jinja2>=2.8->pa
Requirement already satisfied: nbconvert>=5.3.1 in /usr/local/lib/python3.6/dist-packages (from phik>=0.9.8->pa
Collecting pytest-pylint>=0.13.0 (from phik>=0.9.8->pandas-profiling==2.3.0)
  Using cached https://files.pythonhosted.org/packages/64/dc/6f35f114844fb12e38d60c4f3d2441a55baff7043ad4e01377
Requirement already satisfied: jupyter-client>=5.2.3 in /usr/local/lib/python3.6/dist-packages (from phik>=0.9.
Collecting pytest>=4.0.2 (from phik>=0.9.8->pandas-profiling==2.3.0)
  Using cached https://files.pythonhosted.org/packages/0c/91/d68f68ce54cd3e8afa1ef73ea1ad44df2438521b64c0820e5f
Requirement already satisfied: numba>=0.38.1 in /usr/local/lib/python3.6/dist-packages (from phik>=0.9.8->panda
Requirement already satisfied: python-dateutil>=2.5.0 in /usr/local/lib/python3.6/dist-packages (from pandas>=0
Requirement already satisfied: pytz>=2011k in /usr/local/lib/python3.6/dist-packages (from pandas>=0.19->pandas
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib>=1.
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.6/dist-packages (from matplotlib>=1.4->pa
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.6/dist-packag
Requirement already satisfied: bleach in /usr/local/lib/python3.6/dist-packages (from nbconvert>=5.3.1->phik>=0
Requirement already satisfied: jupyter-core in /usr/local/lib/python3.6/dist-packages (from nbconvert>=5.3.1->p
Requirement already satisfied: traitlets>=4.2 in /usr/local/lib/python3.6/dist-packages (from nbconvert>=5.3.1-
Requirement already satisfied: pandocfilters>=1.4.1 in /usr/local/lib/python3.6/dist-packages (from nbconvert>=
Requirement already satisfied: testpath in /usr/local/lib/python3.6/dist-packages (from nbconvert>=5.3.1->phik>
Requirement already satisfied: pygments in /usr/local/lib/python3.6/dist-packages (from nbconvert>=5.3.1->phik>
Requirement already satisfied: defusedxml in /usr/local/lib/python3.6/dist-packages (from nbconvert>=5.3.1->phi
Requirement already satisfied: mistune<2,>=0.8.1 in /usr/local/lib/python3.6/dist-packages (from nbconvert>=5.3
Requirement already satisfied: nbformat>=4.4 in /usr/local/lib/python3.6/dist-packages (from nbconvert>=5.3.1->
Requirement already satisfied: entrypoints>=0.2.2 in /usr/local/lib/python3.6/dist-packages (from nbconvert>=5.
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from pytest-pylint>=0.13.0->phik>
Collecting pylint>=1.4.5 (from pytest-pylint>=0.13.0->phik>=0.9.8->pandas-profiling==2.3.0)
  Using cached https://files.pythonhosted.org/packages/ea/f1/758de486e46ea2b8717992704b0fdd968b7cbc2bc790b976fa
Requirement already satisfied: pyzmq>=13 in /usr/local/lib/python3.6/dist-packages (from jupyter-client>=5.2.3-
Requirement already satisfied: tornado>=4.1 in /usr/local/lib/python3.6/dist-packages (from jupyter-client>=5.2
Requirement already satisfied: py>=1.5.0 in /usr/local/lib/python3.6/dist-packages (from pytest>=4.0.2->phik>=0
Requirement already satisfied: importlib-metadata>=0.12; python_version < "3.8" in /usr/local/lib/python3.6/dis
Requirement already satisfied: packaging in /usr/local/lib/python3.6/dist-packages (from pytest>=4.0.2->phik>=0
Requirement already satisfied: attrs>=17.4.0 in /usr/local/lib/python3.6/dist-packages (from pytest>=4.0.2->phi
Requirement already satisfied: pluggy<1.0,>=0.12 in /usr/local/lib/python3.6/dist-packages (from pytest>=4.0.2-
Requirement already satisfied: more-itertools>=4.0.0 in /usr/local/lib/python3.6/dist-packages (from pytest>=4.
Requirement already satisfied: wcwidth in /usr/local/lib/python3.6/dist-packages (from pytest>=4.0.2->phik>=0.9
Requirement already satisfied: atomicwrites>=1.0 in /usr/local/lib/python3.6/dist-packages (from pytest>=4.0.2-
Requirement already satisfied: llvmlite>=0.25.0dev0 in /usr/local/lib/python3.6/dist-packages (from numba>=0.38
Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-packages (from kiwisolver>=1.0.1->ma
Requirement already satisfied: webencodings in /usr/local/lib/python3.6/dist-packages (from bleach->nbconvert>=
Requirement already satisfied: decorator in /usr/local/lib/python3.6/dist-packages (from traitlets>=4.2->nbconv
Requirement already satisfied: ipython-genutils in /usr/local/lib/python3.6/dist-packages (from traitlets>=4.2-
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in /usr/local/lib/python3.6/dist-packages (from nbformat
Collecting mccabe<0.7,>=0.6 (from pylint>=1.4.5->pytest-pylint>=0.13.0->phik>=0.9.8->pandas-profiling==2.3.0)
  Using cached https://files.pythonhosted.org/packages/87/89/479dc97e18549e21354893e4ee4ef36db1d237534982482c36
Collecting astroid<2.4,>=2.3.0 (from pylint>=1.4.5->pytest-pylint>=0.13.0->phik>=0.9.8->pandas-profiling==2.3.0
  Using cached https://files.pythonhosted.org/packages/64/d3/4ba68bd56297556c9c2e5072d71d1664feaa86d9726c237a9f
```

```
Collecting isort<5,>=4.2.5 (from pylint>=1.4.5->pytest-pylint>=0.13.0->phik>=0.9.8->pandas-profiling==2.3.0)
  Using cached https://files.pythonhosted.org/packages/e5/b0/c121fd1fa3419ea9bfd55c7f9c4fedfec5143208d8c7ad3ce3
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.6/dist-packages (from importlib-metadata>=0.
Requirement already satisfied: wrapt==1.11.* in /usr/local/lib/python3.6/dist-packages (from astroid<2.4,>=2.3.
Collecting lazy-object-proxy==1.4.* (from astroid<2.4,>=2.3.0->pylint>=1.4.5->pytest-pylint>=0.13.0->phik>=0.9.
  Using cached https://files.pythonhosted.org/packages/0e/26/534a6d32572a9dbca11619321535c0a7ab34688545d9d67c2c
Collecting typed-ast<1.5,>=1.4.0; implementation_name == "cpython" and python_version < "3.8" (from astroid<2.4
  Using cached https://files.pythonhosted.org/packages/31/d3/9d1802c161626d0278bafb1ffb32f76b9d01e123881bbf9d91
ERROR: datascience 0.10.6 has requirement folium==0.2.1, but you'll have folium 0.8.3 which is incompatible.
Installing collected packages: confuse, pytest, mccabe, lazy-object-proxy, typed-ast, astroid, isort, pylint, p
  Found existing installation: pytest 3.6.4
    Uninstalling pytest-3.6.4:
      Successfully uninstalled pytest-3.6.4
  Found existing installation: pandas-profiling 1.4.1
    Uninstalling pandas-profiling-1.4.1:
      Successfully uninstalled pandas-profiling-1.4.1
Successfully installed astroid-2.3.2 confuse-1.0.0 isort-4.3.21 lazy-object-proxy-1.4.2 mccabe-0.6.1 pandas-pro
Requirement already satisfied: plotly==4.1.1 in /usr/local/lib/python3.6/dist-packages (4.1.1)
Requirement already satisfied: retrying>=1.3.3 in /usr/local/lib/python3.6/dist-packages (from plotly==4.1.1) (
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from plotly==4.1.1) (1.12.0)
```

```python
#Fetch smoking data file
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session

```
Saving cancerxx - for_import.csv to cancerxx - for_import.csv
```

```python
# Load smoking data
import pandas as pd
import io
df_smoking = pd.read_csv(io.StringIO(uploaded['cancerxx - for_import.csv'].decode('utf-8')))
df_smoking.head()
```

	language	cereal_serve_per_month	cereal_times_per_month	more_than_one_cereal_type	milk_serve_per_month
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

```python
# 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
```

```
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  ...  cigarettes_per_day
count  33672.000000  ...         7602.000000
mean       4.670587  ...           22.540647
std        1.191156  ...           26.525465
min        1.000000  ...            1.000000
25%        4.000000  ...            6.000000
50%        5.000000  ...           15.000000
75%        5.000000  ...           20.000000
max        9.000000  ...           99.000000
```

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

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

	language	cereal_serve_per_month	cereal_times_per_month	more_than_one_cereal_type	milk_serve_per_month
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

```
# 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_
            'walkway_existence', 'walkable_retail', 'walkable_bus_stop', 'walkable_entertainment', 'walkable_relaxation', 's
            'crime_discourages_walking', 'animals_discourage_walking', 'cigarette_even_once', 'cigar_even_once', 'pipe_even_
            '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[features1].replace(replacements1)

df_smoking2.head()
```

	walk_leisure_past_wk	cigarette_even_once	streets_have_walkways	walk_past_wk	pipe_even_once	walkable_bu
0	1.0	0	0	0	0	
1	1.0	0	1	0	0	
2	1.0	0	1	0	0	
3	1.0	1	1	0	0	
4	0.0	0	1	0	0	

```
df_smoking1['number'] = df_smoking1.index
df_smoking2['number'] = df_smoking2.index

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

	language	cereal_serve_per_month	cereal_times_per_month	more_than_one_cereal_type	milk_serve_per_month
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

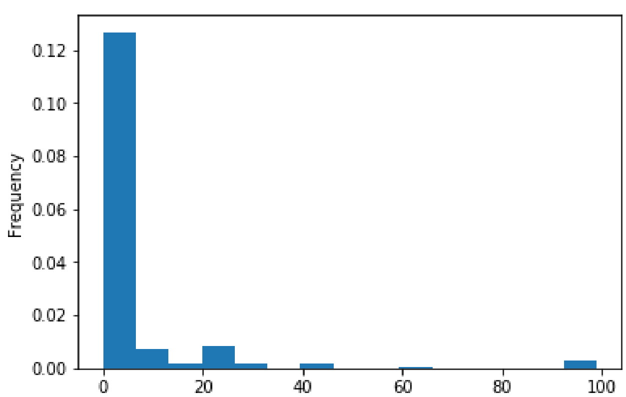
```
df_smoking1 = df_smoking1.drop('number', axis = 1)
df_smoking1.head()
```

	language	cereal_serve_per_month	cereal_times_per_month	more_than_one_cereal_type	milk_serve_per_month
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

```
# 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');
```

```
/usr/local/lib/python3.6/dist-packages/matplotlib/axes/_axes.py:6521: MatplotlibDeprecationWarning:
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed in 3.1. Use 'density' instead.
  alternative="'density'", removal="3.1")
```

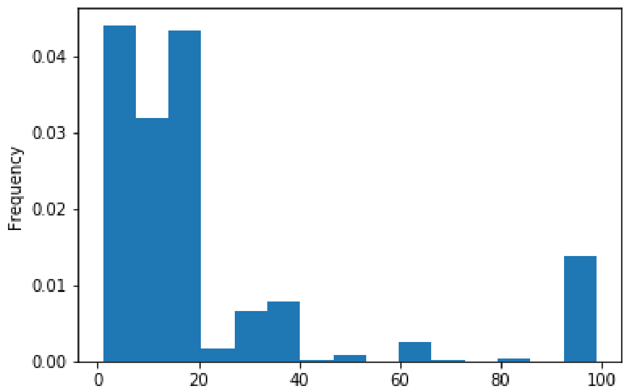



```
# Drop rows where cigarettes_per_day = 0
df_smoking1['cigarettes_per_day'] =  df_smoking1['cigarettes_per_day'].replace ({np.NaN: 0})
df_smoking1 = df_smoking1[df_smoking1['cigarettes_per_day'] > 0]
df_smoking1.shape
```

```
(7602, 92)
```

```
# 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');
```

```
/usr/local/lib/python3.6/dist-packages/matplotlib/axes/_axes.py:6521: MatplotlibDeprecationWarning:
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed in 3.1. Use 'density' instead.
  alternative="'density'", removal="3.1")
```




```
# 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
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()
```

	language	cereal_serve_per_month	cereal_times_per_month	more_than_one_cereal_type	milk_serve_per_month
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

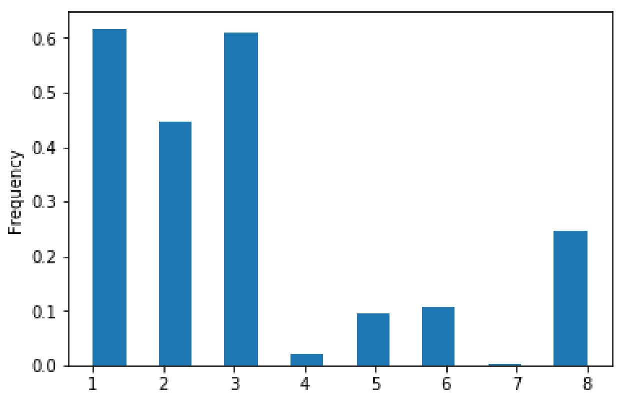
```
# Feature Engineering

# walk_leisure_distance_week = walking_leisure_distance * walk_number_week
df_smoking1['walk_leisure_distance_week'] = df_smoking1['walk_leisure_ distance']*df_smoking1['walk_number_wk']

# single_walk_distance_week = single_walk_distance * walk_number_week
df_smoking1['single_walk_distance_week'] = df_smoking1['single_walk_distance']*df_smoking1['walk_number_wk']

# tobacco_even_once = cigarette_even_once + cigar_even_once + smokeless_even_once
df_smoking1['tobacco_even_once'] = df_smoking1['cigarette_even_once'] + df_smoking1['cigar_even_once'] + df_smoking1['smokel

# red_meat_eat_serve_per_time = red_meat_eat_serve_month / red_meat_eat_times_month
df_smoking1['red_meat_eat_serve_per_time'] = df_smoking1['red_meat_eat_serve_per_month']/df_smoking1['red_meat_eat_times_per_

# bread_eat_serve_per_time = bread_eat_serve_month / bread_eat_times_month
df_smoking1['bread_eat_serve_per_time'] = df_smoking1['bread_eat_serve_per_month']/df_smoking1['bread_eat_times_per_month']

df_smoking1.head()
```

	language	cereal_serve_per_month	cereal_times_per_month	more_than_one_cereal_type	milk_serve_per_month
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 × 97 columns

```
# 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')
```

```
/usr/local/lib/python3.6/dist-packages/matplotlib/axes/_axes.py:6521: MatplotlibDeprecationWarning:
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed in 3.1. Use 'density' instead.
  alternative="'density'", removal="3.1")
Text(0, 0.5, 'Frequency')
```




```
# 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_

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

```
((6081, 96), (6081,), (1521, 96), (1521,))
```

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

	language	cereal_serve_per_month	cereal_times_per_month	more_than_one_ce
language	1.000000	0.436982	0.351576	
cereal_serve_per_month	0.436982	1.000000	0.760684	
cereal_times_per_month	0.351576	0.760684	1.000000	
more_than_one_cereal_type	-0.035361	-0.138573	0.103886	
milk_serve_per_month	0.433675	0.972695	0.735602	
milk_times_per_month	0.349838	0.769347	0.739144	
milk_type	-0.096036	-0.232121	-0.007578	
soda_serve_per_month	0.431958	0.959336	0.721514	
soda_times_per_month	0.342304	0.734191	0.595590	
juice_serve_per_month	0.428804	0.956040	0.720313	
juice_times_per_month	0.332304	0.727421	0.597924	
coffee_serve_per_month	0.426747	0.951691	0.714146	
coffee_times_per_month	0.333119	0.801064	0.622032	
sports_drink_serve_per_month	0.432197	0.957457	0.718121	
sports_drink_times_per_month	0.359200	0.808602	0.625149	
fruit_drink_serve_per_month	0.431355	0.952001	0.713791	
fruit_drink_times_per_month	0.358626	0.798001	0.620712	
fruit_eat_serve_per_month	0.425964	0.957833	0.721305	
fruit_eat_times_per_month	0.384347	0.806646	0.658352	
salad_eat_serve_per_month	0.427673	0.950363	0.713253	
salad_eat_times_per_month	0.382662	0.789765	0.644858	
fries_eat_serve_per_month	0.425416	0.950622	0.710713	
fries_eat_times_per_month	0.361141	0.706499	0.579918	
potatoe_eat_serve_per_month	0.422435	0.936681	0.699211	
potatoe_eat_times_per_month	0.375218	0.743602	0.606211	
beans_eat_serve_per_month	0.421520	0.935026	0.698968	
beans_eat_times_per_month	0.334060	0.704172	0.577761	
grains_eat_serve_per_month	0.422670	0.940141	0.701947	
grains_eat_times_per_month	0.352108	0.698946	0.547232	
vegies_eat_serve_per_month	0.415677	0.928090	0.693861	
vegies_eat_times_per_month	0.359752	0.801514	0.632530	
salsa_eat_serve_per_month	0.421930	0.932706	0.695506	
salsa_eat_times_per_month	0.332938	0.678452	0.541066	
pizza_eat_serve_per_month	0.422585	0.938145	0.699300	
pizza_eat_times_per_month	0.358019	0.679303	0.546140	
tomatoe_eat_serve_per_month	0.418889	0.930008	0.692785	
tomatoe_eat_times_per_month	0.360487	0.700663	0.569326	

cheese_eat_serve_per_month	0.417031	0.926477	0.691735
cheese_eat_times_per_month	0.363737	0.769202	0.610668
red_meat_eat_serve_per_month	0.419657	0.929806	0.694151
red_meat_eat_times_per_month	0.376608	0.780559	0.615793
processed_meat_eat_serve_per_month	0.418972	0.928255	0.692179
processed_meat_eat_times_per_month	0.373554	0.707912	0.571415
bread_eat_serve_per_month	0.417267	0.923150	0.689785
bread_eat_times_per_month	0.339279	0.735331	0.595573
candy_eat_serve_per_month	0.411998	0.922073	0.689743
candy_eat_times_per_month	0.372756	0.707072	0.583550
donut_eat_serve_per_month	0.416284	0.926723	0.690687
donut_eat_times_per_month	0.334741	0.680731	0.556009
cookie_eat_serve_per_month	0.409480	0.912101	0.677290
cookie_eat_times_per_month	0.355908	0.682247	0.559441
ice_cream_eat_serve_per_month	0.414443	0.918537	0.683445
ice_cream_eat_times_per_month	0.350857	0.677407	0.552084
pop_corn_eat_serve_per_month	0.415217	0.921843	0.687277
pop_corn_eat_times_per_month	0.354492	0.669004	0.529327
vitamin_past_month	-0.050629	-0.243404	-0.157238
multivitamin_past_month	-0.037872	-0.162842	-0.096123
multivitamin_days_in_month	-0.029406	-0.150437	-0.089361
calcium_past_month	-0.040267	-0.096730	-0.061498
calcium_days_in_month	-0.034379	-0.086469	-0.060933
vitD_past_month	-0.016192	-0.122617	-0.076643
vitD_days_in_month	-0.013972	-0.111407	-0.068578
vitD_reason	-0.011984	-0.099275	-0.061147
1st_kind_cereal_eaten	-0.066491	-0.213615	0.202229
2nd_kind_cereal_eaten	-0.021112	-0.118378	0.093967
walk_past_wk	-0.100718	-0.114823	-0.085251
walk_number_wk	-0.049873	-0.039521	-0.041604
single_walk_distance	-0.015167	-0.034909	-0.037080
single_walk_time	-0.075258	-0.097345	-0.084728
walk_leisure_past_wk	-0.077325	-0.188538	-0.135776
walk_leisure_number_wk	-0.026543	-0.105001	-0.087298
walk_leisure_ distance	-0.026035	-0.067584	-0.044969
walk_leisure_ time	-0.061651	-0.163052	-0.120797
see_walking_from_home	0.322965	0.612504	0.441254
weather_discourages_walk	0.214795	0.481079	0.334835
walkway_existence	-0.203418	-0.385381	-0.283120

walkable_retail	-0.159764	-0.199678	-0.134860
walkable_bus_stop	-0.188837	-0.181334	-0.142217
walkable_entertainment	-0.150265	-0.176244	-0.124428
walkable_relaxation	-0.141028	-0.274859	-0.193180
streets_have_walkways	-0.188904	-0.217642	-0.159778
traffic_discourages_walking	-0.093775	-0.097254	-0.076176
crime_discourages_walking	-0.096958	-0.069252	-0.066612
animals_discourage_walking	-0.069518	-0.061819	-0.047632
cigarette_even_once	-0.014661	-0.082766	-0.060123
cigar_even_once	0.017100	-0.156603	-0.099829
pipe_even_once	0.021861	-0.104214	-0.052365
smokeless_even_once	0.036964	-0.087348	-0.057695
had_genetic_counseling	-0.011091	-0.026606	-0.011029
genetic_counseling_with_MD	-0.021622	-0.039074	-0.013490
genetic_counseling_for_cancer	-0.015048	-0.023971	-0.022560
walk_leisure_distance_week	-0.020607	-0.037062	-0.034939
single_walk_distance_week	-0.009937	-0.026258	-0.036861
tobacco_even_once	0.019907	-0.162235	-0.107231
red_meat_eat_serve_per_time	0.428273	0.926809	0.701372
bread_eat_serve_per_time	0.448476	0.925059	0.727886

```
# 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 dataset

correlation(XTrain, XVal, 0.98)

XTrain.shape
XVal.shape
```

```
(1521, 81)
```

```
# 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)
```

```
      3    0.286466
```

```
# 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)
```

```
0.2864660417694458
```

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

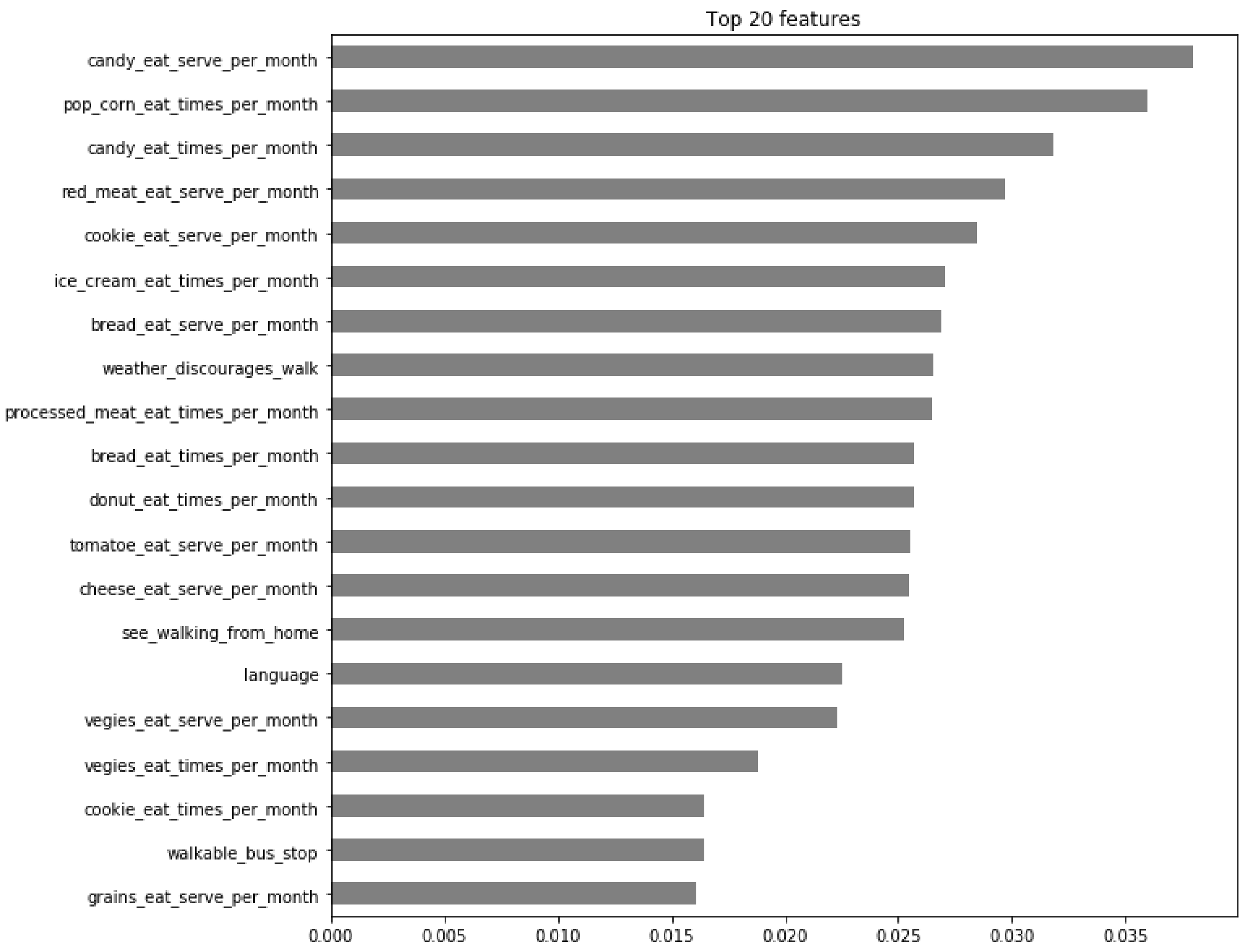
```
# Optimizing Hyperparameters
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier

# 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, 0.61

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)
```

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

```
# 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))
```

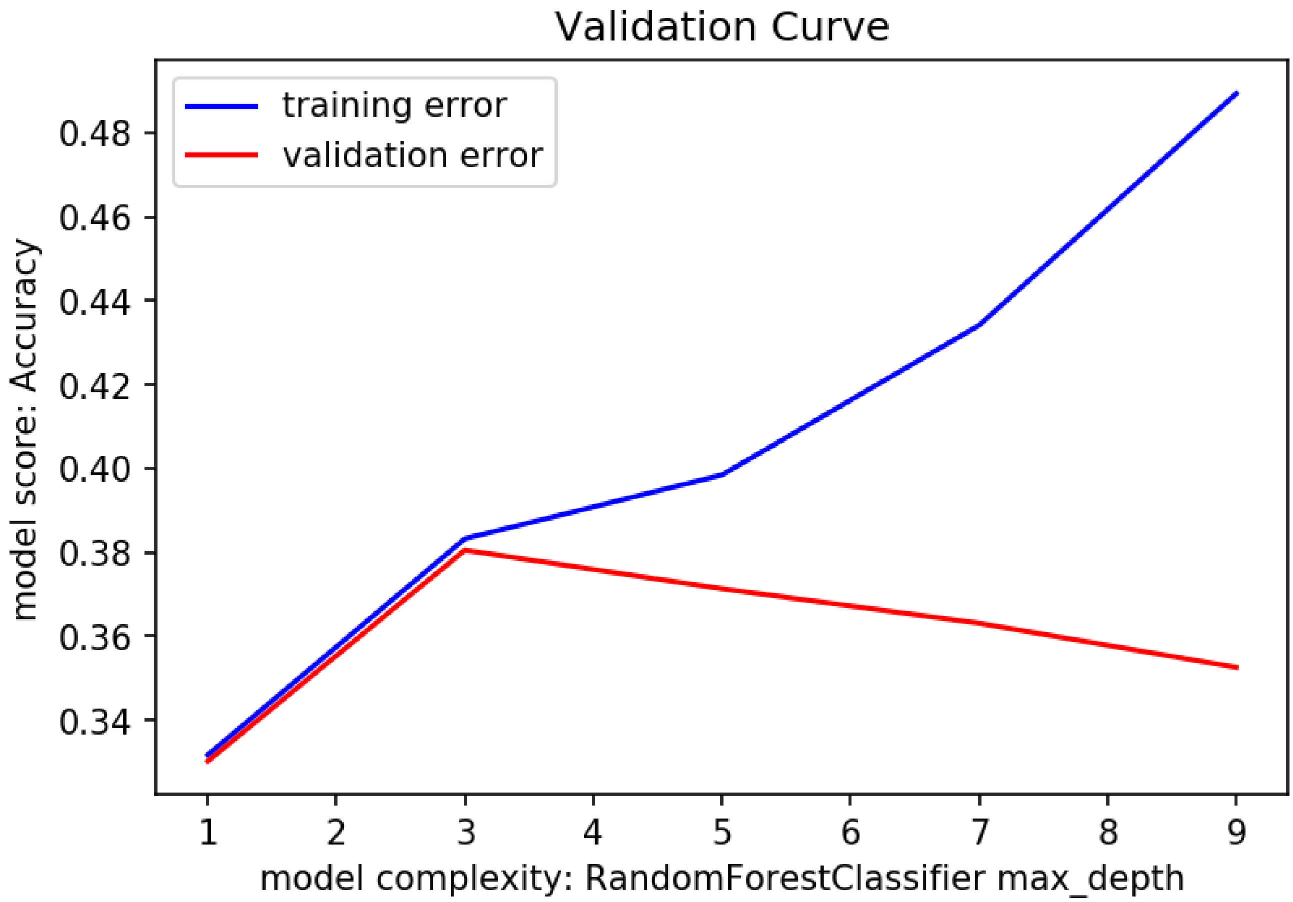
```
Validation Accuracy 0.398422090729783
```

```
# 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');
```




```
# 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();
```




```
# 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, scoring='accuracy', verbose

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

```
Fitting 3 folds for each of 3 candidates, totalling 9 fits
/usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_search.py:266: UserWarning: The total space of
  % (grid_size, self.n_iter, grid_size), UserWarning)
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done   1 tasks      | elapsed:    3.6s
[Parallel(n_jobs=-1)]: Done   4 tasks      | elapsed:    5.7s
[Parallel(n_jobs=-1)]: Done   7 out of   9 | elapsed:    9.8s remaining:    2.8s
[Parallel(n_jobs=-1)]: Done   9 out of   9 | elapsed:   11.2s remaining:    0.0s
[Parallel(n_jobs=-1)]: Done   9 out of   9 | elapsed:   11.2s finished
```

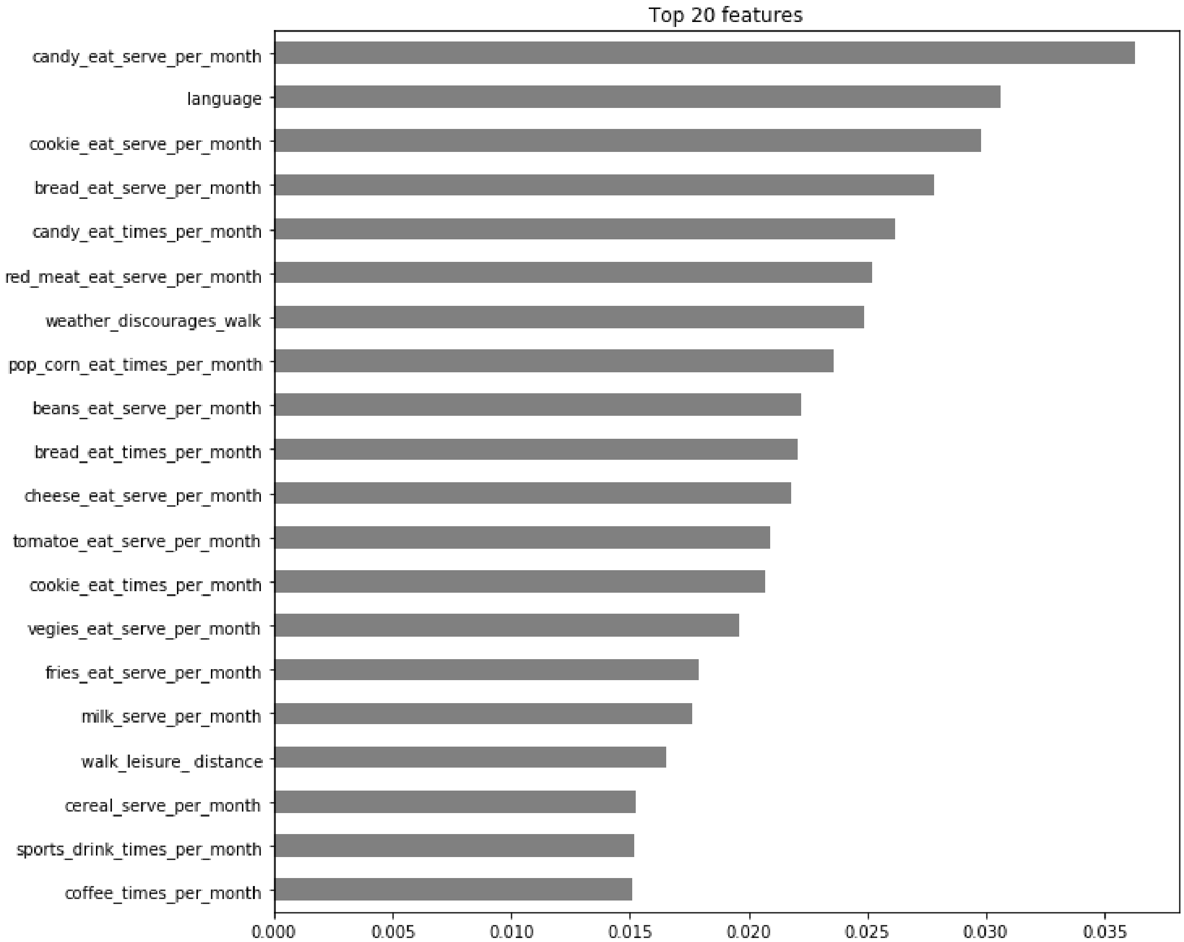
```
from sklearn.model_selection import cross_val_score
k = 3
scores = cross_val_score(pipeline, X_val, y_val, cv=k,
scoring='accuracy')
print(f'Validation Accuracy for {k} folds:', scores);
```

```
Validation Accuracy for 3 folds: [0.39803922 0.39250493 0.39285714]
```

```
print('Best hyperparameters', search.best_params_)
print('Cross-validation Accuracy', search.best_score_)
```

```
Best hyperparameters {'simpleimputer__strategy': 'mean'}
Cross-validation Accuracy 0.3953297155073179
```

```
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');
```




```
# Demonstrate the relatively high cardinatlity of candy_eat_times_per_month
XTrain['cookie_eat_serve_per_month'].value_counts()
```

```
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
```

```
# 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: 0.39316239316239315
Validation Accuracy with cookie_eat_serve_per_month: 0.398422090729783
Drop-Column Importance for cookie_eat_serve_per_month: 0.005259697567389865
```

```
# 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: 0.3793556870479947
Validation Accuracy with language: 0.398422090729783
Permutation Importance: 0.019066403681788302
```

```
# 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)
```

```
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)
```

```
# 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
)
```

```
Collecting eli5
  Downloading https://files.pythonhosted.org/packages/97/2f/c85c7d8f8548e460829971785347e14e45fa5c6617da374711c
     |████████████████████████████████| 112kB 10.2MB/s
Requirement already satisfied: jinja2 in /usr/local/lib/python3.6/dist-packages (from eli5) (2.10.3)
Requirement already satisfied: graphviz in /usr/local/lib/python3.6/dist-packages (from eli5) (0.10.1)
Requirement already satisfied: numpy>=1.9.0 in /usr/local/lib/python3.6/dist-packages (from eli5) (1.16.5)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from eli5) (1.12.0)
Requirement already satisfied: scikit-learn>=0.18 in /usr/local/lib/python3.6/dist-packages (from eli5) (0.21.3
Requirement already satisfied: attrs>16.0.0 in /usr/local/lib/python3.6/dist-packages (from eli5) (19.3.0)
Requirement already satisfied: tabulate>=0.7.7 in /usr/local/lib/python3.6/dist-packages (from eli5) (0.8.5)
Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packages (from eli5) (1.3.1)
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.6/dist-packages (from jinja2->eli5) (
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.6/dist-packages (from scikit-learn>=0.18-
Installing collected packages: eli5
Successfully installed eli5-0.10.1
Using TensorFlow backend.
```

Weight	Feature
0.0168 ± 0.0033	language
0.0053 ± 0.0039	see_walking_from_home
0.0053 ± 0.0066	bread_eat_serve_per_month
0.0049 ± 0.0007	sports_drink_times_per_month
0.0039 ± 0.0066	walk_leisure_ time
0.0036 ± 0.0085	walkable_entertainment
0.0036 ± 0.0007	beans_eat_serve_per_month
0.0033 ± 0.0013	soda_times_per_month
0.0033 ± 0.0013	cookie_eat_times_per_month
0.0030 ± 0.0033	processed_meat_eat_times_per_month
0.0030 ± 0.0007	walkable_bus_stop
0.0030 ± 0.0020	red_meat_eat_times_per_month
0.0026 ± 0.0000	milk_serve_per_month
0.0026 ± 0.0039	tobacco_even_once
0.0026 ± 0.0066	cigar_even_once
0.0026 ± 0.0026	fries_eat_serve_per_month
0.0026 ± 0.0026	salad_eat_times_per_month
0.0023 ± 0.0046	grains_eat_times_per_month
0.0023 ± 0.0033	walk_leisure_ distance
0.0023 ± 0.0007	walk_leisure_number_wk
0.0023 ± 0.0072	coffee_times_per_month
0.0020 ± 0.0013	walk_leisure_past_wk
0.0020 ± 0.0013	juice_times_per_month
0.0016 ± 0.0020	walk_past_wk
0.0016 ± 0.0020	vitD_reason
0.0016 ± 0.0007	single_walk_distance
0.0016 ± 0.0007	grains_eat_serve_per_month
0.0016 ± 0.0007	traffic_discourages_walking
0.0013 ± 0.0066	red_meat_eat_serve_per_month
0.0013 ± 0.0013	animals_discourage_walking
0.0013 ± 0.0013	cereal_serve_per_month
0.0013 ± 0.0013	cheese_eat_serve_per_month
0.0010 ± 0.0007	multivitamin_past_month
0.0010 ± 0.0007	cereal_times_per_month
0.0010 ± 0.0007	fries_eat_times_per_month
0.0010 ± 0.0020	ice_cream_eat_times_per_month
0.0010 ± 0.0007	calcium_days_in_month
0.0010 ± 0.0033	walk_number_wk
0.0010 ± 0.0020	vitD_past_month
0.0007 ± 0.0000	walkable_relaxation
0.0007 ± 0.0000	walk_leisure_distance_week
0.0007 ± 0.0013	vitD_days_in_month
0.0007 ± 0.0026	donut_eat_times_per_month
0.0003 ± 0.0007	vitamin_past_month
0.0003 ± 0.0007	genetic_counseling_for_cancer
0.0003 ± 0.0033	multivitamin_days_in_month
0 ± 0.0000	had_genetic_counseling
0 ± 0.0000	genetic_counseling_with_MD
-0.0000 ± 0.0066	pipe_even_once
-0.0000 ± 0.0026	beans_eat_times_per_month
-0.0000 ± 0.0013	more_than_one_cereal_type
-0.0000 ± 0.0039	cookie_eat_serve_per_month
-0.0000 ± 0.0013	tomatoe_eat_times_per_month
-0.0003 ± 0.0007	crime_discourages_walking

-0.0003 ± 0.0020	fruit_eat_times_per_month
-0.0007 ± 0.0013	calcium_past_month
-0.0007 ± 0.0013	potatoe_eat_times_per_month
-0.0007 ± 0.0026	weather_discourages_walk
-0.0007 ± 0.0026	candy_eat_times_per_month
-0.0007 ± 0.0026	cheese_eat_times_per_month
-0.0007 ± 0.0013	pop_corn_eat_times_per_month
-0.0010 ± 0.0007	single_walk_distance_week
-0.0010 ± 0.0007	walkway_existence
-0.0010 ± 0.0046	vegies_eat_serve_per_month
-0.0010 ± 0.0007	tomatoe_eat_serve_per_month
-0.0010 ± 0.0007	salsa_eat_times_per_month
-0.0013 ± 0.0026	single_walk_time
-0.0016 ± 0.0033	cigarette_even_once
-0.0016 ± 0.0020	fruit_drink_times_per_month
-0.0023 ± 0.0033	streets_have_walkways
-0.0026 ± 0.0013	2nd_kind_cereal_eaten
-0.0026 ± 0.0053	milk_type
-0.0026 ± 0.0026	pizza_eat_times_per_month
-0.0026 ± 0.0039	vegies_eat_times_per_month
-0.0026 ± 0.0026	bread_eat_times_per_month
-0.0030 ± 0.0085	soda_serve_per_month
-0.0030 ± 0.0020	candy_eat_serve_per_month
-0.0030 ± 0.0046	1st_kind_cereal_eaten
-0.0033 ± 0.0026	milk_times_per_month
-0.0036 ± 0.0020	smokeless_even_once
-0.0036 ± 0.0099	walkable_retail

```
# Thus, language is way more important according to feature permutation than according to feature importance in the Random Fo
# Use importances for feature selection
print('Shape before removing features:', X_train.shape)
```

```
Shape before removing features: (6081, 81)
```

```
# 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, 46)
```

```
# Random Forest with reduced features to 46
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 0.4049967126890204
```

```
# Validation Accuracy History
# 0.2864660417694458- baseline guessing the majority class
# 0.4010853478046374- initial fit with optimal hyperparameters
```

```
# 0.398422090729783 - use pipeline with random forest -- with engineered features (0.398422090729783)
# 0.3953297155073179- from cross validation -- with engineered features (0.3953297155073179)
# 0.398422090729783 - doing permutation importance -- with engineered features (0.398422090729783)
# 0.4049967126890204- after removing features of zero importance -- with engineered features (0.4049967126890204)


# Recursive Feature Elimination
from sklearn.feature_selection import RFECV
from sklearn.model_selection import StratifiedKFold

rfc = 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)

rfecv = RFECV(estimator=rfc, step=1, cv=StratifiedKFold(9), scoring='accuracy', verbose = 10)
rfecv.fit(X_train, y_train)
```

```
Fitting estimator with 46 features.
Fitting estimator with 45 features.
Fitting estimator with 44 features.
Fitting estimator with 43 features.
Fitting estimator with 42 features.
Fitting estimator with 41 features.
Fitting estimator with 40 features.
Fitting estimator with 39 features.
Fitting estimator with 38 features.
Fitting estimator with 37 features.
Fitting estimator with 36 features.
Fitting estimator with 35 features.
Fitting estimator with 34 features.
Fitting estimator with 33 features.
Fitting estimator with 32 features.
Fitting estimator with 31 features.
Fitting estimator with 30 features.
Fitting estimator with 29 features.
Fitting estimator with 28 features.
Fitting estimator with 27 features.
Fitting estimator with 26 features.
Fitting estimator with 25 features.
Fitting estimator with 24 features.
Fitting estimator with 23 features.
Fitting estimator with 22 features.
Fitting estimator with 21 features.
Fitting estimator with 20 features.
Fitting estimator with 19 features.
Fitting estimator with 18 features.
Fitting estimator with 17 features.
Fitting estimator with 16 features.
Fitting estimator with 15 features.
Fitting estimator with 14 features.
Fitting estimator with 13 features.
Fitting estimator with 12 features.
Fitting estimator with 11 features.
Fitting estimator with 10 features.
Fitting estimator with 9 features.
Fitting estimator with 8 features.
Fitting estimator with 7 features.
Fitting estimator with 6 features.
Fitting estimator with 5 features.
Fitting estimator with 4 features.
Fitting estimator with 3 features.
Fitting estimator with 2 features.
Fitting estimator with 46 features.
Fitting estimator with 45 features.
Fitting estimator with 44 features.
Fitting estimator with 43 features.
Fitting estimator with 42 features.
Fitting estimator with 41 features.
Fitting estimator with 40 features.
Fitting estimator with 39 features.
Fitting estimator with 38 features.
Fitting estimator with 37 features.
Fitting estimator with 36 features.
Fitting estimator with 35 features.
Fitting estimator with 34 features.
Fitting estimator with 33 features.
Fitting estimator with 32 features.
Fitting estimator with 31 features.
Fitting estimator with 30 features.
Fitting estimator with 29 features.
Fitting estimator with 28 features.
Fitting estimator with 27 features.
Fitting estimator with 26 features.
Fitting estimator with 25 features.
Fitting estimator with 24 features.
Fitting estimator with 23 features.
Fitting estimator with 22 features.
Fitting estimator with 21 features.
```

```
Fitting estimator with 21 features.
Fitting estimator with 20 features.
Fitting estimator with 19 features.
Fitting estimator with 18 features.
Fitting estimator with 17 features.
Fitting estimator with 16 features.
Fitting estimator with 15 features.
Fitting estimator with 14 features.
Fitting estimator with 13 features.
Fitting estimator with 12 features.
Fitting estimator with 11 features.
Fitting estimator with 10 features.
Fitting estimator with 9 features.
Fitting estimator with 8 features.
Fitting estimator with 7 features.
Fitting estimator with 6 features.
Fitting estimator with 5 features.
Fitting estimator with 4 features.
Fitting estimator with 3 features.
Fitting estimator with 2 features.
Fitting estimator with 46 features.
Fitting estimator with 45 features.
Fitting estimator with 44 features.
Fitting estimator with 43 features.
Fitting estimator with 42 features.
Fitting estimator with 41 features.
Fitting estimator with 40 features.
Fitting estimator with 39 features.
Fitting estimator with 38 features.
Fitting estimator with 37 features.
Fitting estimator with 36 features.
Fitting estimator with 35 features.
Fitting estimator with 34 features.
Fitting estimator with 33 features.
Fitting estimator with 32 features.
Fitting estimator with 31 features.
Fitting estimator with 30 features.
Fitting estimator with 29 features.
Fitting estimator with 28 features.
Fitting estimator with 27 features.
Fitting estimator with 26 features.
Fitting estimator with 25 features.
Fitting estimator with 24 features.
Fitting estimator with 23 features.
Fitting estimator with 22 features.
Fitting estimator with 21 features.
Fitting estimator with 20 features.
Fitting estimator with 19 features.
Fitting estimator with 18 features.
Fitting estimator with 17 features.
Fitting estimator with 16 features.
Fitting estimator with 15 features.
Fitting estimator with 14 features.
Fitting estimator with 13 features.
Fitting estimator with 12 features.
Fitting estimator with 11 features.
Fitting estimator with 10 features.
Fitting estimator with 9 features.
Fitting estimator with 8 features.
Fitting estimator with 7 features.
Fitting estimator with 6 features.
Fitting estimator with 5 features.
Fitting estimator with 4 features.
Fitting estimator with 3 features.
Fitting estimator with 2 features.
Fitting estimator with 46 features.
Fitting estimator with 45 features.
Fitting estimator with 44 features.
Fitting estimator with 43 features.
Fitting estimator with 42 features.
Fitting estimator with 41 features.
Fitting estimator with 40 features.
```

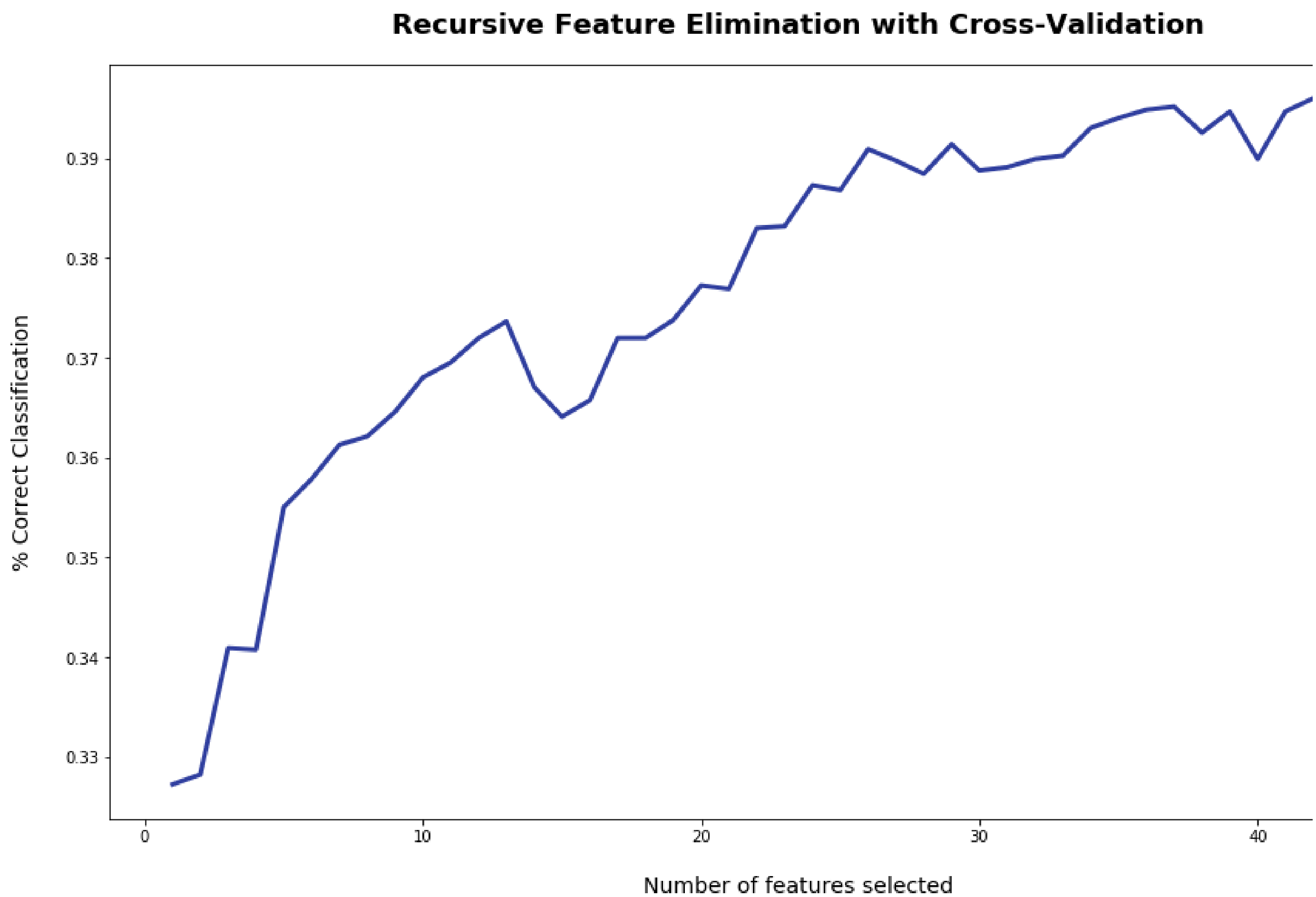
```
Fitting estimator with 40 features.
Fitting estimator with 39 features.
Fitting estimator with 38 features.
Fitting estimator with 37 features.
Fitting estimator with 36 features.
Fitting estimator with 35 features.
Fitting estimator with 34 features.
Fitting estimator with 33 features.
Fitting estimator with 32 features.
Fitting estimator with 31 features.
Fitting estimator with 30 features.
Fitting estimator with 29 features.
Fitting estimator with 28 features.
Fitting estimator with 27 features.
Fitting estimator with 26 features.
Fitting estimator with 25 features.
Fitting estimator with 24 features.
Fitting estimator with 23 features.
Fitting estimator with 22 features.
Fitting estimator with 21 features.
Fitting estimator with 20 features.
Fitting estimator with 19 features.
Fitting estimator with 18 features.
Fitting estimator with 17 features.
Fitting estimator with 16 features.
Fitting estimator with 15 features.
Fitting estimator with 14 features.
Fitting estimator with 13 features.
Fitting estimator with 12 features.
Fitting estimator with 11 features.
Fitting estimator with 10 features.
Fitting estimator with 9 features.
Fitting estimator with 8 features.
Fitting estimator with 7 features.
Fitting estimator with 6 features.
Fitting estimator with 5 features.
Fitting estimator with 4 features.
Fitting estimator with 3 features.
Fitting estimator with 2 features.
Fitting estimator with 46 features.
Fitting estimator with 45 features.
Fitting estimator with 44 features.
Fitting estimator with 43 features.
Fitting estimator with 42 features.
Fitting estimator with 41 features.
Fitting estimator with 40 features.
Fitting estimator with 39 features.
Fitting estimator with 38 features.
Fitting estimator with 37 features.
Fitting estimator with 36 features.
Fitting estimator with 35 features.
Fitting estimator with 34 features.
Fitting estimator with 33 features.
Fitting estimator with 32 features.
Fitting estimator with 31 features.
Fitting estimator with 30 features.
Fitting estimator with 29 features.
Fitting estimator with 28 features.
Fitting estimator with 27 features.
Fitting estimator with 26 features.
Fitting estimator with 25 features.
Fitting estimator with 24 features.
Fitting estimator with 23 features.
Fitting estimator with 22 features.
Fitting estimator with 21 features.
Fitting estimator with 20 features.
Fitting estimator with 19 features.
Fitting estimator with 18 features.
Fitting estimator with 17 features.
Fitting estimator with 16 features.
Fitting estimator with 15 features.
Fitting estimator with 14 features.
```

```
Fitting estimator with 13 features.
Fitting estimator with 12 features.
Fitting estimator with 11 features.
Fitting estimator with 10 features.
Fitting estimator with 9 features.
Fitting estimator with 8 features.
Fitting estimator with 7 features.
Fitting estimator with 6 features.
Fitting estimator with 5 features.
Fitting estimator with 4 features.
Fitting estimator with 3 features.
Fitting estimator with 2 features.
Fitting estimator with 46 features.
Fitting estimator with 45 features.
Fitting estimator with 44 features.
Fitting estimator with 43 features.
Fitting estimator with 42 features.
Fitting estimator with 41 features.
Fitting estimator with 40 features.
Fitting estimator with 39 features.
Fitting estimator with 38 features.
Fitting estimator with 37 features.
Fitting estimator with 36 features.
Fitting estimator with 35 features.
Fitting estimator with 34 features.
Fitting estimator with 33 features.
Fitting estimator with 32 features.
Fitting estimator with 31 features.
Fitting estimator with 30 features.
Fitting estimator with 29 features.
Fitting estimator with 28 features.
Fitting estimator with 27 features.
Fitting estimator with 26 features.
Fitting estimator with 25 features.
Fitting estimator with 24 features.
Fitting estimator with 23 features.
Fitting estimator with 22 features.
Fitting estimator with 21 features.
Fitting estimator with 20 features.
Fitting estimator with 19 features.
Fitting estimator with 18 features.
Fitting estimator with 17 features.
Fitting estimator with 16 features.
Fitting estimator with 15 features.
Fitting estimator with 14 features.
Fitting estimator with 13 features.
Fitting estimator with 12 features.
Fitting estimator with 11 features.
Fitting estimator with 10 features.
Fitting estimator with 9 features.
Fitting estimator with 8 features.
Fitting estimator with 7 features.
Fitting estimator with 6 features.
Fitting estimator with 5 features.
Fitting estimator with 4 features.
Fitting estimator with 3 features.
Fitting estimator with 2 features.
Fitting estimator with 46 features.
Fitting estimator with 45 features.
Fitting estimator with 44 features.
Fitting estimator with 43 features.
Fitting estimator with 42 features.
Fitting estimator with 41 features.
Fitting estimator with 40 features.
Fitting estimator with 39 features.
Fitting estimator with 38 features.
Fitting estimator with 37 features.
Fitting estimator with 36 features.
Fitting estimator with 35 features.
Fitting estimator with 34 features.
Fitting estimator with 33 features.
```

```
Fitting estimator with 32 features.
Fitting estimator with 31 features.
Fitting estimator with 30 features.
Fitting estimator with 29 features.
Fitting estimator with 28 features.
Fitting estimator with 27 features.
Fitting estimator with 26 features.
Fitting estimator with 25 features.
Fitting estimator with 24 features.
Fitting estimator with 23 features.
Fitting estimator with 22 features.
Fitting estimator with 21 features.
Fitting estimator with 20 features.
Fitting estimator with 19 features.
Fitting estimator with 18 features.
Fitting estimator with 17 features.
Fitting estimator with 16 features.
Fitting estimator with 15 features.
Fitting estimator with 14 features.
Fitting estimator with 13 features.
Fitting estimator with 12 features.
Fitting estimator with 11 features.
Fitting estimator with 10 features.
Fitting estimator with 9 features.
Fitting estimator with 8 features.
Fitting estimator with 7 features.
Fitting estimator with 6 features.
Fitting estimator with 5 features.
Fitting estimator with 4 features.
Fitting estimator with 3 features.
Fitting estimator with 2 features.
Fitting estimator with 46 features.
Fitting estimator with 45 features.
Fitting estimator with 44 features.
Fitting estimator with 43 features.
Fitting estimator with 42 features.
Fitting estimator with 41 features.
Fitting estimator with 40 features.
Fitting estimator with 39 features.
Fitting estimator with 38 features.
Fitting estimator with 37 features.
Fitting estimator with 36 features.
Fitting estimator with 35 features.
Fitting estimator with 34 features.
Fitting estimator with 33 features.
Fitting estimator with 32 features.
Fitting estimator with 31 features.
Fitting estimator with 30 features.
Fitting estimator with 29 features.
Fitting estimator with 28 features.
Fitting estimator with 27 features.
Fitting estimator with 26 features.
Fitting estimator with 25 features.
Fitting estimator with 24 features.
Fitting estimator with 23 features.
Fitting estimator with 22 features.
Fitting estimator with 21 features.
Fitting estimator with 20 features.
Fitting estimator with 19 features.
Fitting estimator with 18 features.
Fitting estimator with 17 features.
Fitting estimator with 16 features.
Fitting estimator with 15 features.
Fitting estimator with 14 features.
Fitting estimator with 13 features.
Fitting estimator with 12 features.
Fitting estimator with 11 features.
Fitting estimator with 10 features.
Fitting estimator with 9 features.
Fitting estimator with 8 features.
Fitting estimator with 7 features.
```

```
Fitting estimator with 6 features.
Fitting estimator with 5 features.
Fitting estimator with 4 features.
Fitting estimator with 3 features.
Fitting estimator with 2 features.
Fitting estimator with 46 features.
Fitting estimator with 45 features.
Fitting estimator with 44 features.
Fitting estimator with 43 features.
Fitting estimator with 42 features.
Fitting estimator with 41 features.
Fitting estimator with 40 features.
Fitting estimator with 39 features.
Fitting estimator with 38 features.
Fitting estimator with 37 features.
Fitting estimator with 36 features.
Fitting estimator with 35 features.
Fitting estimator with 34 features.
Fitting estimator with 33 features.
Fitting estimator with 32 features.
Fitting estimator with 31 features.
Fitting estimator with 30 features.
Fitting estimator with 29 features.
Fitting estimator with 28 features.
Fitting estimator with 27 features.
Fitting estimator with 26 features.
Fitting estimator with 25 features.
Fitting estimator with 24 features.
Fitting estimator with 23 features.
Fitting estimator with 22 features.
Fitting estimator with 21 features.
Fitting estimator with 20 features.
Fitting estimator with 19 features.
Fitting estimator with 18 features.
Fitting estimator with 17 features.
Fitting estimator with 16 features.
Fitting estimator with 15 features.
Fitting estimator with 14 features.
Fitting estimator with 13 features.
Fitting estimator with 12 features.
Fitting estimator with 11 features.
Fitting estimator with 10 features.
Fitting estimator with 9 features.
Fitting estimator with 8 features.
Fitting estimator with 7 features.
Fitting estimator with 6 features.
Fitting estimator with 5 features.
Fitting estimator with 4 features.
Fitting estimator with 3 features.
Fitting estimator with 2 features.
Fitting estimator with 46 features.
Fitting estimator with 45 features.
Fitting estimator with 44 features.
Fitting estimator with 43 features.
RFECV(cv=StratifiedKFold(n_splits=9, random_state=None, shuffle=False),
      estimator=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),
      min_features_to_select=1, n_jobs=None, scoring='accuracy', step=1,
      verbose=10)
```

```
#Plot the results of RFE
plt.figure(figsize=(16, 9))
plt.title('Recursive Feature Elimination with Cross-Validation', fontsize=18, fontweight='bold', pad=20)
plt.xlabel('Number of features selected', fontsize=14, labelpad=20)
plt.ylabel('% Correct Classification', fontsize=14, labelpad=20)
plt.plot(range(1, len(rfecv.grid_scores_) + 1), rfecv.grid_scores_, color='#303F9F', linewidth=3)

plt.show()
```




```
# Print the optimal number of features and accuracy after RFE
print('Optimal number of features: {}'.format(rfecv.n_features_))

y_pred = rfecv.predict(X_val)
print ('Accuracy = ', accuracy_score(y_val, y_pred))
```

```
Optimal number of features: 42
Accuracy =  0.41946088099934253
```

```
# Note that this is a 46.4% improvement over baseline
```

```
# Drop unimportant features
print(np.where(rfecv.support_ == False)[0])

X_train.drop(X_train.columns[np.where(rfecv.support_ == False)[0]], axis=1, inplace=True)
X_val.drop(X_val.columns[np.where(rfecv.support_ == False)[0]], axis=1, inplace=True)
```

```
X_val.shape
```

```
[26 29 41 43]
/usr/local/lib/python3.6/dist-packages/pandas/core/frame.py:3940: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-v
  errors=errors)
(1521, 42)
```

```
X_train.shape
```

```
(6081, 42)
```

```
#Fit to RFECV data set to confirm the best accuracy score

pipeline0 = 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
pipeline0.fit(X_train, y_train)
print('Validation Accuracy', pipeline0.score(X_val, y_val))
```

```
Validation Accuracy 0.41946088099934253
```

```
# Seeing if feature scaling will improve accuracy
from sklearn.preprocessing import MinMaxScaler

# Get the numbers for the items to be removed from features above
reduced_features = features.delete([26, 29, 41, 43])

min_max=MinMaxScaler()
# Scaling down both train and test data set
X_train_minmax=min_max.fit_transform(X_train[reduced_features])
X_val_minmax=min_max.fit_transform(X_val[reduced_features])
```

```
#Fit to the scaled data set

pipeline1 = 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
pipeline1.fit(X_train_minmax, y_train)
print('Validation Accuracy', pipeline1.score(X_val_minmax, y_val))
```

```
Validation Accuracy 0.41354372123602895
```

```
# Since scaling does not improve the accuracy score, it is not implemented.
```

```
# Seeing if feature standardization will improve accuracy
from sklearn.preprocessing import scale

X_train_scale=scale(X_train[reduced_features])
X_val_scale=scale(X_val[reduced_features])
```

```
#Fit to the standardized data set

pipeline2 = 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
pipeline2.fit(X_train_scale, y_train)
print('Validation Accuracy', pipeline2.score(X_val_scale, y_val))
```

```
Validation Accuracy 0.410913872452334
```

```
# Since standardizing does not improve the accuracy score, it is not implemented.
```

```
# 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
```

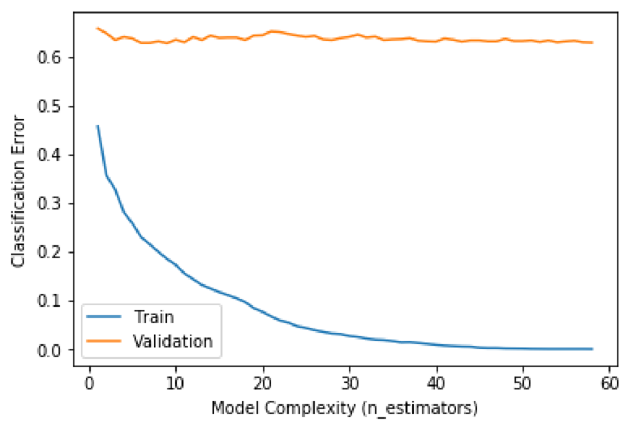
```
((6081, 42), (1521, 42), (6081, 42), (1521, 42))
```

```
#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)
```

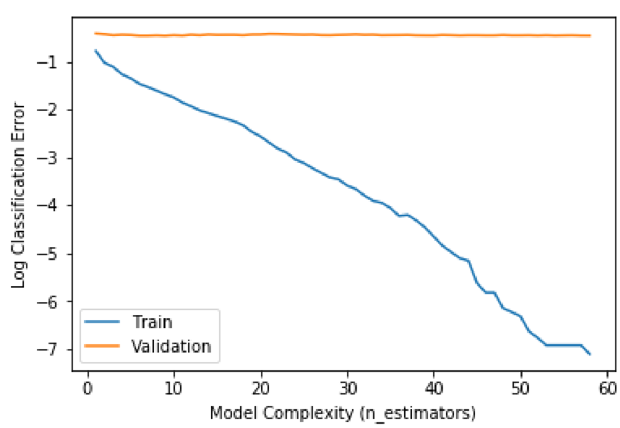
```
[0]	validation_0-merror:0.456997	validation_1-merror:0.65812
Multiple eval metrics have been passed: 'validation_1-merror' will be used for early stopping.

Will train until validation_1-merror hasn't improved in 50 rounds.
[1]	validation_0-merror:0.356685	validation_1-merror:0.648258
[2]	validation_0-merror:0.328071	validation_1-merror:0.634451
[3]	validation_0-merror:0.281533	validation_1-merror:0.641026
[4]	validation_0-merror:0.258675	validation_1-merror:0.637738
[5]	validation_0-merror:0.230719	validation_1-merror:0.628534
[6]	validation_0-merror:0.21559	validation_1-merror:0.628534
[7]	validation_0-merror:0.199967	validation_1-merror:0.631821
[8]	validation_0-merror:0.185496	validation_1-merror:0.627876
[9]	validation_0-merror:0.173327	validation_1-merror:0.635108
[10]	validation_0-merror:0.155731	validation_1-merror:0.629849
[11]	validation_0-merror:0.14422	validation_1-merror:0.641026
[12]	validation_0-merror:0.13238	validation_1-merror:0.634451
[13]	validation_0-merror:0.124979	validation_1-merror:0.643655
[14]	validation_0-merror:0.117744	validation_1-merror:0.639053
[15]	validation_0-merror:0.111495	validation_1-merror:0.639711
[16]	validation_0-merror:0.105081	validation_1-merror:0.639711
[17]	validation_0-merror:0.097188	validation_1-merror:0.634451
[18]	validation_0-merror:0.08469	validation_1-merror:0.643655
[19]	validation_0-merror:0.077125	validation_1-merror:0.644313
[20]	validation_0-merror:0.067752	validation_1-merror:0.652202
[21]	validation_0-merror:0.05953	validation_1-merror:0.650888
[22]	validation_0-merror:0.054925	validation_1-merror:0.646943
[23]	validation_0-merror:0.047525	validation_1-merror:0.643655
[24]	validation_0-merror:0.044072	validation_1-merror:0.641026
[25]	validation_0-merror:0.039632	validation_1-merror:0.642998
[26]	validation_0-merror:0.036014	validation_1-merror:0.635766
[27]	validation_0-merror:0.032725	validation_1-merror:0.634451
[28]	validation_0-merror:0.031409	validation_1-merror:0.638396
[29]	validation_0-merror:0.027627	validation_1-merror:0.641026
[30]	validation_0-merror:0.025654	validation_1-merror:0.645628
[31]	validation_0-merror:0.022365	validation_1-merror:0.639711
[32]	validation_0-merror:0.020062	validation_1-merror:0.641683
[33]	validation_0-merror:0.01924	validation_1-merror:0.634451
[34]	validation_0-merror:0.017267	validation_1-merror:0.635766
[35]	validation_0-merror:0.014636	validation_1-merror:0.636423
[36]	validation_0-merror:0.014965	validation_1-merror:0.638396
[37]	validation_0-merror:0.01332	validation_1-merror:0.633136
[38]	validation_0-merror:0.011511	validation_1-merror:0.631821
[39]	validation_0-merror:0.009538	validation_1-merror:0.631164
[40]	validation_0-merror:0.007893	validation_1-merror:0.637738
[41]	validation_0-merror:0.006907	validation_1-merror:0.635108
[42]	validation_0-merror:0.006085	validation_1-merror:0.631164
[43]	validation_0-merror:0.005756	validation_1-merror:0.633794
[44]	validation_0-merror:0.003618	validation_1-merror:0.633794
[45]	validation_0-merror:0.00296	validation_1-merror:0.631821
[46]	validation_0-merror:0.00296	validation_1-merror:0.631821
[47]	validation_0-merror:0.002138	validation_1-merror:0.637081
[48]	validation_0-merror:0.001973	validation_1-merror:0.632479
[49]	validation_0-merror:0.001809	validation_1-merror:0.632479
[50]	validation_0-merror:0.001316	validation_1-merror:0.633794
[51]	validation_0-merror:0.001151	validation_1-merror:0.630506
[52]	validation_0-merror:0.000987	validation_1-merror:0.633794
[53]	validation_0-merror:0.000987	validation_1-merror:0.629849
[54]	validation_0-merror:0.000987	validation_1-merror:0.631821
[55]	validation_0-merror:0.000987	validation_1-merror:0.633136
[56]	validation_0-merror:0.000987	validation_1-merror:0.629849
[57]	validation_0-merror:0.000822	validation_1-merror:0.629191
[58]	validation_0-merror:0.000822	validation_1-merror:0.629191
Stopping. Best iteration:
[8]	validation_0-merror:0.185496	validation_1-merror:0.627876

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,
```

```
                    missing=None, n_estimators=209, 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)
```

```python
# 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();
```




```python
# 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();
```




```python
# 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))
```

```
/usr/local/lib/python3.6/dist-packages/xgboost/core.py:587: FutureWarning: Series.base is deprecated and will b
  if getattr(data, 'base', None) is not None and \
Gradient Boosting R^2 0.2734730075382791
```

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

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

```
# 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))
```

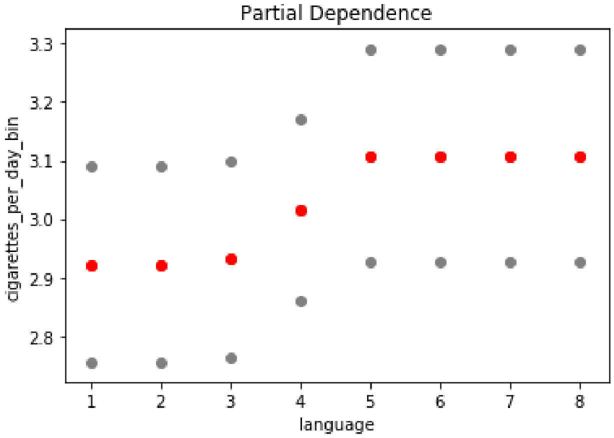
```
# Vary the language feature while holding all other features constant for the first row
example1 = X_val.iloc[[0]]
vary_language(gb, example1)
```

```
Vary language, hold other features constant

Predicted cigarettes_per_day_bin: 3.090%
       language  cereal_serve_per_month  cereal_times_per_month  milk_serve_per_month  soda_times_per_month  ju
31502         1                       2                       3                     3                     0

Predicted cigarettes_per_day_bin: 3.090%
       language  cereal_serve_per_month  cereal_times_per_month  milk_serve_per_month  soda_times_per_month  ju
31502         2                       2                       3                     3                     0

Predicted cigarettes_per_day_bin: 3.099%
       language  cereal_serve_per_month  cereal_times_per_month  milk_serve_per_month  soda_times_per_month  ju
31502         3                       2                       3                     3                     0

Predicted cigarettes_per_day_bin: 3.171%
       language  cereal_serve_per_month  cereal_times_per_month  milk_serve_per_month  soda_times_per_month  ju
31502         4                       2                       3                     3                     0

Predicted cigarettes_per_day_bin: 3.289%
       language  cereal_serve_per_month  cereal_times_per_month  milk_serve_per_month  soda_times_per_month  ju
31502         5                       2                       3                     3                     0

Predicted cigarettes_per_day_bin: 3.289%
       language  cereal_serve_per_month  cereal_times_per_month  milk_serve_per_month  soda_times_per_month  ju
31502         6                       2                       3                     3                     0

Predicted cigarettes_per_day_bin: 3.289%
       language  cereal_serve_per_month  cereal_times_per_month  milk_serve_per_month  soda_times_per_month  ju
31502         7                       2                       3                     3                     0

Predicted cigarettes_per_day_bin: 3.289%
       language  cereal_serve_per_month  cereal_times_per_month  milk_serve_per_month  soda_times_per_month  ju
31502         8                       2                       3                     3                     0

Difference between predictions
[0.         0.00901175 0.07167602 0.11833286 0.         0.
 0.        ]
```

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

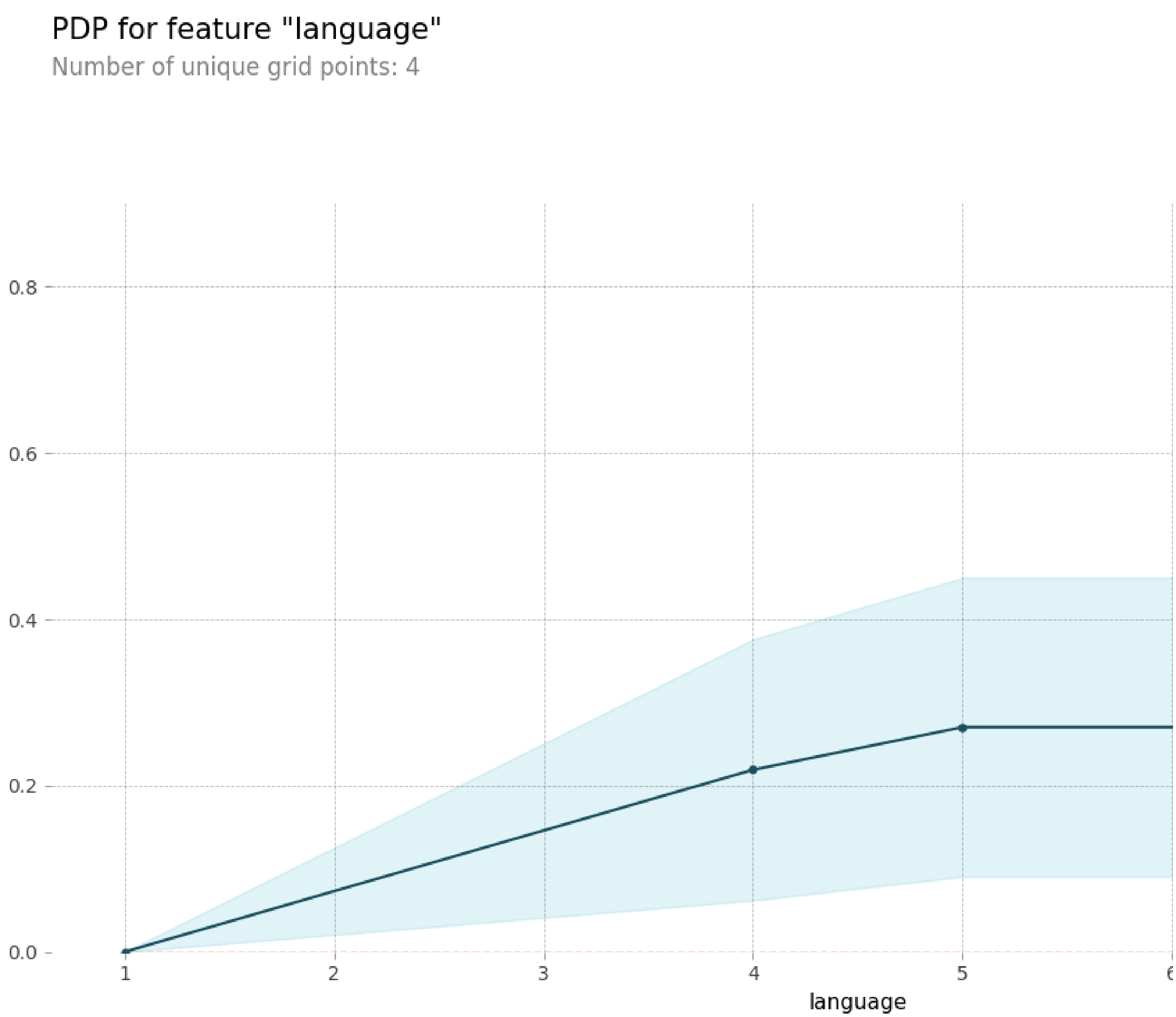
```
Vary language, hold other features constant

Predicted cigarettes_per_day_bin: 2.755%
       language  cereal_serve_per_month  cereal_times_per_month  milk_serve_per_month  soda_times_per_month  ju
27082         1                       1                       2                     2                     0

Predicted cigarettes_per_day_bin: 2.755%
       language  cereal_serve_per_month  cereal_times_per_month  milk_serve_per_month  soda_times_per_month  ju
27082         2                       1                       2                     2                     0

Predicted cigarettes_per_day_bin: 2.764%
       language  cereal_serve_per_month  cereal_times_per_month  milk_serve_per_month  soda_times_per_month  ju
27082         3                       1                       2                     2                     0

Predicted cigarettes_per_day_bin: 2.862%
       language  cereal_serve_per_month  cereal_times_per_month  milk_serve_per_month  soda_times_per_month  ju
27082         4                       1                       2                     2                     0

Predicted cigarettes_per_day_bin: 2.926%
       language  cereal_serve_per_month  cereal_times_per_month  milk_serve_per_month  soda_times_per_month  ju
27082         5                       1                       2                     2                     0

Predicted cigarettes_per_day_bin: 2.926%
       language  cereal_serve_per_month  cereal_times_per_month  milk_serve_per_month  soda_times_per_month  ju
27082         6                       1                       2                     2                     0

Predicted cigarettes_per_day_bin: 2.926%
       language  cereal_serve_per_month  cereal_times_per_month  milk_serve_per_month  soda_times_per_month  ju
27082         7                       1                       2                     2                     0

Predicted cigarettes_per_day_bin: 2.926%
       language  cereal_serve_per_month  cereal_times_per_month  milk_serve_per_month  soda_times_per_month  ju
27082         8                       1                       2                     2                     0

Difference between predictions
[0.         0.00901175 0.09764385 0.06409264 0.         0.
 0.        ]
```

```python
# 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')
```

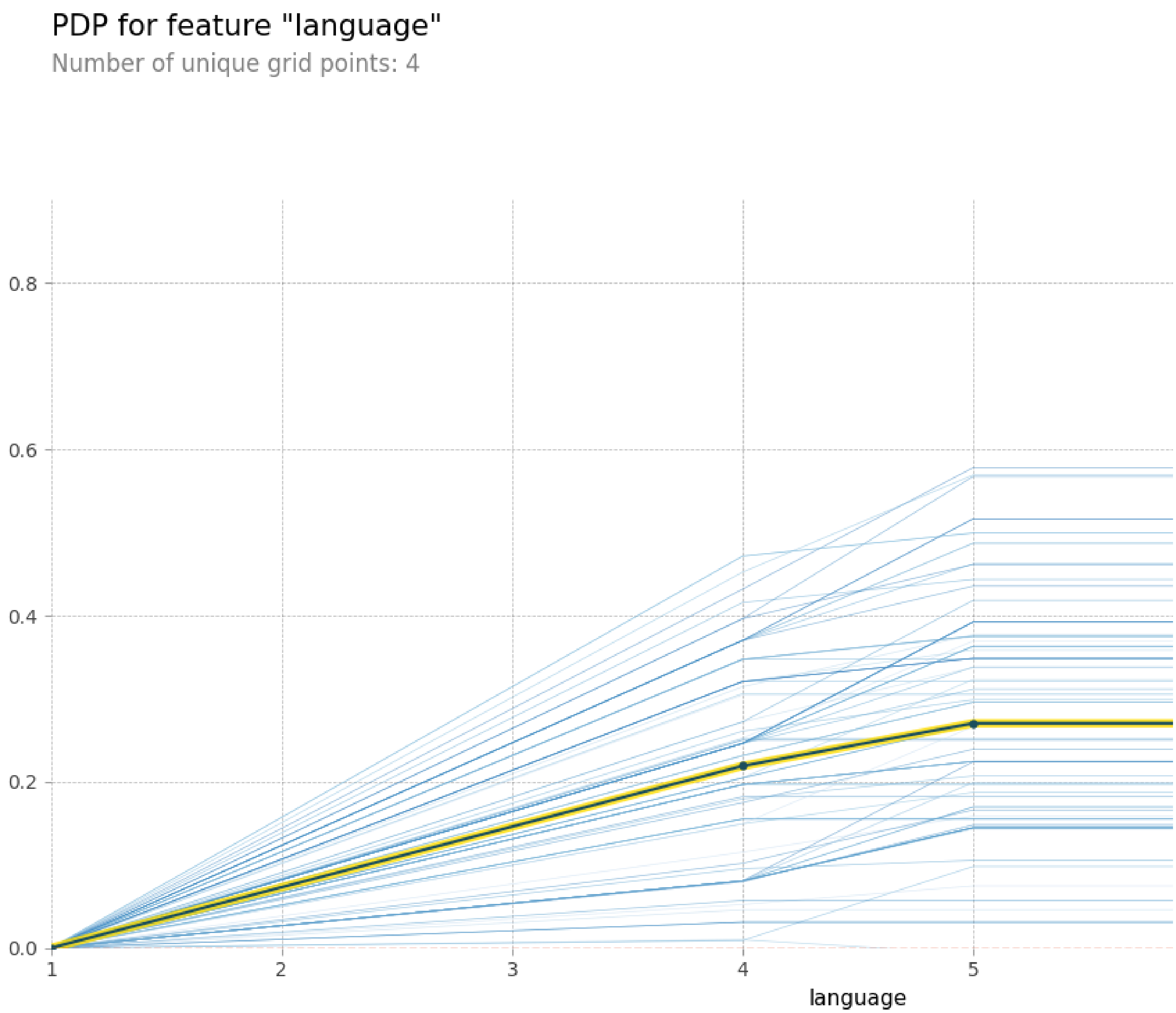



```
# Create patrial dependence plots with one feature
import matplotlib.pyplot as plt
! pip install PDPbox

# 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);
```

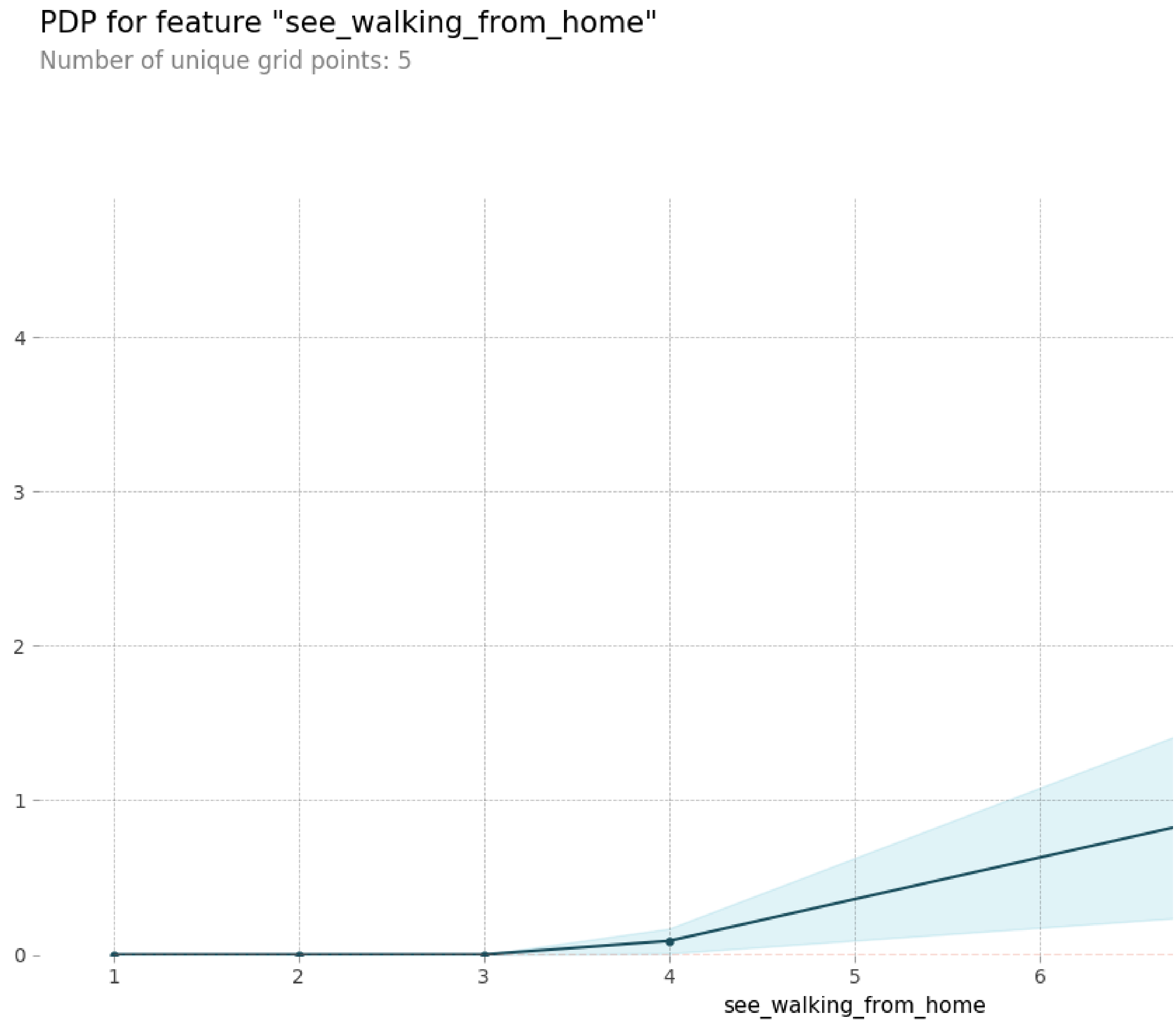
```
Collecting PDPbox
  Downloading https://files.pythonhosted.org/packages/87/23/ac7da5ba1c6c03a87c412e7e7b6e91a10d6ecf4474906c3e736
     |████████████████████████████████| 57.7MB 1.2MB/s
Requirement already satisfied: pandas in /usr/local/lib/python3.6/dist-packages (from PDPbox) (0.24.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from PDPbox) (1.16.5)
Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packages (from PDPbox) (1.3.1)
Requirement already satisfied: matplotlib>=2.1.2 in /usr/local/lib/python3.6/dist-packages (from PDPbox) (3.0.3
Requirement already satisfied: joblib in /usr/local/lib/python3.6/dist-packages (from PDPbox) (0.14.0)
Requirement already satisfied: psutil in /usr/local/lib/python3.6/dist-packages (from PDPbox) (5.4.8)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.6/dist-packages (from PDPbox) (0.21.3)
Requirement already satisfied: pytz>=2011k in /usr/local/lib/python3.6/dist-packages (from pandas->PDPbox) (201
Requirement already satisfied: python-dateutil>=2.5.0 in /usr/local/lib/python3.6/dist-packages (from pandas->P
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.6/dist-packages (from matplotlib>=2.1.2->
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.6/dist-packag
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib>=2.
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dist-packages (from python-dateutil>=2.5.0-
Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-packages (from kiwisolver>=1.0.1->ma
Building wheels for collected packages: PDPbox
  Building wheel for PDPbox (setup.py) ... done
  Created wheel for PDPbox: filename=PDPbox-0.2.0-cp36-none-any.whl size=57690723 sha256=3a1302daad4c5b733f38f1
  Stored in directory: /root/.cache/pip/wheels/7d/08/51/63fd122b04a2c87d780464eeffb94867c75bd96a64d500a3fe
Successfully built PDPbox
Installing collected packages: PDPbox
Successfully installed PDPbox-0.2.0
```



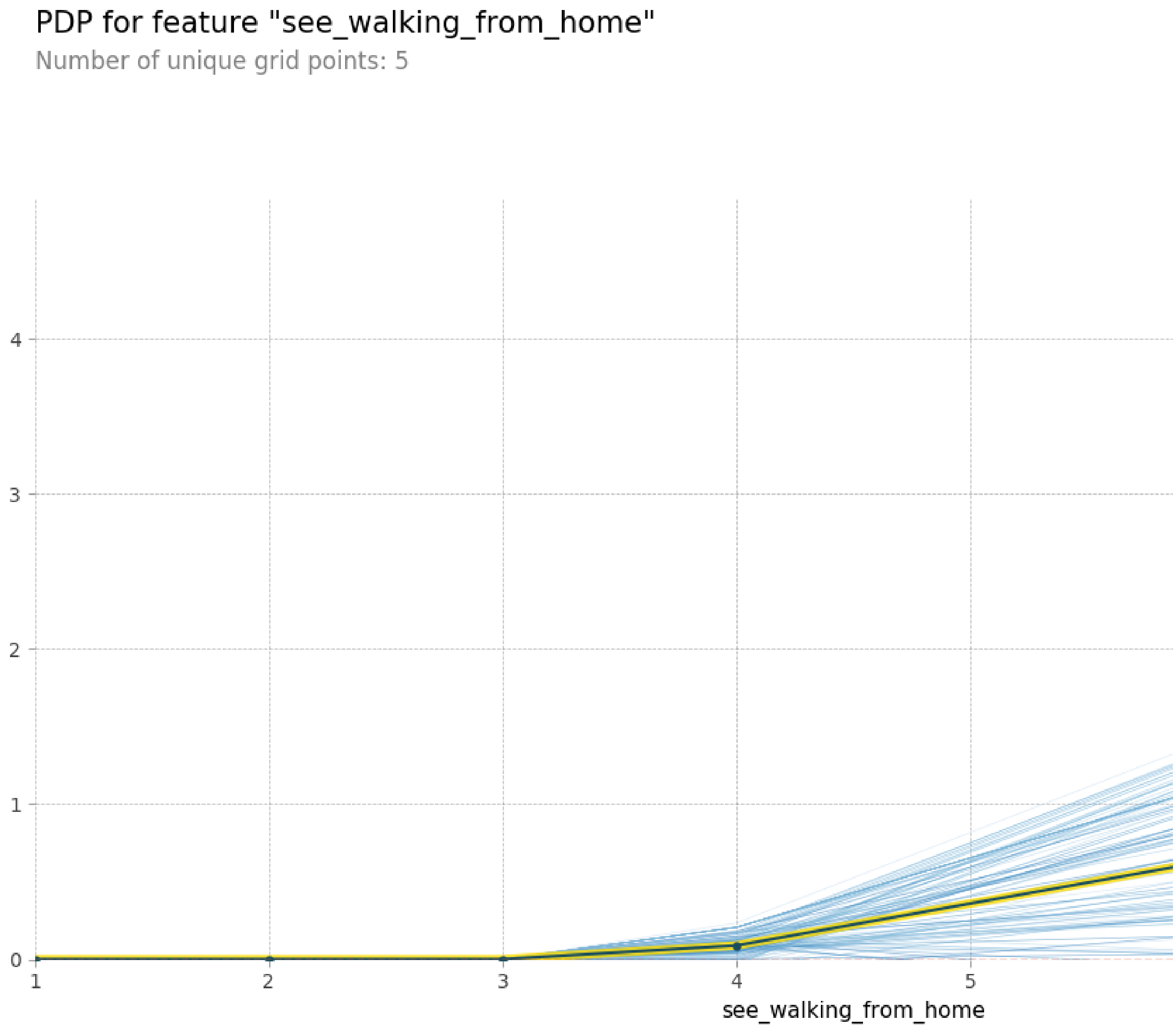
```
# Plot partial dependence plot with ICE lines for the language feature
pdp_plot(isolated, feature_name=feature, plot_lines=True, frac_to_plot=100) # Plot 100 ICE lines
plt.xlim(1,8);
```



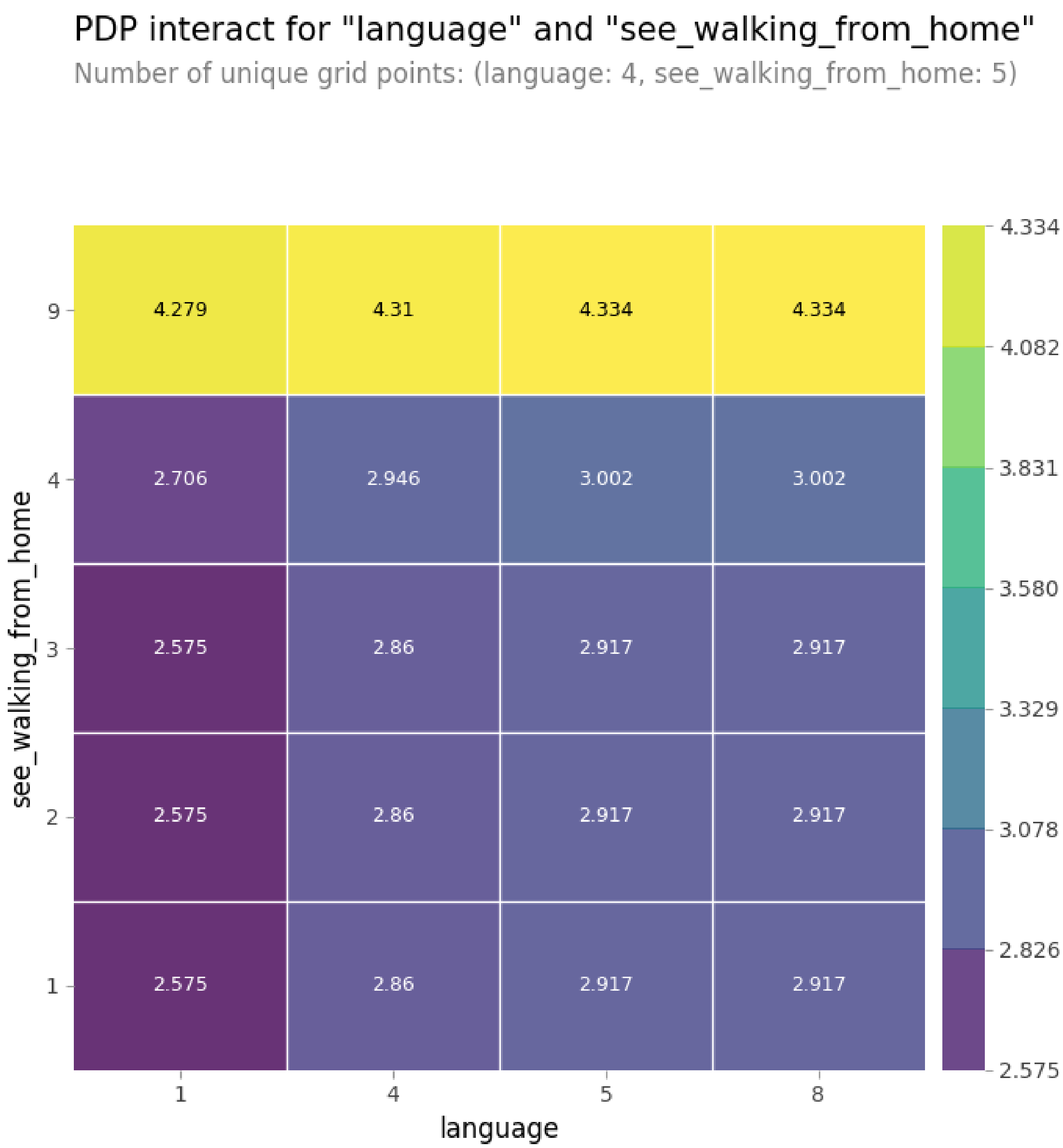

```
# First for the see_walking_from_home feature
plt.rcParams['figure.dpi'] = 100
from pdpbox.pdp import pdp_isolate, pdp_plot
feature = 'see_walking_from_home'
isolated = pdp_isolate(
    model=gb,
    dataset=X_val,
    model_features=X_val.columns,
    feature=feature
)

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



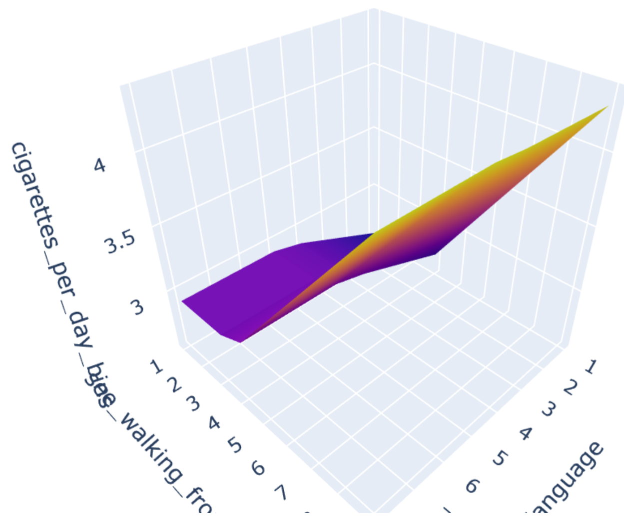

```
# Plot partial dependence plot with ICE lines for the see_walking_from_home feature
pdp_plot(isolated, feature_name=feature, plot_lines=True, frac_to_plot=100) # Plot 100 ICE lines
plt.xlim(1,8);
```




```
# Partial Dependence Plots with 2 features
from pdpbox.pdp import pdp_interact, pdp_interact_plot

features = ['language', 'see_walking_from_home']
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);
```



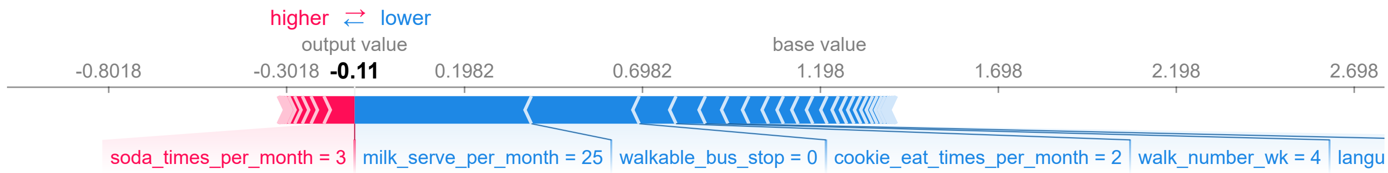

```
# A two feature partical dependence plot in 3D
pdp = interaction.pdp.pivot_table(
    values='preds',
    columns=features[0],
    index=features[1]
)[::-1] # Slice notation to reverse index order so y axis is ascending

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()
```



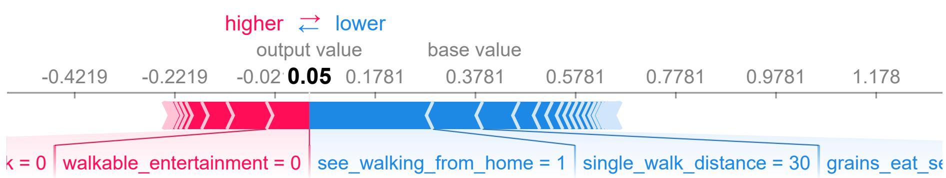

```python
# Test ROC AUC
from sklearn.metrics import roc_auc_score
from sklearn.impute import SimpleImputer
from sklearn.pipeline import make_pipeline
from xgboost import XGBClassifier
import category_encoders as ce

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)



# 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
)
```

```
Collecting shap==0.23.0
  Downloading https://files.pythonhosted.org/packages/60/0d/8bd076821f7230edb2892ad982ea91ca25f2f925466563272e6
     |████████████████████████████████| 184kB 9.5MB/s
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from shap==0.23.0) (1.16.5)
Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packages (from shap==0.23.0) (1.3.1)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.6/dist-packages (from shap==0.23.0) (0.21
Requirement already satisfied: matplotlib in /usr/local/lib/python3.6/dist-packages (from shap==0.23.0) (3.0.3)
Requirement already satisfied: pandas in /usr/local/lib/python3.6/dist-packages (from shap==0.23.0) (0.24.2)
Requirement already satisfied: tqdm in /usr/local/lib/python3.6/dist-packages (from shap==0.23.0) (4.28.1)
Requirement already satisfied: ipython in /usr/local/lib/python3.6/dist-packages (from shap==0.23.0) (5.5.0)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.6/dist-packages (from scikit-learn->shap=
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib->sh
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.6/dist-packages (from matplotlib->shap==0
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib-
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.6/dist-packag
Requirement already satisfied: pytz>=2011k in /usr/local/lib/python3.6/dist-packages (from pandas->shap==0.23.0
Requirement already satisfied: simplegeneric>0.8 in /usr/local/lib/python3.6/dist-packages (from ipython->shap=
Requirement already satisfied: prompt-toolkit<2.0.0,>=1.0.4 in /usr/local/lib/python3.6/dist-packages (from ipy
Requirement already satisfied: pickleshare in /usr/local/lib/python3.6/dist-packages (from ipython->shap==0.23.
Requirement already satisfied: decorator in /usr/local/lib/python3.6/dist-packages (from ipython->shap==0.23.0)
Requirement already satisfied: pexpect; sys_platform != "win32" in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.6/dist-packages (from ipython->shap==
Requirement already satisfied: traitlets>=4.2 in /usr/local/lib/python3.6/dist-packages (from ipython->shap==0.
Requirement already satisfied: pygments in /usr/local/lib/python3.6/dist-packages (from ipython->shap==0.23.0)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from cycler>=0.10->matplotlib->sh
Requirement already satisfied: wcwidth in /usr/local/lib/python3.6/dist-packages (from prompt-toolkit<2.0.0,>=1
Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.6/dist-packages (from pexpect; sys_pla
Requirement already satisfied: ipython-genutils in /usr/local/lib/python3.6/dist-packages (from traitlets>=4.2-
Building wheels for collected packages: shap
  Building wheel for shap (setup.py) ... done
  Created wheel for shap: filename=shap-0.23.0-cp36-cp36m-linux_x86_64.whl size=235685 sha256=18c38da919862d09f
  Stored in directory: /root/.cache/pip/wheels/c1/2c/aa/10d1782fe066536fcd564a2f8adea4dd05f57768236038855b
Successfully built shap
Installing collected packages: shap
Successfully installed shap-0.23.0
Collecting shap
  Downloading https://files.pythonhosted.org/packages/2b/4b/5944c379c94f8f6335dd36b9316292236e3da0dee8da806f60a
     |████████████████████████████████| 266kB 9.0MB/s
Collecting numpy (from shap)
  Downloading https://files.pythonhosted.org/packages/0e/46/ae6773894f7eacf53308086287897ec568eac9768918d913d5b
     |████████████████████████████████| 20.0MB 50.0MB/s
Collecting scipy (from shap)
  Downloading https://files.pythonhosted.org/packages/29/50/a552a5aff252ae915f522e44642bb49a7b7b31677f9580cfd11
     |████████████████████████████████| 25.2MB 1.3MB/s
Collecting scikit-learn (from shap)
  Downloading https://files.pythonhosted.org/packages/a0/c5/d2238762d780dde84a20b8c761f563fe882b88c5a5fb03c0565
     |████████████████████████████████| 6.7MB 43.4MB/s
Collecting pandas (from shap)
  Downloading https://files.pythonhosted.org/packages/86/12/08b092f6fc9e4c2552e37add0861d0e0e0d743f78f1318973ca
     |████████████████████████████████| 10.4MB 34.9MB/s
Collecting tqdm>4.25.0 (from shap)
  Downloading https://files.pythonhosted.org/packages/e1/c1/bc1dba38b48f4ae3c4428aea669c5e27bd5a7642a74c8348451
     |████████████████████████████████| 61kB 25.4MB/s
Collecting joblib>=0.11 (from scikit-learn->shap)
  Downloading https://files.pythonhosted.org/packages/8f/42/155696f85f344c066e17af287359c9786b436b1bf86029bb341
     |████████████████████████████████| 296kB 42.6MB/s
Collecting pytz>=2017.2 (from pandas->shap)
  Downloading https://files.pythonhosted.org/packages/e7/f9/f0b53f88060247251bf481fa6ea62cd0d25bf1b11a87888e53c
     |████████████████████████████████| 512kB 56.8MB/s
Collecting python-dateutil>=2.6.1 (from pandas->shap)
  Downloading https://files.pythonhosted.org/packages/41/17/c62faccbfbd163c7f57f3844689e3a78bae1f403648a6afb1d6
     |████████████████████████████████| 235kB 48.0MB/s
Collecting six>=1.5 (from python-dateutil>=2.6.1->pandas->shap)
  Downloading https://files.pythonhosted.org/packages/73/fb/00a976f728d0d1fecfe898238ce23f502a721c0ac0ecfedb80e
Building wheels for collected packages: shap
  Building wheel for shap (setup.py) ... done
  Created wheel for shap: filename=shap-0.31.0-cp36-cp36m-linux_x86_64.whl size=375005 sha256=530f855c4f72a4b58
  Stored in directory: /root/.cache/pip/wheels/7b/2d/46/ff8959add2e4e99a18a6e90b82f47508bf52fdf7e7d806f7df
Successfully built shap
ERROR: google-colab 1.0.0 has requirement pandas~=0.24.0, but you'll have pandas 0.25.2 which is incompatible.
ERROR: datascience 0.10.6 has requirement folium==0.2.1, but you'll have folium 0.8.3 which is incompatible.
```

```
ERROR: datascience 0.10.6 has requirement folium==0.2.1, but you'll have folium 0.8.3 which is incompatible.
ERROR: albumentations 0.1.12 has requirement imgaug<0.2.7,>=0.2.5, but you'll have imgaug 0.2.9 which is incomp
Installing collected packages: numpy, scipy, joblib, scikit-learn, pytz, six, python-dateutil, pandas, tqdm, sh
Successfully installed joblib-0.14.0 numpy-1.17.3 pandas-0.25.2 python-dateutil-2.8.0 pytz-2019.3 scikit-learn-
```




```python
# Contributrions to making bin 8 (49 - more 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
)
```



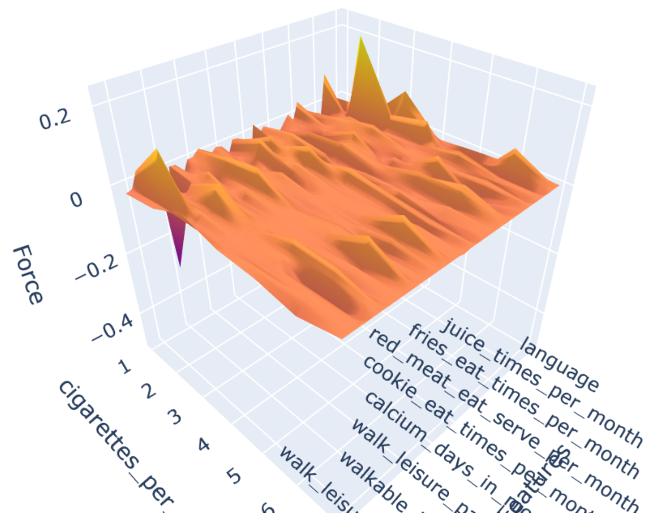

```python
# Featues importances for sample 170

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));
```


(soda_times_per_month, 3.0)
(fries_eat_times_per_month, 3.0)
(cigar_even_once, 0.0)
(donut_eat_times_per_month, 3.0)
(single_walk_distance, 30.0)
(grains_eat_serve_per_month, 1.0)
(cereal_times_per_month, 3.0)
(processed_meat_eat_times_per_month, 3.0)
(multivitamin_past_month, 1.0)
(vitD_days_in_month, 0.0)
(walk_leisure_past_wk, 0.0)
(vitamin_past_month, 1.0)
(see_walking_from_home, 1.0)
(walkable_relaxation, 1.0)
(walk_leisure_number_wk, 0.0)
(walk_leisure_ time, 0.0)
(salad_eat_times_per_month, 2.0)
(calcium_days_in_month, 0.0)
(traffic_discourages_walking, 0.0)
(walkable_entertainment, 0.0)
(red_meat_eat_times_per_month, 3.0)
(grains_eat_times_per_month, 3.0)
(ice_cream_eat_times_per_month, 3.0)
(vitD_reason, 0.0)
(bread_eat_serve_per_month, 3.0)
(beans_eat_serve_per_month, 1.0)
(walk_leisure_distance_week, 0.0)
(coffee_times_per_month, 3.0)
(sports_drink_times_per_month, 0.0)
(tobacco_even_once, 0.0)
(cheese_eat_serve_per_month, 4.0)
(multivitamin_days_in_month, 30.0)
(juice_times_per_month, 3.0)
(walk_leisure_ distance, 0.0)
(red_meat_eat_serve_per_month, 6.0)
(fries_eat_serve_per_month, 3.0)
(cereal_serve_per_month, 25.0)
(language, 5.0)
(walk_number_wk, 4.0)
(cookie_eat_times_per_month, 2.0)
(walkable_bus_stop, 0.0)
(milk_serve_per_month, 25.0)
−0.5
−0.4
−0.3
−0.2
−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
df_bins = pd.DataFrame(columns=np.array(feature_names), data=my_python_list)

df_bins.head(8)
```

	language	cereal_serve_per_month	cereal_times_per_month	milk_serve_per_month	soda_times_per_month	juice
0	-0.063460	-0.058806	0.006714	-0.496689	0.086444	
1	-0.017955	-0.160864	0.016525	0.224091	-0.046407	
2	0.087442	0.047602	-0.038343	0.036599	-0.008106	
3	-0.011019	0.011034	-0.000203	0.059331	-0.002250	
4	-0.001570	-0.071335	-0.009197	-0.127074	0.025907	
5	0.005110	0.059201	-0.000520	-0.101339	0.001079	
6	0.000000	0.000542	0.000000	0.000000	0.000000	

```
# 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()
```

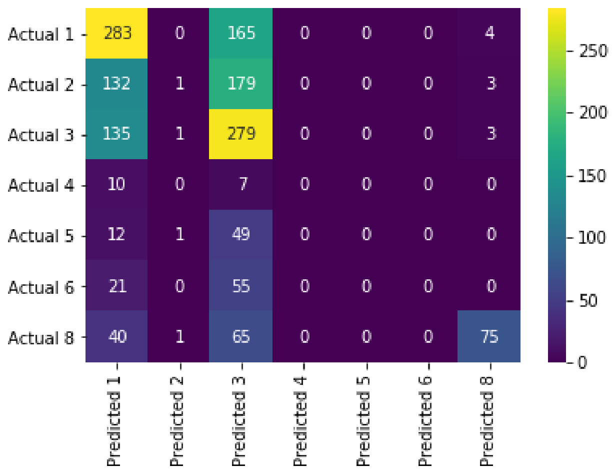



```python
pros = shaps.sort_values(ascending=False)[:3].index
cons = shaps.sort_values(ascending=True)[:3].index

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. fries_eat_times_per_month is 3.0
3. cigar_even_once is 0.0


Cons:
1. milk_serve_per_month is 25.0
2. walkable_bus_stop is 0.0
3. cookie_eat_times_per_month is 2.0
```

```python
# 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')
```

```python
y_pred = pipeline0.predict(X_val)
plot_confusion_matrix(y_val, y_pred);
```




```python
# 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       0.45      0.63      0.52       452
           2       0.25      0.00      0.01       315
           3       0.35      0.67      0.46       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       0.88      0.41      0.56       181

    accuracy                           0.42      1521
   macro avg       0.28      0.24      0.22      1521
weighted avg       0.39      0.42      0.35      1521
```

```python
# Another way to get a classification report using an ROC_AUC approach (https://stackoverflow.com/questions/39685740/calcula
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
```

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

precision recall f1-score support pred

```python
# 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  0.079808  0.217352  0.328526  0.032976  0.167012  0.070402  0.032201
1  0.189750  0.215191  0.335306  0.048586  0.057663  0.058306  0.046568
2  0.159886  0.216339  0.327539  0.039561  0.086386  0.050499  0.037919
3  0.030064  0.086333  0.076470  0.028838  0.031843  0.028758  0.028475
4  0.227320  0.196754  0.311454  0.039435  0.056393  0.056143  0.037474

          7
0  0.071723
1  0.048630
2  0.081869
3  0.689219
4  0.075027
```

```python
# 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
```

```python
# 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()
```